

École polytechnique de Louvain

Salamander Individual Identification for population tracking

Author: **Sarra LAKSACI**

Supervisors: **Olivier BONAVENTURE, Sébastien JODOGNE**

Readers: **Edouard CHATZOPOULOS,**

Academic year 2022–2023

Master [120] in Computer Science

Abstract

This thesis addresses the research question of individual identification within the context of fire salamanders. Our experimental framework centers on the extraction of their unique identity patterns, notably the yellow spot formations present on their back, as a first step. We then explore two different methods for the identification task: the first one relies on Local Binary Patterns as a descriptor for the identity pattern, while the second employs a patch-wise approach leveraging image moments as a feature. By focusing on handcrafted features, our research aims to deepen the understanding of this complex task.

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Sébastien Jodogne and Prof. Olivier Bonaventure. Their guidance, advice, and time have been invaluable. This research journey on salamander identification was filled with challenges and surprises, and their support and insightful suggestions have been constant motivators throughout this experimental work.

I would also like to express my appreciation to Renaud Lafourt for generously sharing his dataset, and to Louis Uytterhaegen for his dedicated assistance during his internship. Additionally, I extend my thanks to Adrien Goffin for dedicating his time to test aspects of this work, for his detailed answers to my questions and his contribution to the custom dataset presented in this thesis.

In addition, I am grateful to Edouard Chatzopoulos for agreeing to review this work. I appreciate Lucile Diercks for her insightful comments and suggestions during our thesis meetings, and Amaury Fierens for his valuable feedback on the concluding chapters of this document.

Finally, I would like to thank my family and friends for their support and encouragement.

Contents

1	Introduction	10
2	Salamanders	13
2.1	Information on Salamanders	13
2.2	Tracking Salamanders	14
3	Theoretical background	16
3.1	Convolutional Neural Network (CNN)	16
3.2	K-means Clustering	18
3.3	K-Nearest Neighbors	19
3.4	Support Vector Machine (SVM)	20
3.5	Metrics	21
3.6	Losses	22
3.7	Local features	23
3.7.1	Local Binary Patterns	23
3.7.2	Image Moments	23
3.8	Color spaces	24
3.9	Mathematical morphology	25
4	Dataset	26
4.1	Segmentation dataset	26
4.2	Identification dataset	29
4.3	Conclusion	32
5	Segmentation	33
5.1	U-Net segmentation	33
5.1.1	Input data	34
5.1.2	Architecture	35
5.1.3	Hyper-parameters	37
5.1.4	Results	39
5.1.5	Post-processing	42

5.2	K-means clustering	43
5.2.1	Input data	43
5.2.2	Results	46
5.2.3	Post-processing	47
6	Identification	49
6.1	Local Binary Patterns	50
6.1.1	Circular Local Binary Pattern	51
6.1.2	Uniform Local Binary Pattern	52
6.1.3	Histogram	53
6.1.4	Distance measures	53
6.1.5	Results	54
6.2	Body Alignment	58
6.2.1	Affine Image Transformations	59
6.2.2	Curved salamanders	61
6.3	Patch-wise approach	65
6.3.1	Feature Space	66
6.3.2	Classifier	67
6.3.3	Results	68
6.4	Conclusion	73
7	Results and analysis	75
7.1	Evaluation metrics	75
7.2	Closed set evaluation and discussion	75
7.3	Open set evaluation and discussion	77
7.4	Conclusion	78
8	Conclusion	80
A	Additional processing	86
B	Volunteer Evaluation and Insights: Testing the Segmentation Model	88
B.1	Context	88
B.2	Segmentation results	88

List of Figures

2.1	Caption	13
4.1	Examples of excluded samples:	27
4.2	Examples of our custom dataset image categories.	28
4.3	Mandermatcher labeling interface.	30
4.4	Images of most frequent individual: ID30.	31
4.5	Frequency distribution of occurrences per individual.	31
5.1	Example of a narrow image sample after resizing.	34
5.2	U-Net architecture. From [31].	36
5.3	Training and validation dice coefficient and dice loss.	39
5.4	Visualization of U-Net segmentation results. (a) original image sample, (b) segmented image, (c) predicted mask and (d) original mask.	40
5.5	Two salamanders samples results. (a) original image sample, (b) segmented image, (c) predicted mask and (d) original mask.	41
5.6	Example of flawed prediction with head hole.	42
5.7	Examples of corrected masks.	42
5.8	Examples of connected legs closing.	43
5.9	Visualization of the 3D RGB space for a salamander body.	44
5.10	K-means color segmentation in RGB space.	45
5.11	Visualization of K-means segmentation results. (a) U-Net segmented salamander in RGB space, (b) U-Net segmented salamander in CIELAB space, (c) K-means segmented spots on CIELAB image.	46
5.12	Example of small noise in the segmented spots.	48
5.13	K-means segmentation before and after opening operation.	48
6.1	Feature extraction workflow using Local Binary Patterns.	51
6.2	(a) The basic LBP operator From [2]. (b) The CLBP neighborhood.	52
6.3	Impact of different radii on CLBP encoded image (nb_points= 8)	55
6.4	Impact of different nb_points on CLBP encoded image (radius= 3)	55

6.5	Impact of different radii on U-CLBP encoded image (nb_points= 8)	55
6.6	Impact of different nb_points on U-CLBP encoded image (radius= 3)	55
6.7	Example of feature histograms associated to CLBP encoded images. ($r=2$; $nb_points= 16$).	56
6.8	Example of feature histograms associated to U-CLBP encoded im- ages. ($r=2$; $nb_points= 16$).	56
6.9	Results of the comparison experiment.	57
6.10	Example of goal salamander normalization.	58
6.11	Affine transformation on salamander mask.	60
6.12	Affine transformation on salamander images.	61
6.13	Contours features for curvature detection.	62
6.14	Approximated curve and key points for curvature detection.	63
6.15	Example of curve detection.	64
6.16	Upper body rotation.	65
6.17	Construction of the bag-of-patches.	66
6.18	Visualization of image moments for pairs of similar patterns.	69
6.19	Visualization of image moments for a pair of patterns.	70
6.20	Visualization of patch neighbors with pixel features.	72
6.21	Visualization of patch neighbors based on image moments features only.	72
7.1	Comparison of query image to voted identities.	76
7.2	Variant of comparison experiment.	78
A.1	Examples of glare correction.	87
A.2	Example of glare correction introducing spot alterations.	87
B.1	Fire-salamander category.	89

Chapter 1

Introduction

Fire Salamanders (*salamandra salamandra*) are amphibians characterized by their lizard-like appearance and vibrant spots, found across central Europe. In Belgium, fire salamanders are part of Wallonia’s red list of species [4], thus piquing the interest and concern within the community of amphibian observers and volunteers. This species might face the risk of population decline due to habitat alterations and loss, driven by deforestation and pollution in addition to the threat of a deadly fungus. Moreover, these creatures, known for their heightened sensitivity to environmental changes, serve as valuable bio-indicators. Given these factors, the task of tracking salamanders assumes considerable importance.

The field of research for such a problem is animal individual identification. Researchers worldwide closely examine the behavior of various creatures to track their movements and estimate population sizes. At the heart of this effort lies the Capture-Mark-Recapture (CMR) method. CMR involves capturing a subset of individuals, marking them and then releasing them back into their habitat. Upon re-encounter, the marked individuals provide a basis for estimating the total population size through statistical methods.

In the case of wildlife, camera traps are gaining in popularity, gradually replacing traditional marking methods like GPS collars and physical tags. This shift is driven by the advantages of monitoring entire populations, rather than a fixed set of tagged individuals. As the use of camera traps accumulates animal footage, the appeal of machine learning techniques for automated identification is growing. Computer vision techniques, which are increasingly common for addressing biological questions, are being applied. However, the integration of automated machine learning techniques has not yet yielded significant progress in the case of fire salamanders.

In the past, tracking methods for amphibians involved physical marking, such as toe-clipping [28]. The invasive nature of this approach, which required cutting a portion of the animal’s digit, encouraged the exploration of non-intrusive alternatives. One initial solution involved manual inspection of photographs followed by labeling.

In more recent times, the observation and tracking of salamander populations in Belgium have, for some part of the community, relied on software to ease the tedious tasks involved. One such software is ManderMatcher [35], which involves manually encoding the spots of individual salamanders. This manual map produced is then used to compare with previously stored maps in the database, without utilizing computer vision in its process. Another software, known as WildID [39], is employed by certain volunteers. Initially designed as a tool for processing trap camera images, this software offers useful features that were originally tailored for identifying giraffe patterns. However, it still necessitates significant manual processing before initiating its analysis. Additionally, it shows limitations as a result of the obvious differences between fire salamanders and giraffe patterns. As for the question of automatic individual identification for the population tracking of salamanders, no answer has been found so far. Francois Duchene’s master thesis on the topic [6] explored the method of blobs and centroids. Despite a promising hypothesis about the extraction of identity patterns from the collected photographs as a preliminary step to the identification method employed, this approach fell short of achieving the anticipated results.

The goal of this work is to carry forward the momentum of previous experimental work done on the topic, by conserving the idea of preliminary extraction of animal patterns before delving into identification. It is important to highlight that throughout this research, a focal point has been on providing a solution that takes as input raw volunteer images as uploaded on platforms such as Observations.be. Moreover, our preliminary hypothesis revolves around exclusive reliance on machine learning techniques to fulfill the identification task. In that context, we have investigated numerous potential paths with the objective of solving the question of computer vision automated individual identification of salamanders.

This thesis’s structure is organized into eight chapters. Chapter 2 gives general information about fire salamanders, followed by a description of the most used tracking method by Belgian volunteers. Chapter 3 focuses on the theoretical background required to understand the following chapters. It introduces different computer vision concepts and machine learning algorithms used in this work. Chapter 4 gives information about the dataset that was assembled in the context

of this thesis. Chapter 5 presents the work done for the extraction of the identity patterns. It includes two parts: the segmentation of the animal body and the segmentation of the animal's spots. Chapter 6 presents the different approaches explored for the identification task. Namely the Local Binary Patterns approach for which we highlight the encountered limitation and the patch-wise proposed approach. In chapter 7, the results of such approach will be discussed. Finally, chapter 8 concludes this work with further discussions on the topic of salamander individual identification and future work.

Chapter 2

Salamanders

The focus of this thesis being fire salamanders, it is important to first understand the characteristics that play a central role in their identification as well as the reasons that make population tracking important for this species. This chapter begins with general information on the animal, including physical characteristics and mobility behaviour, followed by a description of the most used tracking method by Belgian volunteers with a focus on the tracking protocol of Natagora association.



Figure 2.1: Caption

2.1 Information on Salamanders

Salamanders are an integral part of the biodiversity in Belgium, particularly in the wooded regions, primarily in deciduous forests. Salamanders in Belgium exhibit unique physical features, characterized by their sleek, cylindrical bodies, which appear blackish and are partially covered in yellow spots, occasionally orange, on the head, back, lower flanks, throat, and limbs. These spots can merge to form continuous bands on the back and flanks. The coloration of salamanders displayed through these spots serves as a warning signal of their toxicity, as they

possess venom in various skin glands. Aside from this warning purpose, the infinite combinations of spots across the whole body of the animal can serve as a great identity index. Indeed, the colored print reaches a stable and definitive pattern in adulthood which is unique to the individual. Adults generally measure between 15 and 20 cm in length ([17] [4]).

These fascinating creatures hold ecological significance and are considered a protected species in Wallonia, being classified as endangered. It is strictly prohibited to capture, disturb, or intentionally cause harm to individuals of these species in their natural habitats [4]. Salamanders' place in the red list of endangered species comes as a result of numerous threats of which the most important is the *Batrachochytrium salamandrivorans* (Bsal), an Asian pathogenic fungus that consumes the amphibian's skin causing death shortly after infection. Despite their ecological importance, tracking and observing salamanders can be challenging due to their nocturnal behavior. Adults tend to stay in close proximity to their breeding sites and lead terrestrial lives throughout the year, with females venturing into water only to deposit their larvae. The longevity of salamanders is noteworthy, as they can live up to 20 years in their natural habitat, contributing to their value as subjects for population tracking and observation.

2.2 Tracking Salamanders

Efforts to monitor salamander populations and understand the impact of the B.sal fungus which appeared in Belgium since 2013 have been undertaken by volunteers from Natagora Basse-Meuse ([19] [18][20] [21]). Volunteers are involved in salamander population surveys in the Argenteau region, conducting counts along a chosen forest route of approximately 1 km. These surveys take place at least twice a year, preferably in May and from September to October, during rainy and mild nights. The selected route traverses a forested area inhabited by a salamander population. Additionally, they typically follow a public path or trail. During the surveys, all observed amphibians on the designated transect are counted, and the total number of individuals for each species is recorded, regardless of age or sex. In addition to the mandatory autumn visits, two optional visits may be conducted in spring to provide a comprehensive understanding of the seasonal dynamics of salamander populations. Furthermore, salamanders are sometimes observed in large groups, numbering in the hundreds or even thousands, near forested areas close to roads. Tragically, a portion of these gatherings might fall victim to being crushed by vehicles as they traverse these roads, which fragment their natural habitats. In these areas, volunteers undertake observation activities to assist salamanders in safely crossing the roads.

Aside from the specifics protocol for observation nights along specific routes defined by Natagora's team, other volunteers follow a similar tracking methodology which consists in night walks in wooded regions and taking photographs upon encounter of an individual. These photographs are optionally uploaded to platforms such as Observation.be then carefully browsed to count the number of individuals and spot previously seen individuals. The latter requires sharp attention and long hours of manual labelling.

Chapter 3

Theoretical background

This chapter introduces various computer vision techniques and concepts applied in this project. These encompass the neural network family employed for the task, the machine learning algorithms—specifically, K-Means clustering and K-Nearest-Neighbors—as well as the metrics and losses associated with these approaches.

3.1 Convolutional Neural Network (CNN)

Neural networks [1] come in many variations depending on the task at hand and the type of data it receives as input. One of such kinds is the Convolutional Neural Network, also referred to as CNN, which holds a pivotal role in the realm of deep learning applied to visual data. This is because CNNs tackle various challenges that conventional feedforward networks encounter when processing images, with the most significant issue being the degradation of spatial information. Notably, CNNs consist of distinct types of layers, as we shall elucidate shortly, which facilitate the preservation of pixel positional details within two-dimensional data, such as images.

CNNs were initially engineered for the task of classification. In that context the bottom layers consist in convolutional layers and pooling layers which perform feature extraction while the top layers consist in fully connected layers that perform the final classification.

Convolutional layer

The convolutional layer aims to extract the essential information from the input image by applying filters or kernels through a convolutional operation. This operation can be simply described as the dot product of the kernel and the overlapping

sub-regions of the image as we slide the kernel across the entire image. The resulting matrix of this operation is a feature map which highlights discriminative aspects of the image. The success of this extraction lies in the choice of the filter for which the weights are defined upon training. The hyper-parameters at this stage include the padding, stride and kernel size in addition to the activation function. Padding involves adding zeros along the image’s periphery to maintain the same resolution for the feature map. Stride refers to the distance between two consecutive positions of the kernel. The activation function is applied to introduce non-linearity in the model enabling the extraction of more complex information. Common activation functions are the Rectified Linear Unit (ReLU) [3] and the Sigmoid [23] function. The definitions of these functions are displayed in Equations 3.1 and 3.2.

$$f(x) = \max(0, x) \quad (3.1)$$

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.2)$$

Pooling layer

The pooling layer, also referred to as the downsampling layer, aims to reduce the size of the computed feature map, simplifying subsequent computations while discarding unnecessary and redundant information. This downsampling operation can be executed in various ways, including MaxPooling. MaxPooling consists in sliding a window across the image and selecting the maximum of each sub-region. Additionally, there is the AveragePooling method that operates similarly but returns the average value from each sub-region instead of the maximum.

Dense layer

The dense layers reside at the end of the CNN architecture, producing the final output of the task. A stack of dense layers is preceded by a flattening operation, which transforms the feature map resulting from the last convolutional layer into a 1D vector. It is important to note that dense layers are not an inherent part of all CNNs. While they play a role in tasks like classification and object detection, they are omitted from CNNs used for semantic segmentation. Activation functions, introduced in the context of convolutional layers, are also present in these layers.

3.2 K-means Clustering

K-means [30] is a commonly used iterative clustering algorithm. It falls in the category of unsupervised learning as it relies only on input data and no labels. Clustering consists in dividing data into distinct groups known as clusters, each containing data points that share similarities. K-Means accomplishes this task by creating K clusters that gather the most similar points and separate different ones. Similarity and dissimilarity is computed through a distance measure such as the Euclidean distance. Each cluster has a centroid which is the mean of all points contributing to that particular group.

Bellow is the description of the algorithm:

Algorithm 1 K-Means Algorithm

Require: Data points X with n samples, Number of clusters K

- 1: Initialize cluster centroids randomly: $C \leftarrow$ randomly select K points from X
 - 2: **while** not converged **do**
 - 3: Assign each data point to the nearest centroid: $S_i \leftarrow \{x \in X \mid \operatorname{argmin}_{c_j \in C} \operatorname{distance}(x, c_j) = c_i\}$
 - 4: **for** $i = 1$ to K **do**
 - 5: Update centroid c_i as the mean of points in S_i : $c_i \leftarrow \frac{1}{|S_i|} \sum_{x \in S_i} x$
 - 6: **end for**
 - 7: **end while**
 - 8: **return** Cluster assignments S and cluster centroids C
-

K-Means clustering proves to be an effective approach for color segmentation. Indeed, the segmentation task consisting in assigning a class to each pixel of the image, we can interpret the pixels as the data points and the color space as the feature space. Within this context, the application of the K-Means algorithm with K equal to the desired color groups performs well. Formally, we want to find the best set of centroids C^{opt} such that each centroid represents a group of similar pixels which are the closest to it i.e. a segmented region.

Despite being an unsupervised method, it is still possible to assess the results of a clustering on given data. One possible measure to assess a K-Means clustering is the Silhouette score introduced in section 3.5

3.3 K-Nearest Neighbors

K-Nearest Neighbors [30], or KNN, is a supervised classification algorithm known as a lazy classifier. This naming aptly fits as creating a KNN model simply consists in storing the dataset. Subsequent predictions are executed by calculating the distance between a new instance and all dataset samples, ultimately returning the class most frequently represented among the K nearest neighbors. The simplicity of this approach emphasizes the significance of the choice of the feature space and the distance measure. Indeed for this approach to be effective, it is crucial to represent the samples in a feature space that accentuates the disparities between the classes while grouping similar samples. Additionally, a good distance measure is important to correctly quantify the similarities between two samples as it becomes a decisive factor when all K nearest neighbors are of distinct classes. The default distance measure for a KNN model is the Euclidean distance. Other distances such as Mahalanobis distance may be more suited depending on the data at hands.

Mahalanobis distance

Mahalanobis [29] distance is a statistical measure initially designed to identify similarities between skulls based on measurements. It computes the distance between a point and a distribution in a multi-dimensional space. It takes into account the correlations and variances among different dimensions, unlike the Euclidean distance which treats all dimensions independently and equally. However, the Mahalanobis distance approaches the standard Euclidean distance when the values are normalized.

$$D_M(\mathbf{x}, \boldsymbol{\mu}, \mathbf{C}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \quad (3.3)$$

Where $\mathbf{x}$ is the data point, $\boldsymbol{\mu}$ is the mean vector of the distribution, and $\mathbf{C}$ is the covariance matrix.

The `scipy.spatial.distance` [33] module has a built-in mahalanobis function that calculates the distance between a point and a distribution. It takes as input the vector of the point to be compared, along with the distribution's mean and the precomputed covariance matrix. To benefit from the advantages of the mahalanobis distance in the context of cluster analysis or classification with the KNN approach, it is possible to introduce it as the distance parameter of the `KNeighborsClassifier` from the `sklearn.neighbors` module.

3.4 Support Vector Machine (SVM)

The Support Vector Machine [30], SVM in short, is a classification algorithm originally designed for binary classification tasks. In such cases, it works towards separating the two classes within the feature space. In a linearly separable case the objective is to select a hyperplane that maximizes the margin; hence, it's also referred to as a "Large Margin Classifier". The name "Support Vector Machines" originates from its reliance on just three data points, known as support vectors, at all stages to define the separator. Predictions for new samples are made based on the side of the hyperplane the sample is on. SVMs are still applicable in the case of non-linearly separable data. The solution is to elevate the data to a higher dimensional space where it becomes linearly separable. The kernel trick facilitates this by efficiently simulating higher-dimensional results. Some commonly used kernels include polynomial, radial basis function "RBF" and sigmoid kernels.

Formally, finding the best suited hyperplane to separate the data points is expressed as the optimization problem in Equation 3.4 where w and b are respectively the weights and bias that define the hyperplane. Once the optimal parameters for the hyperplane are found, the classification of the data points is performed as outlined in Equation 3.5

In scenarios involving more than two classes, the classification process using SVMs involves reducing the multi-class problem into several binary classifications. This can be done through a one-vs-all method, which constructs binary classifiers for each class. In the context of N different classes, we would have N SVM classifiers identifying their designated class as positive and all other classes as negative. Upon prediction, the final predicted class would be the one of the classifier with the higher positive prediction score. Another method is the one-vs-one multi-class SVM for which the prediction is made through a max-wins voting strategy.

$$\begin{cases} \text{Minimize} & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{Such that} & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \text{for all training samples } (\mathbf{x}_i, y_i) \end{cases} \quad (3.4)$$

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \quad (3.5)$$

3.5 Metrics

In this subsection we will introduce metrics that can be used to assess the previously presented algorithms for the three tasks anticipated in this work: semantic segmentation, clustering and classification.

Accuracy

The accuracy [7] is a metric that represents the portion of correctly assigned labels over the whole dataset. Equation 3.6 shows the definition of the accuracy metric for a binary classification where true positives (TP) and true negatives (TN) are correctly classified samples, and false positives (FP) and false negatives (FN) are erroneous classifications. In the more general case with N classes the definition becomes: number of correct predictions over total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.6)$$

Dice coefficient

Dice coefficient is a metric employed in the task of semantic segmentation. It is defined by Equation 5.1 where A and B are two sets. It computes the intersection between the two sets and divides it by their sizes to quantify the overlap. In semantic segmentation A and B would represent the true and predicted masks.

$$\text{Dice Coefficient} = \frac{2 \cdot |A \cap B|}{|A| + |B|} \quad (3.7)$$

Silhouette score

The silhouette score [36] allows us to evaluate the effectiveness of a clustering algorithm by calculating the cohesion within clusters and separation between them. It thus quantifies the quality of the clusters formed. By definition, a well-performing clustering should exhibit tight cohesion within each group, implying that data points within the same cluster are closely grouped together. Simultaneously, there should be substantial separation between different clusters, implying that data points from distinct groups are farther apart. Equation 3.8 presents the Silhouette Score where n is the total number of samples, $a(i)$ represents the average distance of data point i to other points within the same cluster, and $b(i)$ represents the smallest average distance of data point i to points in other clusters.

$$\text{Silhouette Score} = \frac{1}{n} \sum_{i=1}^n \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (3.8)$$

3.6 Losses

The selection of a loss function plays a pivotal role in training a neural network and determining its effectiveness. It measures the disparity between the objective aimed for and the achieved result. In the context of semantic segmentation, three prominent loss functions are commonly employed: the Binary Cross Entropy, the Dice Loss, and the Jaccard Loss, which are discussed in the following subsections. These loss functions serve as critical tools for guiding the network’s optimization process and aligning its predictions with the desired segmentation outputs.

Binary Cross Entropy

Binary Cross Entropy [12], or BCE in short, is a well known loss in deep learning for binary tasks. It measures the dissimilarity between the predicted probabilities and the true probabilities. It is defined in Equation 3.9 where $\hat{y}$ is the predicted value. Categorical Cross Entropy is variant of BCE for multi-class tasks formulated in Equation 3.10 with N as the total number of samples, p_i being the probability of belonging to class i and y_i being a binary indicator equal to 1 if i is the correct classification.

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \quad (3.9)$$

$$\text{Categorical Cross-Entropy Loss} = - \sum_{i=1}^N y_i \cdot \log(p_i) \quad (3.10)$$

Dice Loss

Building upon the Dice Coefficient metric is the Dice Loss [12]. This loss is widely used in computer vision community as it is both simple and intuitive. Its mathematical formulation is presented in Equation 3.11.

$$\mathcal{L}_{\text{Dice}}(A, B) = 1 - \frac{2|A \cap B|}{|A| + |B|} \quad (3.11)$$

Jaccard Loss

The Jaccard Loss, as we can see in Equation 3.12, is similar to the Dice Loss in its form. It quantifies the loss by subtracting the intersection over the union. The subtracted quantity, which is called the Jaccard index or IoU score, computes how important the overlap is between the true pixel mask and the predicted mask of segmentation.

$$\mathcal{L}_{\text{Jaccard}}(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \quad (3.12)$$

3.7 Local features

Considering our initial hypothesis that machine learning methods will be employed for the identification task, our primary focus lies in performing effective feature extraction prior to introducing machine learning algorithms. With this goal in mind, the pivotal question revolves around identifying discriminative features within our images. By doing so, we aim to ensure that the subsequent classification process yields accurate and appropriate results.

3.7.1 Local Binary Patterns

Local Binary Patterns (LBP) [24] is a texture descriptor designed to effectively capture texture variations within an image region. This descriptor operates on gray scale images by comparing pixel intensities within a specified neighborhood in relation to the central pixel. The method generates a binary vector by concatenating the outcomes of pixel intensity comparisons in a clockwise traversal of the region. The binary code is then transformed into a decimal value, resulting in a numerical score. Consequently, a texture can be represented by one of several potential values within a 3x3 region. LBP stands out in the classification of patterns and prints due to its ability to accurately capture intricate textures, facilitating successful discrimination. Many variants of LBP exist to adapt to various print characteristics.

3.7.2 Image Moments

Image Moments are descriptors constructed from weighted averages of pixel intensities, known as moments. These moments serve to characterize various properties of images, including objects following segmentation, as well as features related to contours in binary images, such as centroid and orientation. Image Moments are categorized into three groups: (i) Spatial Moments: These encapsulate raw moments and quantify the distribution of pixel intensities. (ii) Central Moments: These moments are computed with respect to the centroid $(\bar{x}, \bar{y})$ of the image. They offer insights into the arrangement of pixel intensities around the centroid. (iii) Central Normalized Moments: These normalized central moments are derived by normalizing the central moments, as depicted in Equation 3.15. Spatial moments of order $(p+q)$ are mathematically expressed as shown in Equation 3.13, and central moments are defined in relation to the centroid as depicted in Equation 3.14.

$$m_{pq} = \sum_x \sum_y x^p y^q I(x, y) \quad (3.13)$$

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \quad (3.14)$$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}} \quad (3.15)$$

3.8 Color spaces

Colored images can be represented in various color spaces. RGB, HSV, and CIELAB are three among the numerous options available. One might choose one over the others depending on the specific image processing and manipulation goals. Along the various steps of this project, the data was converted from one color space to the other to tackle encountered issues.

RGB

RGB is the most widely recognized color space for colored images and is constructed from a combination of Red (R), Green (G), and Blue (B) components. In an RGB image, each pixel's attributes are determined by the intensities of these three primary colors. Despite its intuitive nature, the RGB representation does have drawbacks, particularly in tasks like color segmentation. This is because variations in lighting conditions can lead to value fluctuations across all three channels, affecting the accuracy of color-based analysis.

HSV

The HSV color space, on the other hand, offers greater robustness against changes in lighting conditions compared to RGB. It employs three components: Hue, Saturation, and Value. Hue indicates the dominant color's wavelength, Saturation determines the color's intensity, and Value represents the color's brightness. This makes HSV particularly valuable for tasks involving color manipulation and adjustments.

CIELAB

The CIELAB, or LAB, color space is another 3 channels color representation. The first channel "L" represents light, the second "a" expresses the green-to-magenta color range and the final channel "b" expresses the blue-to-yellow color range. The light "L" channel of this representation is independent of color information while "a" and "b" encode color. This particularity makes the CIELAB color space a good candidate for various color analysis tasks.

3.9 Mathematical morphology

Mathematical morphology defines a set of operations primarily used on binary images where the foreground is denoted as 1s and the background as 0s. These operations are constructed through the expansion and contraction of the foreground; based on a structuring element K , which is often a circular disk. This simple yet effective method can be used for pre/post-processing of images in computer vision tasks. This subsection presents four mathematical morphology operators that were used in the context of this project.

Dilation

Dilation involves the expansion of the foreground in a binary image. It is particularly useful to fill holes within an object or enlarge features. Conceptually, we can envision the center point of the structuring element K moving within the perimeter of the foreground. Subsequently, any background pixels intersecting the structuring element along its probe are labeled as part of the dilated image's foreground.

Erosion

Erosion stands as the dual of dilation. The foreground of an eroded image maintains its original shape with noticeable reduction in size. For this reason, we can think of erosion as a dilation of the background. This operation proves beneficial in scenarios, such as separating interconnected objects.

Opening

Based on the previously introduced mathematical morphology operations, we can define the opening operation as the combination of erosion followed by a dilation. This operation results in removing parts of the image that are smaller than the structuring element K . Opening can be effectively used to filter out unwanted shapes from the initial image.

Closing

Similarly to the opening operation, closing results from applying of the dilation operation followed by an erosion. This operation is useful for filling small holes and gaps within objects, restoring their original shape and connectivity.

Chapter 4

Dataset

This work focuses on the task of individual identification for the case of the salamander population. For this purpose and to guarantee an effective training of the supervised deep learning algorithm employed and to ensure a good assessment of the identification algorithm implemented, it is crucial to build a good dataset. We define a good dataset for our task by the quality of the annotation and the size of the data. Evidently, the quality of the data plays a crucial role in determining the accuracy of the identification algorithm. In this chapter we will present the dataset that was curated for this purpose. It can be split in segmentation data and identification data. For both entities, we will present the sources of the raw images, explain the labeling process, and display the resulting composition of the dataset.

4.1 Segmentation dataset

The segmentation dataset is an essential part for the segmentation task. Indeed, as we will thoroughly explain in the next chapter, our identification process relies on the extraction of the salamander’s pattern through a deep learning algorithm which requires a significant number of samples to learn from. These samples consist in pairs composed of the salamander image and a manually crafted mask which indicates the contour of the animal’s body.

Images

- **Cut:** The starting point for this dataset is the dataset crafted by Francois Duchène for his thesis on the same topic [6]. Although the goal was to extend this dataset, after careful examination a number of images were excluded. These excluded images are: (i) the samples that did not include the presence of a salamander. Their corresponding masks being a black image, these

samples introduced a severe class imbalance leading the network towards a strong background bias despite the regularization (ii) the samples that portrayed a heavily occluded salamander.



Figure 4.1: Examples of excluded samples:

(i) top images do not include salamanders (ii) bottom images include parts of salamanders.

- **Mandermatcher:** A first category of images composing our custom dataset are the images provided by Jeroen Speybroeck the developer of the salamander identification software *Mandermatcher*¹. These images are of professional quality with an optimal visual display of the animal as the salamander is sat straight on a contrasting background with adequate lighting.
- **In the wild:** The second category of images composing our custom dataset is what we can describe as *In the wild* shots. Indeed these images photographed outdoors in Louvain-la-Neuve, Spy, Julienne, Andenne, Beloeil and Stambruges share common visual conditions linked with shooting animals in their natural habitat which constitutes a complex background. These *In the wild* conditions include (i) the presence of foliage of different color (ii) shift in luminosity (iii) possible occlusion of parts of the animal's body. Despite

¹<https://jeroenspeybroeck.shinyapps.io/mandermatcher/>

the common visual appearance of these images, their sources are different. Samples collected from Louvain-la-Neuve, Spy, Julienne and Andenne are part of Francois Duchène’s original dataset [6] and include the contribution of Adrien Goffin a amphibian enthusiast from the region of Andenne. Additional images were collected by Prof Bonaventure as well as Théo De Waele, Louis Uytterhaegen and myself.



Figure 4.2: Examples of our custom dataset image categories.

(i) top images are Mandermatcher category (ii) bottom images are In the wild photographs.

Masks

The original dataset was composed of 836 annotated images before the selective cuts. My contribution to the manual annotation of the salamander masks was carried out with the RectLabel² software and consists in a total of 400 labeled images.

Totals

Category	ManderMatcher	In the wild	All
Nb samples	330	835	1165

Table 4.1: Segmentation dataset composition.

²<https://rectlabel.com>

4.2 Identification dataset

The second part of our custom dataset consists in the data used for the assessment of the chosen identification algorithm. For the different images collected, the annotation for this part of the dataset consists in deriving an identity from the animal’s visual pattern. Consequently, different images of the same individual should be assigned the same identity code. Once again our dataset originated from different sources which we will present in this section.

Images

- **Segmentation images:** The images collected for our segmentation dataset were annotated with the ID label of the individual photographed.
- **Observations .be:** Additional pictures were downloaded from the Observations.be³ platform to cover a larger time interval in our salamander observations thus raising the odds of re-encountering the same individuals.
- **Boleil & Stambruges:** These images were shared by Renaud Lafourt and collected in the context of his study on the distribution of the salamander population in the region [13]. The images display similar characteristics to the ones described for our *In the wild* category of segmentation images.

Identities

The identification of salamander individuals carried out for the creation of this dataset was made in one of two ways : (i) In the context of an internship, Louis Uytterhaegen annotated the individual identities with use of the ManderMatcher software⁴. This process consists in assigning a code for each defined section in the body of the animal such as left/right cheek. Depending on the body part, this code can be binary to indicate the presence or absence of a spot or numeric to indicate the number of spots within the described region. The resulting identity is described as a sequence of numbers made of the concatenated codes. (ii) In the context of his thesis on the topic of salamanders [13], Renaud Lafourt annotated salamander identities manually based on careful visual inspection.

³<https://observations.be/species/458/>

⁴<https://jeroenspeybroeck.shinyapps.io/mandermatcher/>



Figure 4.3: Mandermatcher labeling interface.
Red rectangle represents the pattern characteristics panel.

Totals

We will now present an overview of the composition of the identification dataset. This includes the number of records, individuals and maximum individual frequency.

	Nb annotated images	Nb individuals	Max individual freq
Mandermatcher id	1072	637	19
Manual id	557	436	4

Table 4.2: Identification dataset composition.

We can conclude from the visualization in Figure 4.4 of the most frequent individual amongst the ManderMatcher labeled images, that the frequencies expressed within the identification dataset do not represent the re-encounters of the same individual across multiple sightings but rather the number of photographs taken of the same individual. This definition of the number of samples for a given individual allows for duplicates and burst mode during movement of the animal. Figure 4.5 indicates that the majority of individuals are encountered only once. We can conclude that the dataset contains a small subset of individuals having a large number of photographs taken and a majority of individuals only having one photograph.



Figure 4.4: Images of most frequent individual: ID30.

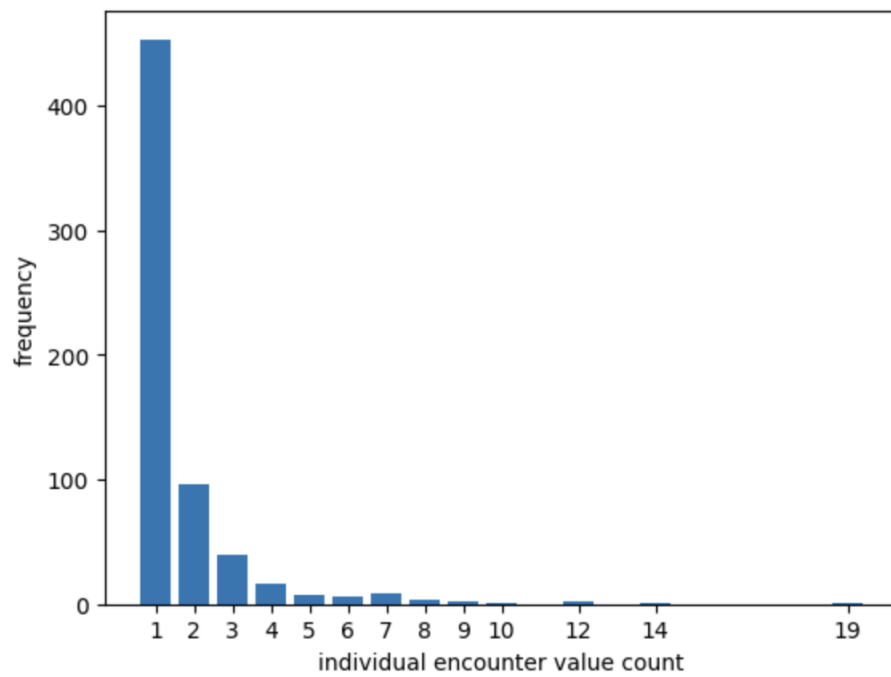


Figure 4.5: Frequency distribution of occurrences per individual.

4.3 Conclusion

In conclusion, this chapter presented the dataset used in this thesis. It is a custom dataset curated for the task of individual identification of salamanders. The dataset can be divided into two parts: segmentation data and identification data. We have explained the sources of the images, the labeling process, and the resulting composition of the dataset for both parts. The segmentation data was built as an extension of François Duchène’s work [6], while the identification data was annotated by amphibian enthusiasts based on visual observation or Mandermatcher software.

The segmentation data includes professional photographs that depict salamanders in optimal conditions in terms of lighting and animal positioning, as well as amateur photographs that capture the animal in its natural habitat, thereby presenting a more complex visual setting. The annotation of this data consisted in identifying the contour of the animal using a polygon to generate a binary mask which separates the pixels composing the salamander in the image from the background.

The identification data contains images collected from different sources. It includes the segmentation data in addition to images collected from Observations.be: the largest platform of nature observation in Belgium and the Netherlands, and images shared by Renaud Lafourt. The annotation process involved assigning a unique identifier to the salamander individual present in each image. The images provided by Renaud Lafourt were manually identified [13] meanwhile Louis Uytterhaegen identified individuals using ManderMatcher as part of his internship on the subject.

The resulting dataset is composed of 1165 samples for semantic segmentation of salamander bodies and 1629 images of 1073 identified salamander individuals.

Chapter 5

Segmentation

Image segmentation is a task which consists in a pixel wise classification of the elements present in a given image. In the case of a binary classification we can describe it as identifying the pixels composing our region of interest such that all other pixels are labeled as background. In the following chapter we will discuss the segmentation techniques that have been implemented and applied to our dataset images before tackling the individual identification problem at hands. Indeed, the identification of salamanders being directly linked to the uniqueness of the individual's pattern, it is wiser to extract those patterns from the presented images before feeding them to the chosen identification algorithm. The motivation behind such an approach is to avoid any possible noise introduced by the background such as yellow grass which can be mistaken for additional spots in the animal's print, thus completely changing the produced identity. Namely, we will introduce the U-net [31] model that was trained for the segmentation of the salamander body, and the K-means [8] model for the segmentation of the back spot pattern. For both of the aforementioned methods, we will define their input data, present their hyper-parameters and discuss their results and possible post-processing computations.

5.1 U-Net segmentation

As the name states, the first segmentation step relies on the U-net architecture. This architecture, which we will thoroughly explain in the next section, is a type of convolutional neural network [14] known to thrive in semantic segmentation tasks of different fields such as medical imaging [34]. This architecture was introduced in 2015 by O.Ronneberger &al with the innovative idea of skip-connections linking opposite convolution and deconvolution blocks of its U-shape form. We expect this method to provide us with an output image in which the salamander body is clearly preserved following its contour while the background is canceled out.

5.1.1 Input data

We have presented in the previous chapter the dataset that was gathered in the context of this thesis. Across the different steps of our work, these images will receive an adequate processing to suit the needs of the U-Net model and fix the possible anomalies noticed upon trials. In the case of the U-Net segmentation, the input image respects the following characteristics :

- **Non destructive resizing:** As the U-net architecture requires a fixed input size for all of the samples which are fed into it, resizing the dataset images is a mandatory step. We have defined the fixed size of our samples to (256,256). However, given the wide differences of the photograph views present in our custom dataset, it is noticeable that strict resizing of those images can lead to distortion. For example, in the case of very narrowly cropped images as shown in Figure 5.1, destructive resizing results in stretched out unrecognisable salamanders leading to poor mask predictions. The solution is to take into consideration the aspect ratio of our samples in order to linearly adjust images which can be resized without distortion and introduce zero padding to the ones which cannot.

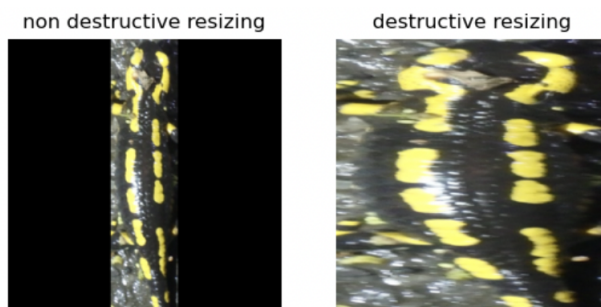


Figure 5.1: Example of a narrow image sample after resizing.

- **Color space:** The input images of the U-Net model are expressed in the RGB color space, the input shape of the U-Net model is thus (256,256,3).
- **Standardisation:** Upon trials, we noticed that the standardisation of the input images to upright position of the animal across all samples greatly improved the results of the segmentation in contrast to a training set with varying body positions of the salamander. Consequently, the training and validation samples follow an upright standardisation which means that the animal's head points the top and its tail points to the bottom. Naturally, as our dataset is composed of images collected by different people across different locations, the dataset still shows variations in the visual presentation

of the animal such as the curve its body can take when it is curled in or its position in the image when the photograph is taken in a diagonal manner.

- **Augmentation:** Dynamic image augmentation was introduced in the training of our U-net segmentation model through the use of Tensorflow’s ImageDataGenerator module¹. The applied augmentations include vertical flip, width and height shift, and zoom and shear range with a “nearest” fill mode. No drastic augmentation was applied as we did not want to cancel the upright standardisation (example: no sizeable rotation range filter), instead the chosen filters introduced regularisation with respect to (i) the shifts in the animal’s body which can be noticed in nature (for example pregnant salamanders show a wider body, and juveniles are shorter) and (ii) natural picture artefacts (for example: use of bright flashlight).

5.1.2 Architecture

One specificity of semantic segmentation convolutional networks, contrary to CNNs used for classification tasks for example, is that the input size must be identical to the output size to conserve the full resolution of the image. To respect this constraint, a type of segmentation networks consists in stacking convolution layers from start to finish (FCN: Fully Convolutional Network)[15]. U-Net models, contrary to FCNs, make use of downsampling convolutions and upsampling convolutions bridged by a bottleneck convolution in an auto-encoder fashion and additional skip-connections connecting analogous blocks from opposite ends (i.e. encoding and decoding sides) of the network.

¹https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

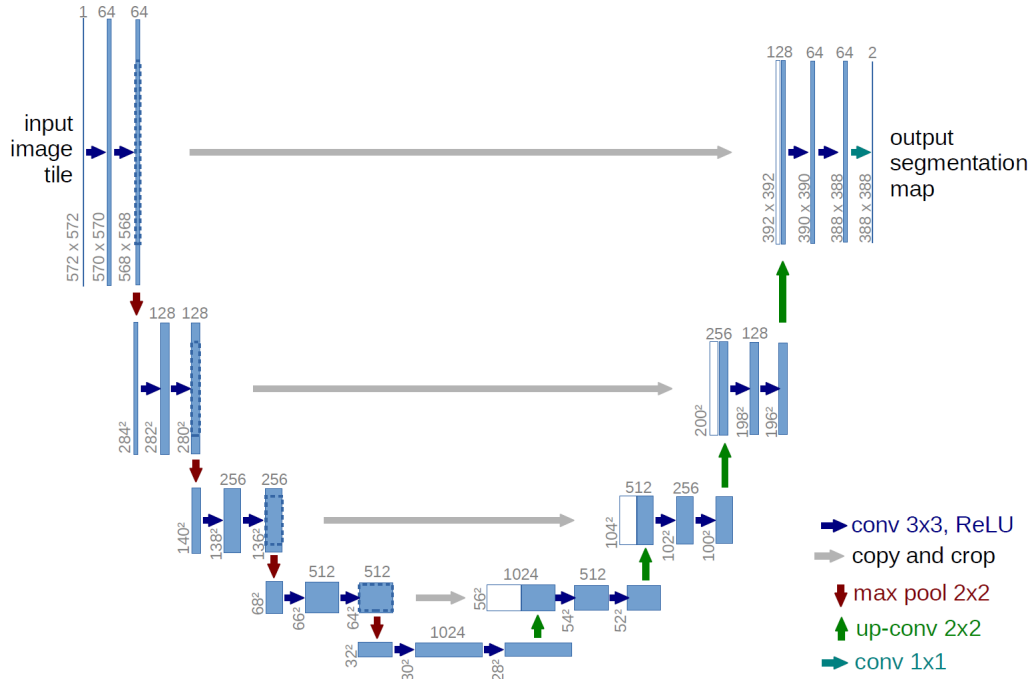


Figure 5.2: U-Net architecture. From [31].

- **Downsampling block:** the downsampling blocks of the U-Net sit on the left side of the architecture. It aims to build an embedding of the input image which gathers the important information. It is composed of two stacked convolution layers of (3x3) filters followed by a maxpooling layer of (2x2) filter. The input of the first downsampling block is the input image of the network. The produced feature map is the input of the following downsampling blocks until reaching what is called the bottleneck.
- **Upsampling block:** The upsampling blocks of the U-Net are located on the deeper side of the network. They aim to restore the full resolution of the image. Each upsampling block consists of a deconvolution layer, also called transposed convolution [22], followed by a concatenation layer and a double convolution block. The input of the transposed convolution layers is the feature map produced by the upsampling block preceding it. As for the higher resolution feature map it computes, it is concatenated with the corresponding skip-features which are defined as the feature map output by the facing downblock in the U-Net. The resulting feature map from the concatenation is passed through a double convolution of (3x3) filters.
- **Bottleneck:** This block consists in a double convolution of (3x3) filters sitting between the encoding and decoding blocks. Its input is the feature

map resulting from the last downsampling block and its output is passed to the first upsampling block, which directly precedes it.

Based on the previously explained elements, we can describe the architecture of the U-Net model implemented for the semantic segmentation of salamander bodies as follows.

	Nb filters	Activation	Kernel initializer
Downblock 1	64	ReLu	he_normal
Downblock 2	128	ReLu	he_normal
Downblock 3	256	ReLu	he_normal
Downblock 4	512	ReLu	he_normal
Bottleneck	1024	ReLu	he_normal
Upblock 1	512	ReLu	he_normal
Upblock 2	256	ReLu	he_normal
Upblock 3	128	ReLu	he_normal
Upblock 4	64	ReLu	he_normal
Output	1	Sigmoid	-

Table 5.1: U-Net architecture details.

5.1.3 Hyper-parameters

The trainable parameters represent the set of parameters which are estimated through the learning of the model, and are required during inference for the decision making. Our U-Net model has 34,513,345 trainable parameters. Hyper-parameters, on the other hand, are not estimated by the model but rather manually set beforehand and influence the efficiency of the training. For example, choosing a large learning rate can lead to divergence. Consequently, an important part of building a deep learning model resides in the choice of its hyper-parameters. This step of fine-tuning, which consist in finding the right combination of hyper-parameters in order to optimize the obtained results for our task is crucial.

Careful fine-tuning of the U-net model led to the following choices:

Hyper-parameter	Value
Loss function	Dice Loss
Smoothing factor	10
Optimizer	Adam
Learning rate	1e-5
Metric	Dice coefficient
Epoch	60
Batch size	32

Table 5.2: Hyper-parameters used for training.

Among the hyper-parameters presented in Table 5.2, we have the Dice coefficient introduced in chapter 3 and the loss derived from it which is called Dice Loss. The Dice Loss is defined, as shown in equation 5.2, based on the Dice coefficient expressed in equation 5.1. This definition of the Dice Loss is a value which is minimized as the model improves its predictions. For our semantic segmentation task we can compute the Dice Loss with the introduction of a smoothing factor (denoted as epsilon in equation 5.3). In some version of this loss the smoothing is set to 1 in order to avoid undefined cases. However, a smoothing of 1 for our U-Net model outputs blank masks of black background. This was a result of overfitting on the background class because of the class imbalance known in this task. Increasing the smoothing factor of the Dice Loss helped alleviate the issue by reducing the impact of false negatives during training.

$$Dice(y_{true}, y_{pred}) = \frac{2 * |y_{true} \cap y_{pred}|}{|y_{true}| + |y_{pred}|} \quad (5.1)$$

$$DiceLoss(y_{true}, y_{pred}) = 1 - Dice(y_{true}, y_{pred}) \quad (5.2)$$

$$DiceLoss(y_{true}, y_{pred}) = 1 - \frac{2 \sum_{i=1}^N y_{true,i} y_{pred,i} + \epsilon}{\sum_{i=1}^N y_{true,i}^2 + \sum_{i=1}^N y_{pred,i}^2 + \epsilon} \quad (5.3)$$

5.1.4 Results

After training the U-Net model we introduced, with the chosen hyper-parameters on a training set of 895 samples and validation set of 135 samples from our custom dataset presented in chapter 4, excluding the dynamical augmentation step, over 60 epochs we reached the results presented below:

- For the training data the dice_coefficient reached 0.8 with a dice loss of 0.19
- For the test data the dice_coefficient reached 0.81 with a dice loss of 0.189

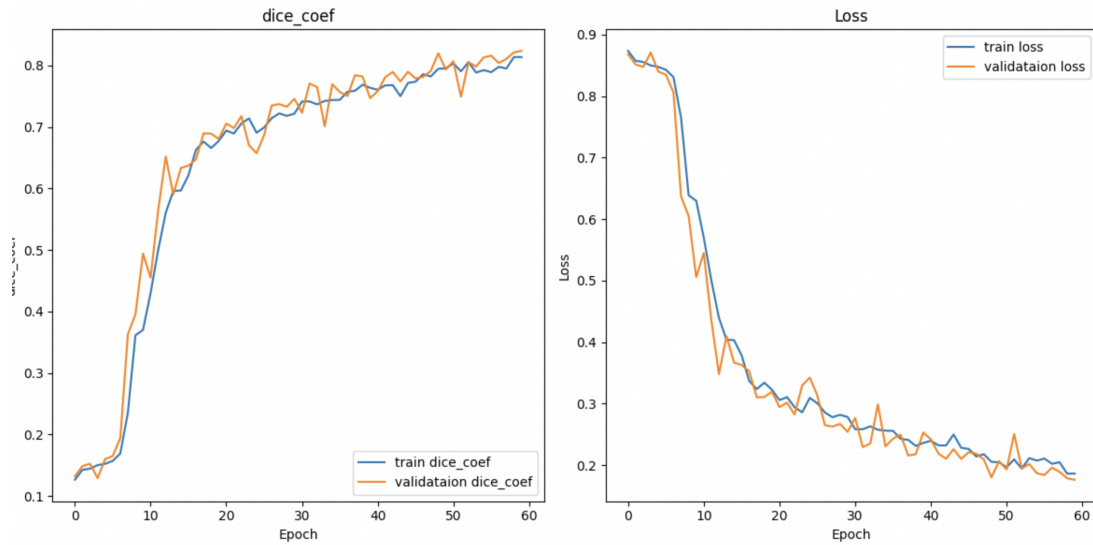


Figure 5.3: Training and validation dice coefficient and dice loss.

Figure 5.3 shows the learning curve of the chosen model during its training. Such plot is a good tool to judge the presence or absence of over-fitting as we can notice the behavior of the model across the learning epochs on the training data and validation data. The plots of Figure 5.3 show a noticeable similarity between the history callback curve for the training samples and validation samples. We can conclude that the trained model does not display any signs of overfitting on its training data.

Evaluation of the trained U-Net model on the test set composed of 135 samples scored 0.8 for dice coefficient and 0.19 for dice loss.

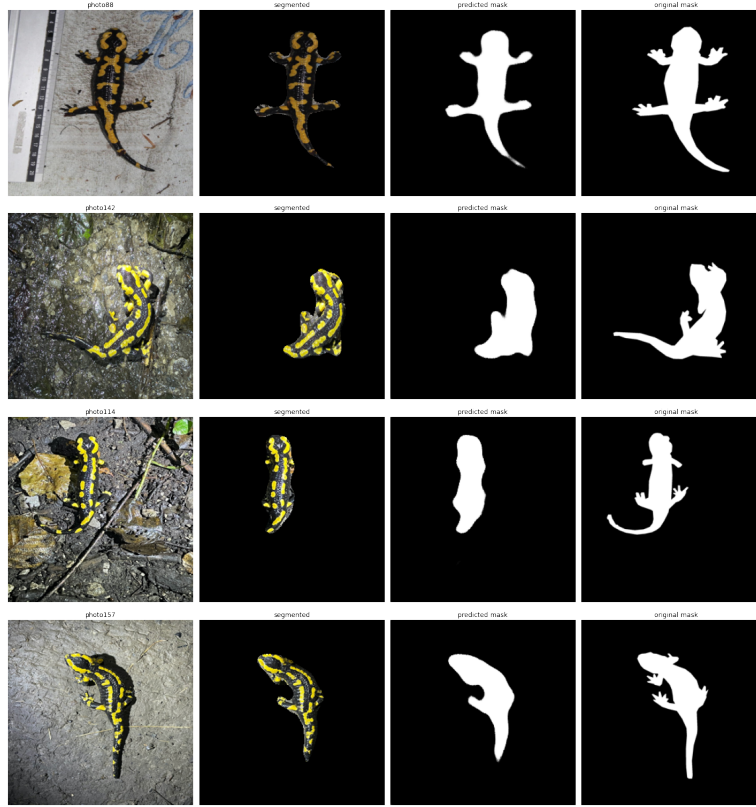


Figure 5.4: Visualization of U-Net segmentation results. (a) original image sample, (b) segmented image, (c) predicted mask and (d) original mask.

Figure 5.4 presents some of the images segmented by our trained model. For each of these examples we can see on the far left, the original image and on the far right the true mask which was manually created. In the middle we can find the prediction mask and the extracted salamander body. The top example presents the closest similarity between prediction mask and true mask as we can notice that the tail is mostly preserved and the four legs are clearly extracted. This good performance is directly linked to the quality of the photograph as it is a sample from the Mandermatcher category of images presented in chapter 4. The remaining examples are samples from the InTheWild category as we can notice from the background. For this category we notice that the model successfully extracts the body of the animal whether it is straight or curved, but the tail can be cut short and the toes ignored.



Figure 5.5: Two salamanders samples results. (a) original image sample, (b) segmented image, (c) predicted mask and (d) original mask.

Despite the fact that most of our dataset images contain only one salamander, as it is the preferred set-up of photographs in the context of salamander observation, the few images containing a couple of salamanders found in the data present interesting results. Naturally as our model was not trained to perform instance segmentation, the former does not differentiate between the different salamanders present in the picture. An interesting point to notice is that in some cases such as the top example in Figure 5.5, the model has a better prediction mask for the salamander that is sitting in a vertical position than the one in a horizontal position. This is a direct result of the standardisation step as the model was trained on upright salamanders. The third example further corroborates this theory as it shows the strongest similarity between predicted and true masks and its original images portrays two facing salamanders in vertical position.

5.1.5 Post-processing

Upon visual inspection of the U-Net predictions across the validation set, a number of anomalies were spotted. Indeed, an apparent weakness in the model's performance is the presence of holes in the predicted mask. These holes, when present, are located on the black center of the salamander's head. We suspect that this phenomenon is linked to the flash reflection on the humid animal body. The other anomaly that was unveiled is linked to the salamander's tail

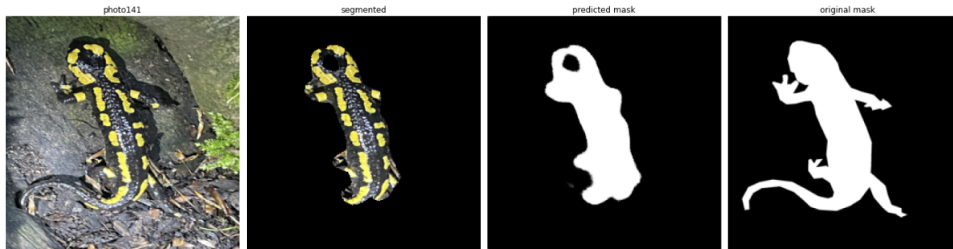


Figure 5.6: Example of flawed prediction with head hole.

Since the problem is defined on a binary image, we can use mathematical morphology to solve it. Precisely we applied the closing operation for 1 iteration with a 20x20 square structuring element on our 256x256 binary predicted masks. This post-processing step is a good solution for attaching tails and filling in flash gaps as show in the Figure 5.7. However, it comes with a disadvantage. As we can notice in Figure 5.8 applying this closing operation to all prediction masks can introduce false-positives into correct masks by connecting the legs of the animal.

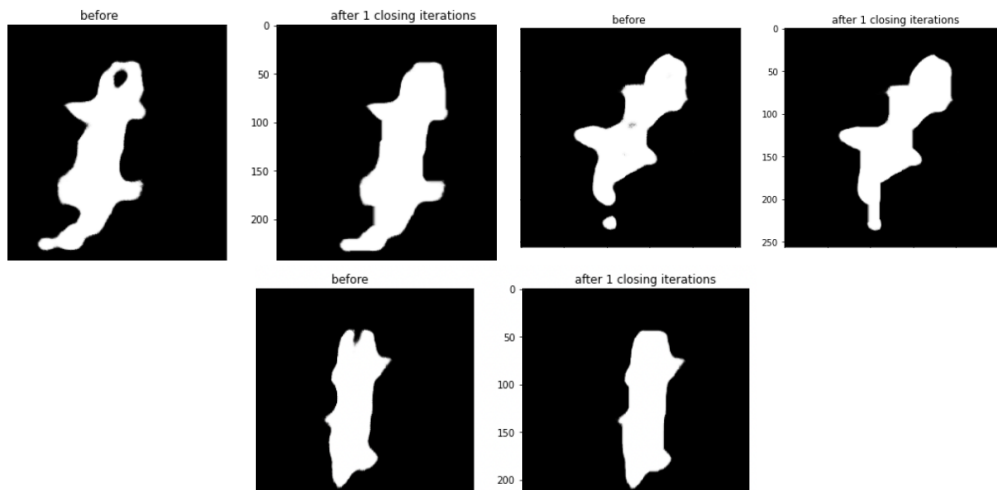


Figure 5.7: Examples of corrected masks.

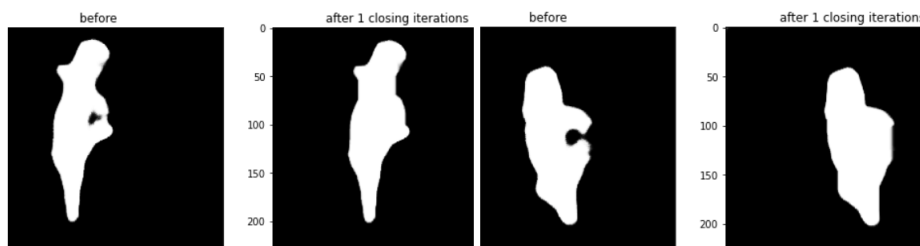


Figure 5.8: Examples of connected legs closing.

In our specific context, we prefer a false positive to a false negative. Indeed the motivation behind the use of the semantic segmentation was to extract the information related to the animal’s body from the image while removing all background information. In that perspective, it is more important to ensure that the body of the animal is preserved in the segmented image even if that means that some background noise is introduced in that information. Additionally, the next step following the U-Net segmentation will focus on extracting the yellow spot from the salamander’s back and removing the rest of its body as it constitutes the background in the new segmentation problem. Following this context we can decide that admitting some false positives introduced by the closing operation is acceptable since these false positives will be ruled out during the second round of segmentation.

5.2 K-means clustering

K-means algorithm [8] is used in the domain of computer vision in order to solve color segmentation problems. This is precisely useful for the extraction of the salamander’s back pattern as we can visually notice the strong contrast of colors between the yellow spots and the dark remaining skin.

5.2.1 Input data

The input images for K-means clustering are generated from the output of the U-Net segmentation. These images consist in the salamander body which was extracted from the original sample. Experimentation showed that running K-means on such images introduced some problematic noise in the produced mask because of the false positives present in the semantic segmentation mask. Since the problem was persistent in the RGB space, another color space was investigated for this task:

Color space: The initial intuition behind the use of K-means color segmentation for the extraction of the spot pattern was to introduce the RGB segmented image of the animal's contour and rely on the distance between yellow points and black points in the RGB space as shown in Figure 5.9. Unfortunately, when the semantic segmentation includes background pixels into the predicted mask, these pixels are generally lighter than the black body and labeled in the spot cluster of K-means which groups the yellow pixels. A solution for this is to convert our segmented images into another color space which is more suitable; CIELAB color space is a good candidate [26].

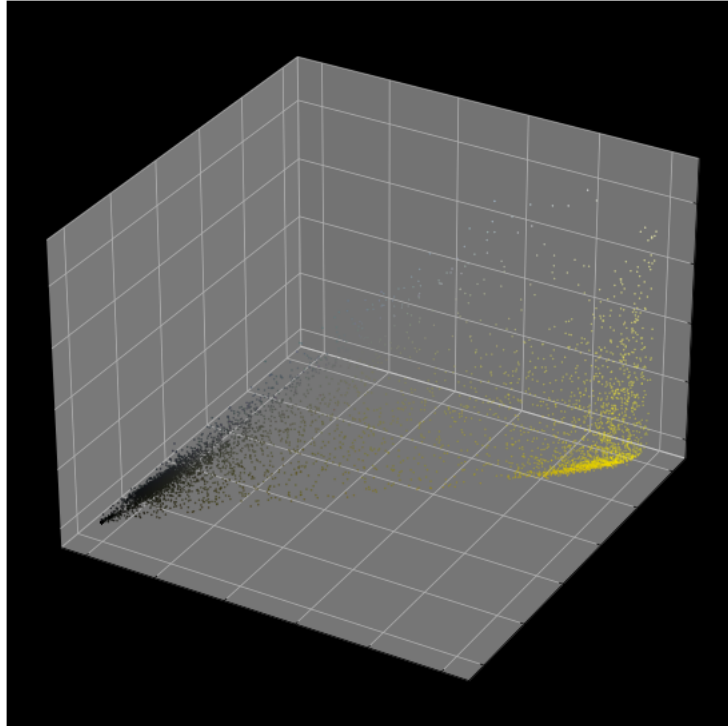


Figure 5.9: Visualization of the 3D RGB space for a salamander body.

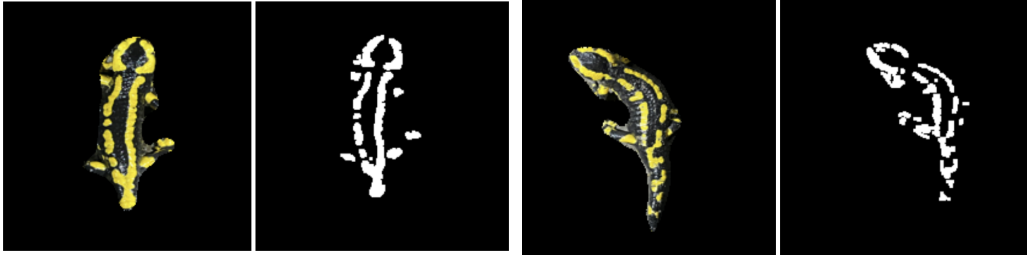


Figure 5.10: K-means color segmentation in RGB space.

In Figure 5.10 above we can see two examples of K-means color segmentation results on our U-Net segmented salamanders. The left side shows a correct segmentation as the yellow pattern is expressed as the foreground of the binary mask and the rest of the pixels are background pixels. The right side shows the expected flaw of this method which is the persistence of U-Net false-positives into K-means false-positives. Indeed we can notice in the far right mask that an additional straight spot is concatenated to the pattern of the animal. This straight spot is the representation of the grey edge included in the segmented contour which is in reality the ground on which the salamander was standing.

Hyper-parameters

The color segmentation of the CIELAB images was implemented using Sklearn’s [27] K-means function available in its cluster module. The hyper-parameters used are detailed in the table below:

	Nb clusters	n_init	max_iter
value	3	10	10

Table 5.3: K-means color segmentation hyper-parameters.

- **nb_clusters:** The number of clusters that the K-means algorithm will use to group the colors in the image.
- **n_init:** The number of times the K-means algorithm will be run with different centroid seeds.
- **max_iter:** The maximum number of iterations for each run of the K-means algorithm. If the algorithm doesn’t converge before the maximum number of iterations is reached, it will stop and return the results obtained so far.

5.2.2 Results

In this subsection, we present the results of applying the K-means color segmentation algorithm to the salamander body in order to extract its distinctive back spots.

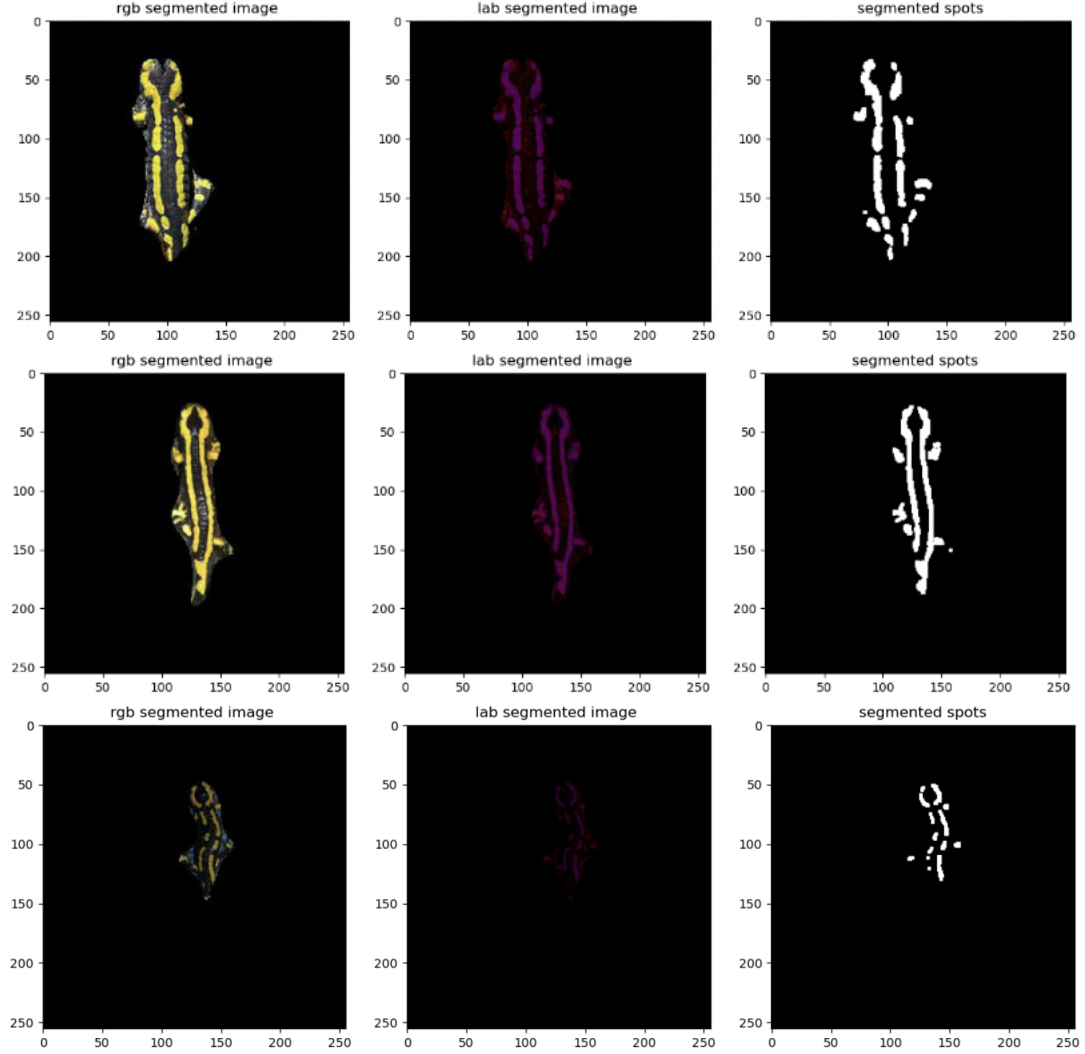


Figure 5.11: Visualization of K-means segmentation results. (a) U-Net segmented salamander in RGB space, (b) U-Net segmented salamander in CIELAB space, (c) K-means segmented spots on CIELAB image.

In the presented results of 5.11 we can see for each example the U-Net segmented body in RGB space on the left followed by its CIELAB converted image and on

the right the segmented spots using K-means algorithm on the CIELAB image which visually encodes the unique identity of the portrayed individual.

Silhouette evaluation

An evaluation of the performance of the K-means clustering on our data is possible with the Silhouette Score introduced in chapter 3. This metric allows us to judge the quality of the clustering by computing the intra-cluster distance and inter-cluster distance. Precisely, the Silhouette evaluation was conducted by computing the Silhouette Score for each image sample using the `sklearn.metrics` module² with an Euclidean distance. The Silhouette Score is a value between -1 and 1, where a score closer to 1 indicates a better clustering result. The final spot segmentation performance can be expressed as the average Silhouette Score across the dataset. Following this definition, the k-means clustering algorithm reached a Silhouette Score of 0.935 on the segmentation dataset. This score indicates the effectiveness of the algorithm in identifying and separating the different clusters in the dataset, with a score closer to 1 indicating better performance.

5.2.3 Post-processing

In some cases we can notice that some noise is present in the segmented spots. This noise comes in the form of small dots appearing in the middle of black regions of the mask. They come as a result of the flashback present on the black back of the animal. Indeed the CIELAB space taking into consideration the lightness composing the pixel assigns some bright flash points to the foreground cluster of the K-means segmentation as we can see in Figure 5.12. This anomaly being minor in the produced masks, we can fix it with an additional post-processing step : 1 iteration of morphological opening with a 3x3 rectangle structuring element. Figure 5.13 shows the impact of this operation.

²<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics>

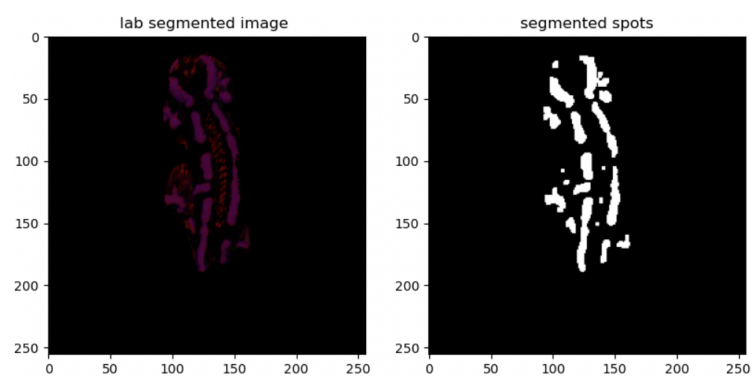


Figure 5.12: Example of small noise in the segmented spots.

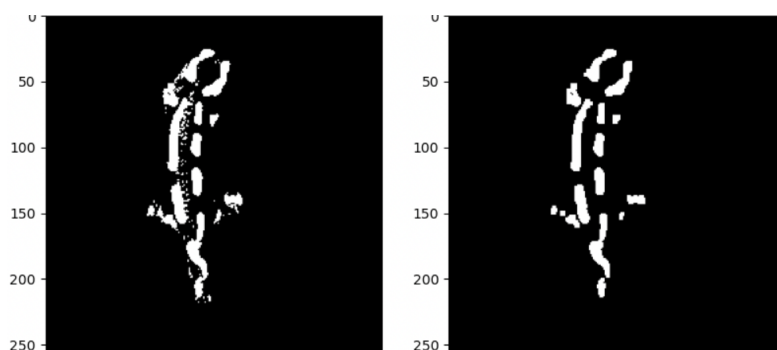


Figure 5.13: K-means segmentation before and after opening operation.

Chapter 6

Identification

Animal individual identification is a prominent research field in biology, where computer vision has emerged as a valuable tool for image-based identifications. The primary goal of these projects is to utilize distinct visual characteristics of animals to determine their unique identities. These characteristics range from giraffe spot patterns [16] and tiger stripes [10] to shark dorsal fin contours [11] and lemur faces [5]. In the case of salamanders, visual identification is possible based on their dorsal patterns.

Despite the wide range of potential visual features to utilize in this task, it is important to note that there is no universal approach for animal identification. The effectiveness of a particular method depends on the specific characteristics of the target animal. Moreover, the challenges associated with identifying a species are heavily influenced by its environment, which affects the conditions for capturing images. For example, wild animals like cheetahs or elephants are observed in vast national parks and cannot be approached closely. Consequently, most recognition systems are developed to target one species only, and a variety of recognition studies introduce diverse workflows.

Two main categories of approaches can be distinguished. One approach relies entirely on deep learning techniques and aims to construct similarity networks. These networks take two images as input and calculate the distance between activation values in the latent space to quantify the similarity between the individuals in the images. The second approach utilizes hand-crafted descriptors to capture specific visual features, such as color patterns, texture details, and shape characteristics and is followed by classification or clustering algorithms. Additionally, hybrid systems which combine deep learning algorithms with hand-crafted features have emerged. This can be done in one of two ways : (i) input the processed hand-crafted features into a classification network or (ii) leverage deep learning algorithms as a first

step to facilitate the feature description. This usually involves an object detection network which produces a bounding box around the targeted area (area can be the body, face or pelage of the animal) or a segmentation network for a finer ROI definition.

In this chapter, the focus will be on presenting the approaches investigated in our research to identify individual salamanders and their limitations. Building upon the previous chapter, where we discussed the application of image segmentation for extracting the distinct spot patterns of the animals, we can now establish that our methodology falls within the hybrid category of re-identification methods. To advance our work, the subsequent step is the exploration of some local feature descriptors that emphasize the subtle differences between individual salamander patterns, enabling accurate identification. Furthermore, the challenges that emerged during this step will be addressed and the strategies employed to overcome them will be discussed.

6.1 Local Binary Patterns

Local Binary Patterns [24] are a known texture descriptor in computer vision which proved successful in a number of identification cases such as face recognition [2] and ear recognition [9]. As for animal re-identification, Local Binary Patterns were used in the LemurID system [5] which identifies red-bellied lemurs individuals based on cropped face images. This simple yet efficient hand-crafted feature originally aims to capture the texture variation in an image region (as explained in chapter 3). Despite the fact that LBPs excel in cases where texture analysis is a crucial key to the identification task on grey-scale or colored images, it can still be useful to describe patterns of binary images by capturing the spatial relationships of the patterns within the binary image regions. Nevertheless, in the case of binary images, the patterns extracted by LBP may have fewer variations due to the limited range of intensity values. Moreover, variants of the traditional LBP descriptor such as Circular LBP (described in section 6.1.1) which is designed for circular regions or blob objects looks like a good fit for the extracted salamander spots.

Figure 6.1 presents the steps involved in the feature extraction process using the Local Binary Patterns (LBPs) descriptor in the context of our work. The implementation details are discussed after providing an overview of the workflow.

The first step is to transform the segmented pattern into an encoded image based on the chosen LBP variant and the specified parameters. This encoding process captures the local patterns within the image.

Next, the encoded image is divided into neighborhood blocks. For each block, a local occurrence histogram is computed, which represents the distribution of local patterns within that block.

Finally, the set of local LBP histograms is concatenated to form the global feature vector, which represents the complete pattern information of the image. This global feature vector can be used for subsequent analysis or recognition tasks.

To measure the similarity between the produced histograms, various distance metrics can be employed, such as the Kullback-Leibler divergence or the chi-square distance. These metrics provide a quantitative measure of the dissimilarity between two histograms, enabling the comparison and classification of different images based on their LBP features.

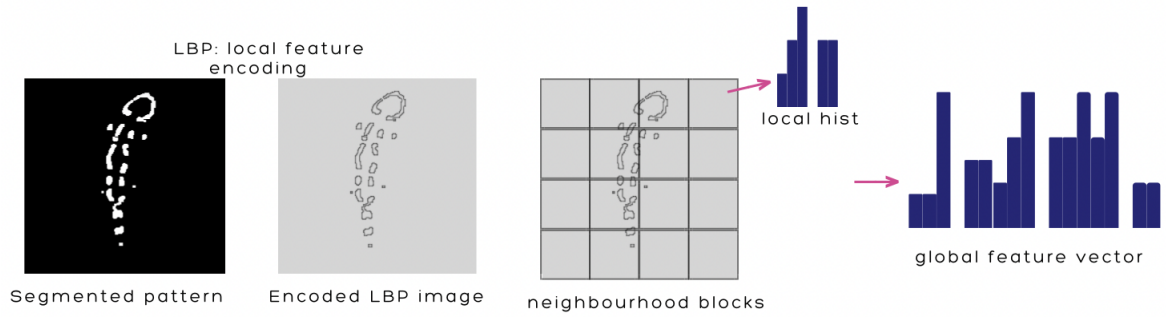


Figure 6.1: Feature extraction workflow using Local Binary Patterns.

6.1.1 Circular Local Binary Pattern

Circular Local Binary Pattern (CLBP), is an extension of the traditional LBP descriptor which suits circular patterns and blob objects. In particular, instead of following the strict neighborhood definition made in the basic LBP operator which consists in thresholding the pixels in a fixed 3x3 image patch, CLBP neighborhood is defined as the circle around the center pixel with a chosen radius r . From this circle, we sample a number of points p to threshold with respect to the center pixel c . For a given image region, we can define different CLBP neighborhood according to the chosen radius r and number of points p as illustrated in Figure 6.2. The CLBP feature extracted from the region can be computer as shown in equation 6.1 and equation 6.2 where g_i represents the value of pixel i (equal to 0 or 1 in our binary spot patterns).

$$CLBP(p, r) = \sum_{i=0}^{p-1} s(g_i - g_c) \quad (6.1)$$

$$s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (6.2)$$

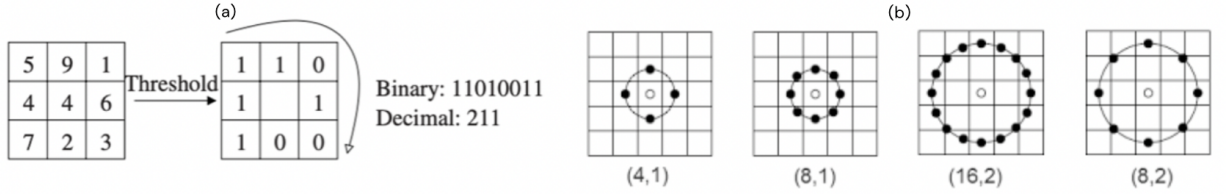


Figure 6.2: (a) The basic LBP operator From [2]. (b) The CLBP neighborhood.

In Figure 6.2 we can see (a) The basic LBP operator: The 3x3 image patch is thresholded according to the center pixel. The encoded patch resulting from the threshold is used to compute the binary pattern using a clockwise reading. The binary code is transformed into a decimal one which is the assigned value for the region. (b) The CLBP neighborhood: The 5x5 image patch can be encoded using different neighborhoods. The neighborhood is determined by the couple (number of points, radius). A larger radius captures a more global context whereas a smaller one captures finer details. Similarly, a higher number of samples from the neighborhood circle increases the level of detail as well as the computational complexity.

It is worth noting that CLBP can be utilized as a multi-scale descriptor. This can be done by computing the CLBP for a given center pixel using different radii values then concatenating the resulting local histograms to form one multi-scale histogram for that region. LemurFaceID [5], the red-bellied lemurs re-identification system cited in section 6.1, generated 3 histograms for each image patch with radii 2, 4 and 8 and concatenated them after normalization.

6.1.2 Uniform Local Binary Pattern

The uniform LBP method [24] filters out the non-uniform patterns and assigns a unique label to each uniform pattern. By considering only the uniform patterns, the uniform LBP method focuses on capturing the local visual variations that are likely to carry more discriminative information. This LBP method can be applied to any variant, including CLBP. (Uniform CLBP will be denoted as U-CLBP)

6.1.3 Histogram

For each neighborhood block defined on the LBP encoded image we derive a histogram. Each bin in the histogram represents an LBP pattern and its value expresses the occurrence of that pattern within the image.

Normalization

To ensure fair comparison and eliminate the influence of image size or intensity variations, we apply normalization to the histogram. This normalization step adjusts the values within the histogram to a common scale, typically by dividing each bin value by the total number of pixels or by the sum of all bin values.

6.1.4 Distance measures

As explained in section 6.1, the intended workflow measures the similarity between individuals by computing similarity measures on the extracted features in their histogram form. Two measures were investigated, those are the Kullback-Leibler divergence and the Chi-square distance.

Kullback-Leibler divergence

Kullback-Leibler divergence quantifies how much a given distribution p differs from another distribution q based on the formula expressed in Equation 6.3. It can be used for histogram comparison between a test image and a reference image [37]. KL divergence assigns a minimum score for histograms that are highly similar.

$$D_{KL}(P\|Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right) \quad (6.3)$$

Chi-square distance

Chi-square distance is a statistical measure that compares two histograms such that a lower score expresses similarity between the histograms. The Chi-square score between two vectors H_1 and H_2 is defined in equation 6.4

$$D_{\chi^2}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \quad (6.4)$$

6.1.5 Results

The results subsection of the Local Binary Patterns for Identification method presents the outcomes of various trials conducted to address the salamander re-identification problem using LBP features. It also highlights the limitations encountered during this approach. The selected LBP variants, along with their specific parameters, were applied to our salamander spot images to generate distinctive histograms. These histograms were then compared using the two chosen metrics. Furthermore, we provide insights into the impact of different parameters on the encoded image and histogram. Finally, the limitations of the method will be discussed, as it failed to meet our expectations in certain aspects.

Impact of variants and parameters

We can observe in Figure 6.3 and Figure 6.4 the impact of the circle radius and number of samples parameters on the encoded image when using the basic CLBP operator. Similarly, Figure 6.5 and Figure 6.6 present the same analysis using the U-CLBP operator.

In Figure 6.5, we can observe that a radius of 1 captures the contour of the pattern, while larger radius values result in a more abstract representation. This is because a smaller radius captures finer details. Furthermore, Figure 6.5 and Figure 6.6, which show U-CLBP results, exhibit more distinct patterns compared to Figure 6.3 and Figure 6.4, which present CLBP results.

Significantly, the disparity between the CLBP feature histograms and U-CLBP feature histograms can be seen in Figure 6.7 and Figure 6.8. In Figure 6.7, we observe two CLBP encoded images with their respective feature histograms on the right. Despite the evident visual difference between the two patterns, the histograms produced by the CLBP method are identical. Consequently, the classic CLBP fails to differentiate salamander patterns. In Figure 6.8, two U-CLBP encoded images with their respective feature histograms are shown. This variant of the Local Binary Pattern descriptor generates distinct histograms for the pair of unique patterns.

Based on these simple visualizations, we can conclude that the U-CLBP might be a more suitable approach for salamander pattern discrimination. It captures finer details and provides more distinctive feature histograms, enabling better differentiation between individual salamanders.

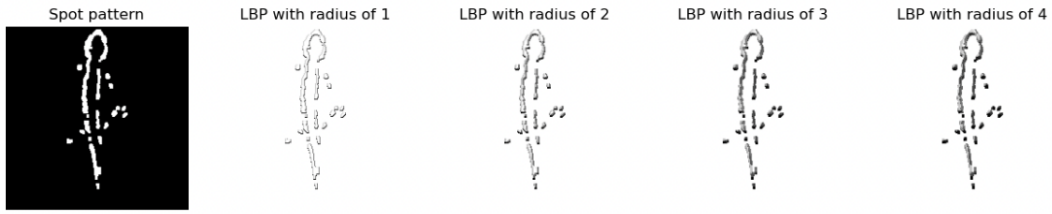


Figure 6.3: Impact of different radii on CLBP encoded image (nb_points= 8)

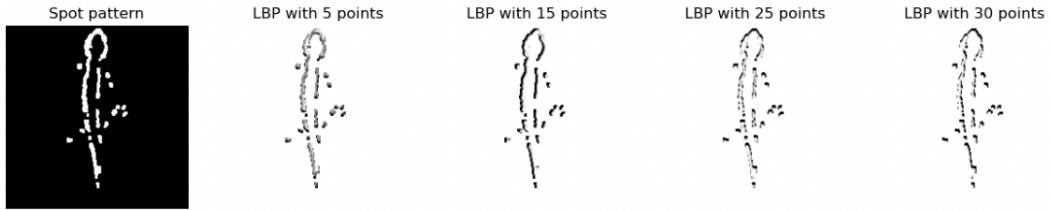


Figure 6.4: Impact of different nb_points on CLBP encoded image (radius= 3)

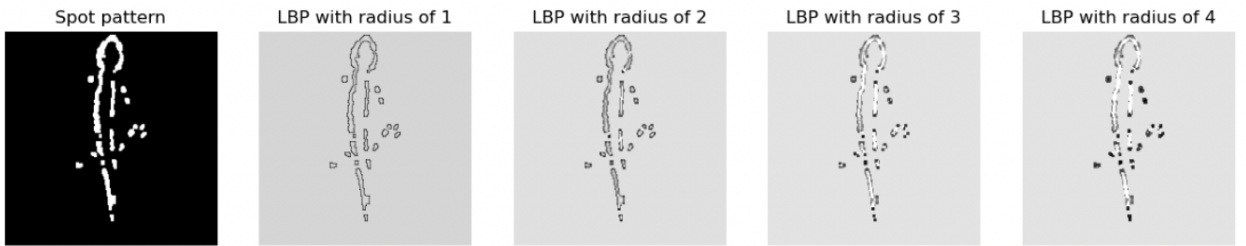


Figure 6.5: Impact of different radii on U-CLBP encoded image (nb_points= 8)



Figure 6.6: Impact of different nb_points on U-CLBP encoded image (radius= 3)

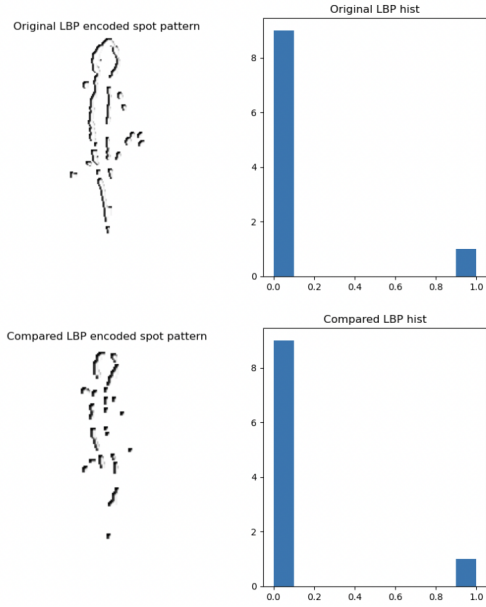


Figure 6.7: Example of feature histograms associated to CLBP encoded images. ($r=2$; $nb_points=16$).

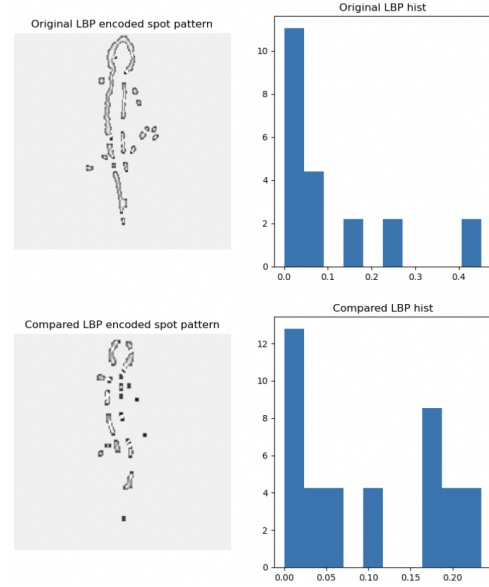


Figure 6.8: Example of feature histograms associated to U-CLBP encoded images. ($r=2$; $nb_points=16$).

Limitations

Using U-CLBP with parameters $r=2$ and $nb_points=16$, which provided distinctive histograms compared to other parameter settings, we conducted an experiment to assess the method's suitability.

In this experiment, we selected an individual salamander and created three modified copies by obscuring different parts of the spot patterns incrementally. The goal was to evaluate if the similarity scores obtained from our comparison workflow could accurately reflect the level of dissimilarity between the copies. However, the results of this trial, depicted in Figure 6.9, revealed a significant limitation in the KL divergence scores. Surprisingly, the most similar copy was assigned the highest score, while the most dissimilar copy received the lowest score. This discrepancy indicated that the identification based on KL divergence failed to capture the incremental differences between the modified copies.

On the other hand, the Chi score demonstrated more promising results. The order of the distance scores corresponded to the level of modification, indicating a better alignment with the expected outcome. However, despite respecting the

order, the dissimilarity score assigned to a completely different pattern was minimal compared to our generated copies, which were visually much closer to the original template. This inconsistency in results raises concerns about the reliability of the Chi score in accurately capturing the true dissimilarity between patterns. Indeed, throughout our identification dataset, the Chi score values ranged. However, it is notable that all calculated scores were below 0.3, indicating that most patterns were classified as similar. This suggests that the histograms generated by the method were not distinctive enough to capture the required level of detail that defines the identity we aim to derive. Consequently, the method fails to accurately distinguish between patterns, as it fails to capture the necessary level of dissimilarity.

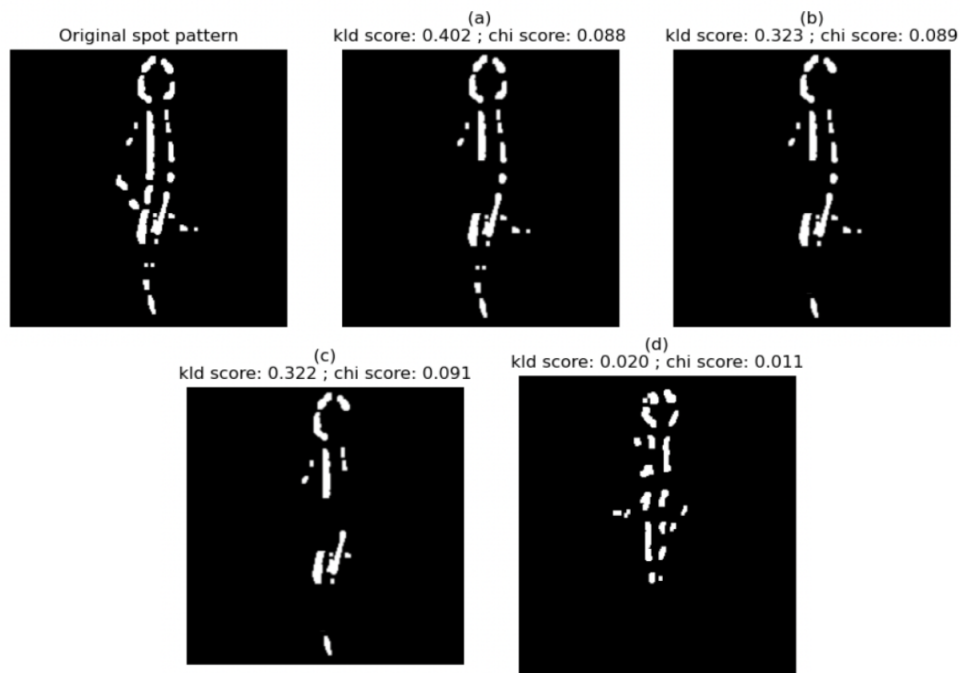


Figure 6.9: Results of the comparison experiment.

6.2 Body Alignment

In order to ensure accurate identification of salamanders based on their patterns, body alignment plays a crucial role. Alignment is a common component of animal identification workflows prior to image matching. This process is usually conducted using pose-estimation models or manually annotated landmarks to determine the affine transformation parameters.

We implemented a simple body alignment process to address this requirement and enhance the consistency of our dataset. Figure 6.10 displays an example of our desired outcome. The template salamander displays a number of characteristics which we believe will facilitate the identification method: (i) an upright orientation, (ii) a centered position, (iii) zoomed-in focus on the body length, and (iv) a straight salamander body.

By aligning the salamander images according to these characteristics, we aim to establish a standardized representation that simplifies the pattern recognition task and enables more accurate and efficient identification of salamanders. This alignment process improves the comparability and consistency of the dataset, laying a solid foundation for subsequent analysis and classification algorithms



Figure 6.10: Example of goal salamander normalization.

6.2.1 Affine Image Transformations

To achieve body alignment, we employed affine image transformations. These transformations enable us to apply various geometric operations such as translation, rotation, scaling, and shearing to the salamander images. By experimenting with different transformation parameters, we aimed to align all salamanders in the dataset, making their patterns more comparable and improving the effectiveness of our identification approach.

Contour features

When working with binary images, the OpenCV library offers various functions to extract useful information from contours. These features include area, perimeter, bounding rectangle, and more.

- **Bounding rectangle:** The `cv.boundingRect` function allows us to calculate the bounding rectangle of a foreground object within a contour. This rectangle provides the smallest possible rectangular region that encloses the object.
- **Fitted line:** The OpenCV library provides the function `cv.fitLine`, which can be used to compute a fitted line that approximates the shape of a contour. This line represents the best linear fit to the contour points and can be useful for estimating the orientation or direction of the object represented by the contour.

Transformations

The previously presented contour features were leveraged to apply the required affine transformations to approach our chosen template. Precisely, rotation, scaling and translation were applied to the salamander mask and then replicated on the corresponding image.

- **Rotation:** The angle of rotation calculated aims to align the mask's fitted line with the vertical axis. This rotation puts the animal in the upright position, eliminating any skew or tilt in the image.
- **Scaling:** To ensure a consistent size and proportionality between all aligned images, they were cropped according to the bounding rectangle. This ensures that the salamanders are uniformly scaled.

- **Translation:** The translation vectors calculated aim to center the salamander body in the image. It places the rotated fitted line in the middle of the horizontal axis, thus repositioning the salamander to the center of the image.

Figure 6.11 exhibits the results of the affine transformation on three samples of salamander masks. For each of the three samples, the five columns show: (i) the original mask with the plotted bounding rectangle, (ii) the original mask with the fitted line, (iii) the rotated mask, (iv) the scaled mask after rotation and (v) the translated mask after scaling.

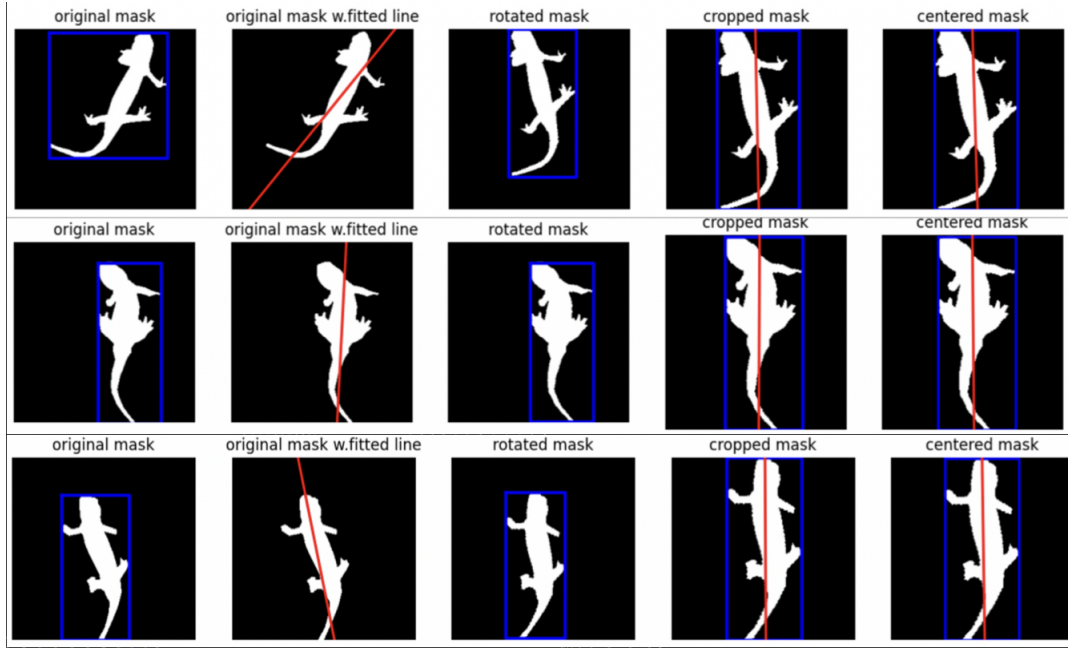


Figure 6.11: Affine transformation on salamander mask.

Figure 6.12 exhibits the results of the affine transformation on three interesting samples of segmented salamanders. For each of the three samples, the five columns show: (i) the original image, (ii) the original image with the fitted line, (iii) the rotated image, (iv) the scaled image after rotation and (v) the translated image after scaling. We can observe that the aligned salamander of the bottom example respects the characteristics of our alignment template. However, the remaining examples express the limitations of our current alignment method.

Indeed, the upright and centered salamander produced in the top example is curved which does not respect the straight salamander format we expect from our alignment. The second limitation, as we can observe it in the second row of Figure 6.12, lies in the fitted line. In some samples it fails to retrace the axis of the animal.

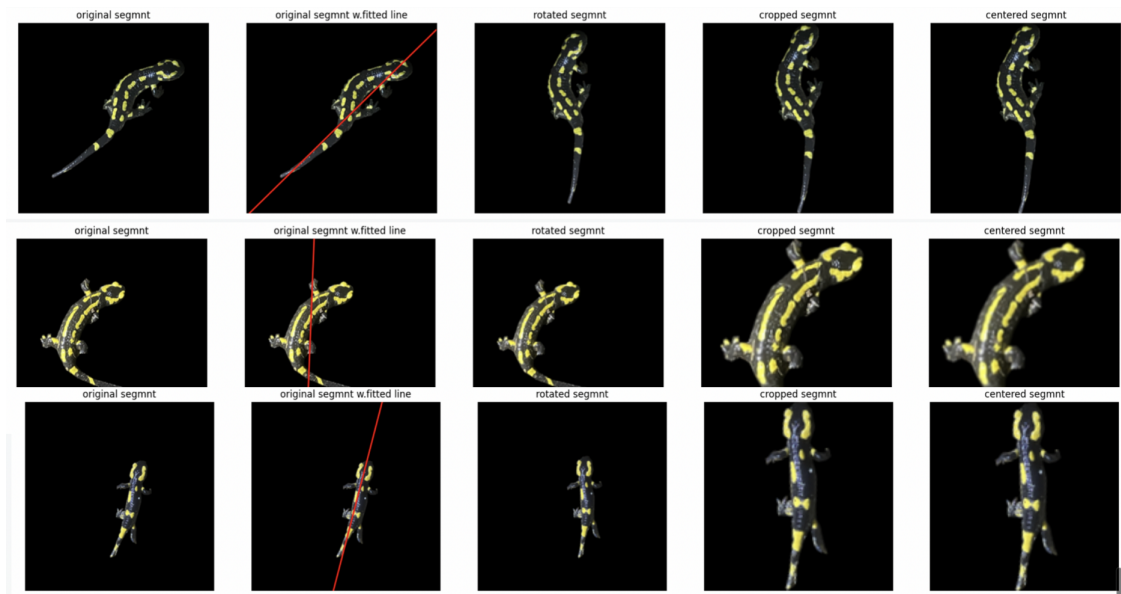


Figure 6.12: Affine transformation on salamander images.

6.2.2 Curved salamanders

As highlighted in the previous section, one of the challenges encountered in the alignment step is the presence of curved salamanders in the dataset. Despite the desired objective of straightening all salamanders using affine transformations, there are instances where certain salamanders exhibit inherent curvature that cannot be effectively addressed by the mentioned techniques.

In our case, the curved salamanders represent a small subset within the grouped dataset, but their curved shape does not reflect the natural position of the animal. To address this issue, we sought a straightforward approach to handle the presence of curved salamanders. Our objective was to find a simple and effective solution that could detect the curved salamanders in the images and straighten them when possible. By leveraging mask contours, we aim to identify and quantify the degree of curvature in salamander images, providing valuable insights for the subsequent stages of identification and analysis.

Figure 6.13 displays a number of contour features investigated to determine the presence of a curved salamander. We can observe in the four columns of Figure 6.13 (i) the original salamander mask, which represents the raw binary image of

the salamander silhouette. (ii) the convexity defects, highlighted by points or regions where the contour deviates from its convex hull. (iii) the convex hull, which is the smallest convex polygon that completely encloses the salamander contour. It provides an approximation of the salamander's outer boundary. (iv) the approximated curve that closely fits the salamander's body with highlighted points of interest (more in Figure)

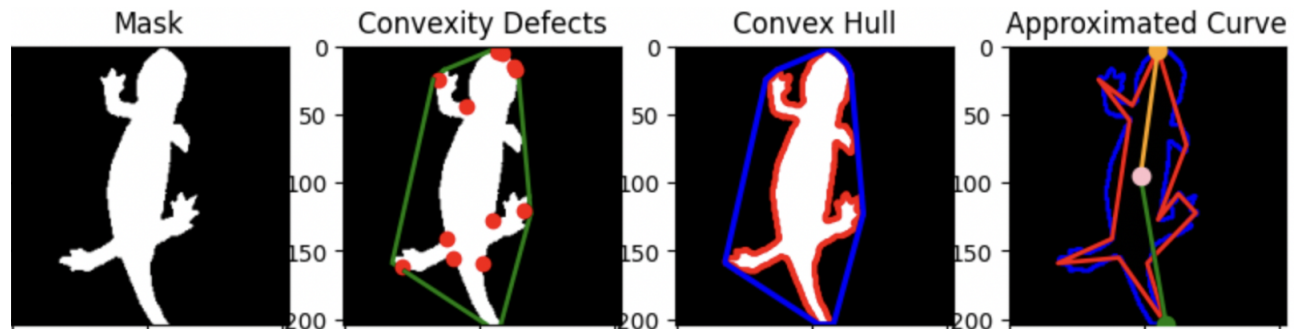


Figure 6.13: Contours features for curvature detection.

- **Convex hull:** The convex hull is a contour feature that helps in identifying the outermost boundaries of an object or shape. It is computed using the `convexHull` function in OpenCV, which finds the smallest convex polygon that encloses the salamander contour. The convex hull provides a simplified representation of the salamander's shape by encompassing all the points on the contour.
- **Convexity defects:** Convexity defects are deviations or concavities observed in the contour of an object. They occur when the contour has indentations or irregularities. The `convexityDefects` function in OpenCV is used to detect and measure these defects. One intuition was that analyzing convexity defects, we can identify regions of significant curvature or irregularities in the salamander's body shape.
- **Approximated curve:** The approximated curve is a technique used to approximate the salamander's contour with polygon. The `approxPolyDP` function in OpenCV is used to generate the approximated curve by fitting a curve with fewer vertices to the salamander contour. This approximation provides a more simplified representation of the salamander's shape, making it easier to analyze and align.

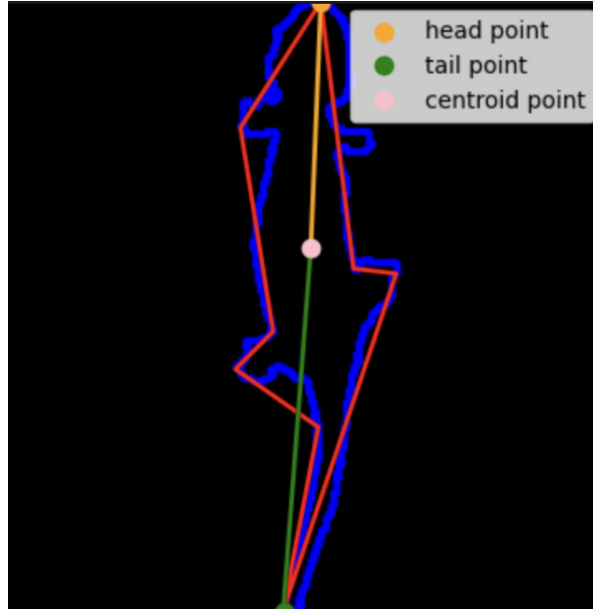


Figure 6.14: Approximated curve and key points for curvature detection.

The approximated curve can be leveraged to detect the curvature of the salamander body by defining a set of points of interests and computing the angle between them. Precisely, we define the head point as the highest point in the approximated polygon and the tail point as the lowest point in the approximated polygon. Using the centroid of the foreground mask, we define the upper body axis as the line combining the head point and centroid. Similarly, the lower body axis joins the tail point and centroid. A straight salamander as portrayed in our template of Figure 6.10, has an angle of 180° . Hence, a curved salamander can be detected by comparing the angle between the upper body axis and lower body axis with a chosen threshold.

Figure 6.14 shows the approximated curve of a mask sample with the three defined key points. The upper body axis is plotted in orange whereas the lower body axis is plotted in green. We can observe that the two lines approach an angle of 180° which reflect the desired straight salamander position.

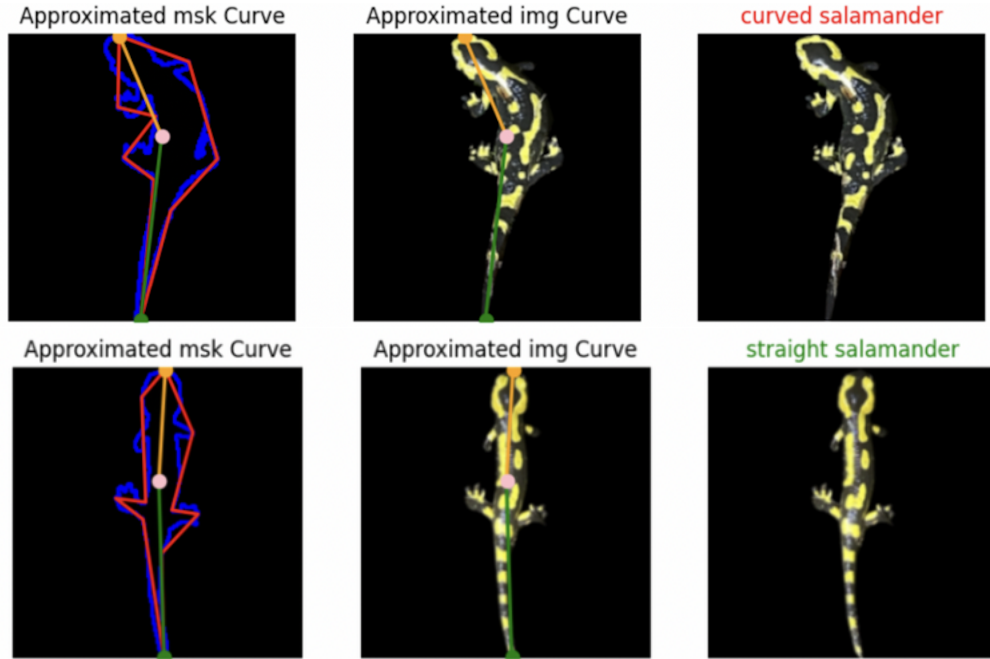


Figure 6.15: Example of curve detection.

Figure 6.15 exhibits two examples of our simple curvature detection based on the approximated curve feature. The three columns of Figure 6.15 depict : (i) the approximated curve on the mask contour along with the derived key points (ii) the three key points of interest, extracted from the approximated curve, on the image sample (iii) the segmented image with the assigned category (curved or straight).

Evidently, we can notice on the approximated curve column, that the first example shows a curved position while the second shows a straight position. After some trials we decided to set the threshold for the angle to the value of 165.

As previously mentioned, the salamander is naturally found in most instances in a straight position. Observations of the animal in its natural habitat show that in often cases: when photographing an encountered individual, if the animal is curved, it will most likely resume to a straight position within seconds as it moves. This simple curve detection step can help us rule out those odd situations from the image set. Nevertheless, to preserve a maximum of images from our dataset, we decided to implement an additional upper body rotation. This consists in aligning the upper body axis, which joins the head point to the centroid, with the vertical axis as shown in Figure 6.16. This straightforward solution can be applied to samples which portray mild distortion.

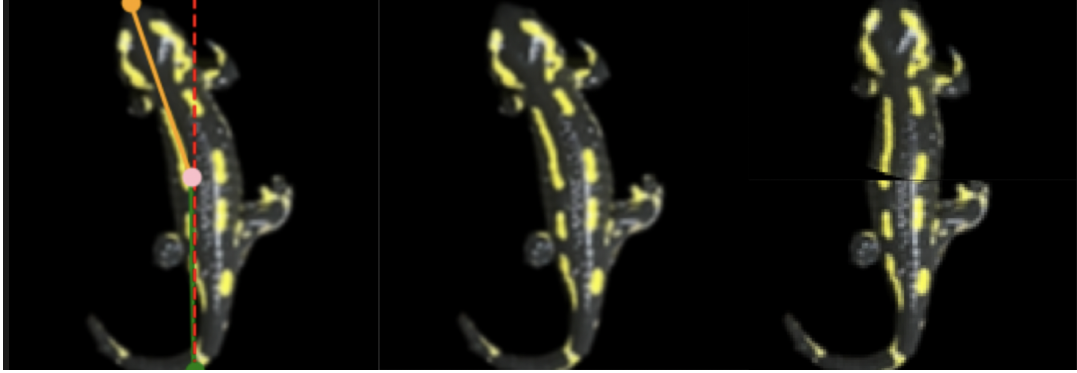


Figure 6.16: Upper body rotation.

6.3 Patch-wise approach

The second identification approach explored within this project, following the LBP method, involves a patch-wise strategy that leverages image moments as primary features at the patch level. Inspired by the character classification process in OCR (Optical Character Recognition) [25], this approach takes as input binary images of the aligned pattern masks and compares sub-regions of the identity pattern for a classification at the patch level. The expectation with this patch-wise approach is that the divide-and-conquer strategy will provide more discerning power in this context where spot-patterns might resemble each other in their overall appearance. The workflow devised for pattern identification hinges on several steps:

The first steps as depicted in Figure 6.17 focus on the organization of the patches and their representation. (i) Patch Generation: Patches are generated for each binary image, guided by hyper-parameters such as patch size and stride. (ii) Zero-Patch Discarding: Patches without any foreground content—termed zero-patches—are eliminated from consideration as these background patches are identical across all data samples and labeled with various identities thus constituting noise. (iii) Feature Vector Reduction: Patches are condensed into feature vectors, a process further elucidated in the subsequent point. (iv) Bag-of-Words Style Gathering: The obtained patch vectors are collated in a bag-of-words [38] manner, forming the foundation for the classification. (v) Classifier Training: A classifier is trained using labeled patch vectors, facilitating the learning of patterns. K-Nearest Neighbors classification was investigated for the task along with the Support Vector Machine algorithm with a one-vs-one method. The prediction of new sample identities, is accomplished through a majority vote among the patch-level classifications.

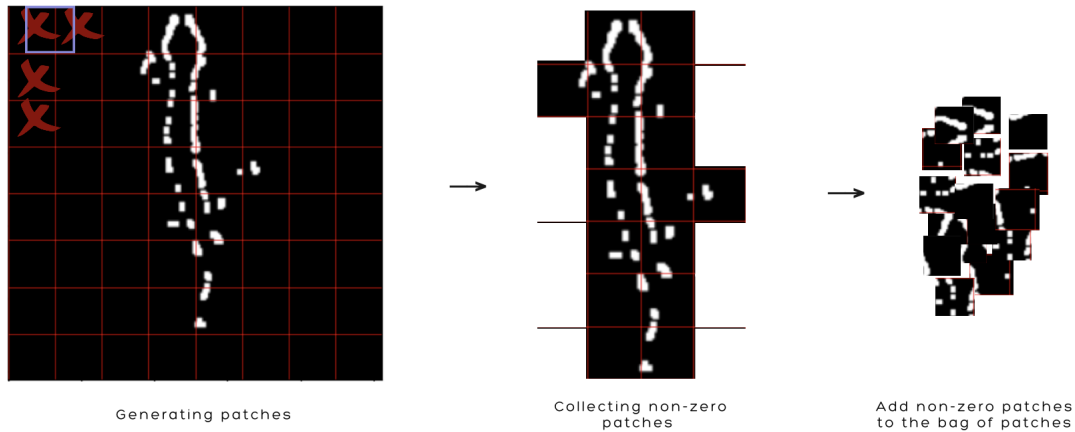


Figure 6.17: Construction of the bag-of-patches.

Figure 6.17 depicts the two first steps of the patch-wise approach which are the patch generation and the selection of non-zero patches to include in the bag-of-patches. This operation is repeated for all spot patterns images.

6.3.1 Feature Space

As highlighted in the introduction to this patch-wise approach, the classification employs two straightforward algorithms. One approach involves introducing a separator between samples within the feature space, while the other relies on the proximity of similar samples to infer class labels. In both instances, the feature space's role is crucial, as it holds the potential to significantly impact classification outcomes based on sample positioning. Our objective is to represent patches through meaningful feature vectors, capturing subtle distinctions between bloboid patterns and grouping similar ones. Investigated features include pixel intensities, image moments and patch coordinates.

- **Pixels:** A first feature to explore for the comparison of the patches is the pixel intensities. This comes as a of the initial intuition drawn from OCR (Optical Character Recognition) techniques. This feature can be included using the values of all the pixels or by lowering the number of dimensions of the data.
- **Image Moments:** Within the OpenCV library, image moments which we introduced in chapter 3, comprise 24 values. These values include the three categories of moments with 10 raw moments, 7 central moments and 7 normalized central moments. Image moments are expected to capture intricate information about patches' patterns and structures.

- **Coordinates:** In order to maintain the spatial information associated with each patch generated in relation to the original image and, consequently, the position of the pattern patch across the animal's body, (x,y) coordinates are computed based on the centroid of the pattern.

6.3.2 Classifier

Having established an effective feature representation for the patches, the remaining identification step involves classification, as detailed in the workflow description of this approach. The primary classifier employed is the K-Nearest-Neighbors (KNN) classifier, functioning through a majority vote mechanism. Moreover, we explored the application of Support Vector Machine (SVM) classification within the same feature space. The limitations of the latter will be discussed.

- **KNN:** A KNN classifier as explained in chapter 3 is a simple yet efficient supervised classification algorithm. An initial approach undertaken during first experiments consisted in training multiple classifiers for different patch regions; for example one KNN would classify head patterns while another would be trained solely on patches of lower regions. Introducing overlap in the window parse of the binary image quickly discarded this approach as the growing number of regions would make the training and prediction of such large set of classifiers computationally inefficient. Additionally, the coordinates introduced in the feature vector compensated for the initial need for the separation of the patches into different regions prior to the comparison. Instead, we now have one KNN classifier trained on the bag-of-patches. The distance measure used is the mahalanobis distance and K is set to 5 neighbors. It is worth noting that given the specificity of the data at hands being salamander individuals which show a rare maximum frequency of 3-4 appearances for an individual, the predicted patch often has 5 different labels among its nearest neighbors.
- **SVM:** The SVC module from Sklearn [32] was used. It employs the "one-vs-one" strategy for multi-class classification. After conducting multiple experiments and fine-tuning the hyper-parameters of the multi-class SVM classifier, a distinct behavior was observed. Following the training of the classifier, a single identity was consistently predicted for all input spot patterns. This assigned identity corresponded to one of the most frequently occurring individuals. This behavior results from the highly imbalanced nature of the salamander individual tracking data, as highlighted in chapter 4. The SVM's decision boundary displayed a significant bias towards classes with a greater number of samples, resulting in less than optimal performance for

the minority classes. This phenomenon arises from the SVM's pursuit of a hyperplane that maximizes the margin between classes while minimizing classification errors.

6.3.3 Results

In this subsection the focus will be on the interesting observations noticed across the different experiments conducted. One tool leveraged in this context is visualization. The goal behind such experiments is to lay a first assessment of the effectiveness of the approach chosen and understand the impact of different features on the classification. A first visualization experiment focused on the behavior of image moments bar charts between different bloboid patterns. The objective was to closely verify if such a promising feature could capture the fine differences that embody the contrast between individuals in the population. A second visualization experiment focused on the neighbors selected by the KNN. This allows us to verify if similar patches are clustered together in the feature space. Additionally, the second visualization experiment more than numerical metrics allowed us to understand which features introduced noise in the classification and how they impacted it.

Image Moments visualization

Image moments are by definition weighted averages of pixel values that describe the foreground of a binary image. We thus expect moment values to show synchronized behavior with the changing of the patterns as different categories of moments rise and drop to represent opposite curves and shapes in the stray of spots. The remaining question, regards the level of detail at which image moments can still discriminate between different identities. The goal of this visualization experiment is to unveil the power and limits of image moments in our specific context.

Figure 6.19 and Figure 6.18 present four interesting comparisons between pairs of patches. Each row consists of five sub-figures: one patch and four bar charts representing the moments of that patch, all presented on the same line. Precisely, the four bar charts display: (i) Moments of the image: (left of the patch) displays all 24 moments in a single plot, thus highlighting the scale different between them. (ii) Spatial moments of the image: (right of the patch) the blue bars each represent one of the 10 spatial moments of the image. (iii) Central moments of the image: the yellow bars represent the central moments. (iv) Central normalized moments: the green bars represent central normalized moments.

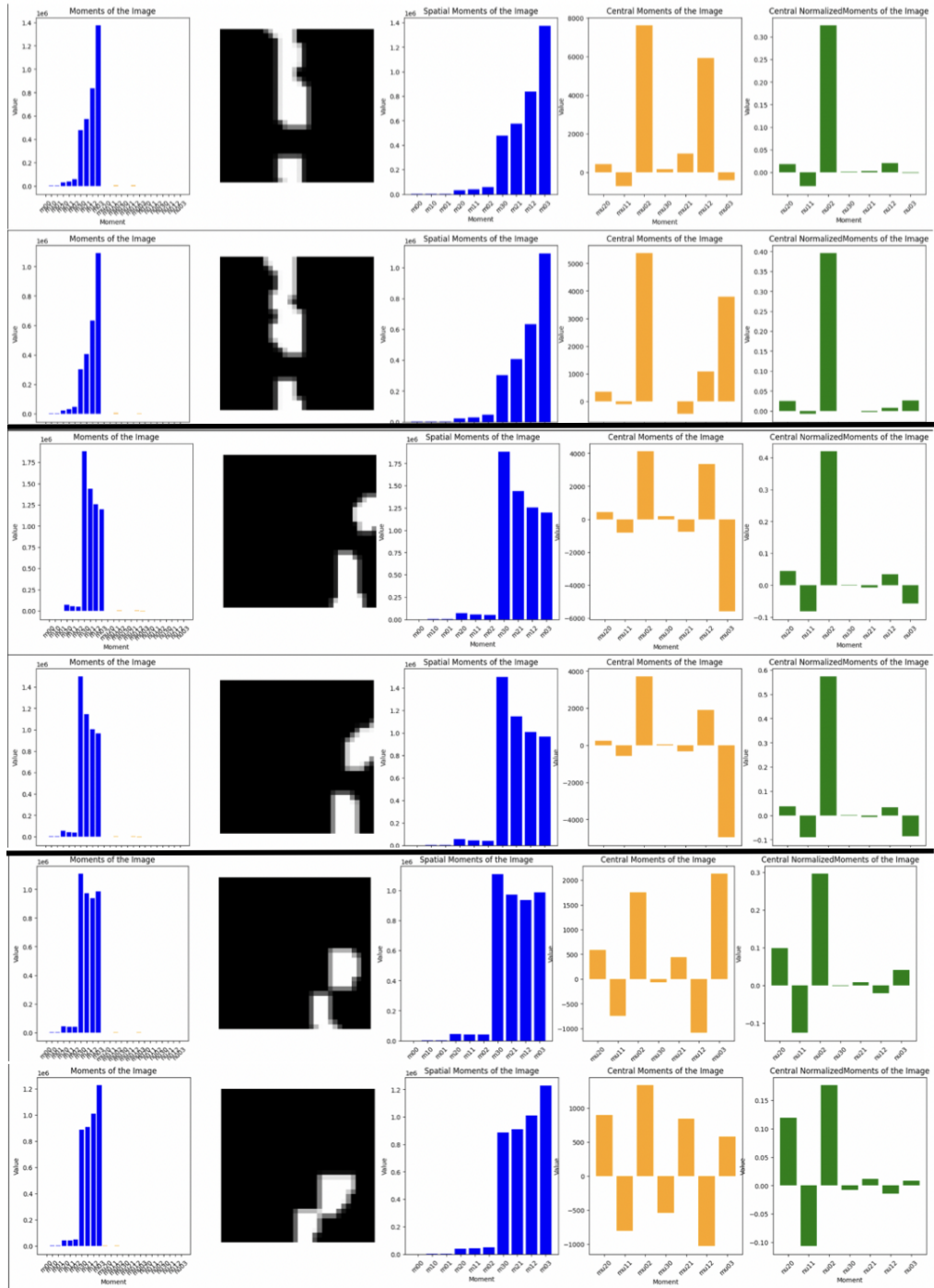


Figure 6.18: Visualization of image moments for pairs of similar patterns.

Figure 6.18 is composed of four pairs of highly similar patches. Across these pairs, distinct parallels in the plotted moment values are evident. In fact, the behavior of the majority of moments remains consistent between the two patches constituting each pair of compared patterns. The remaining moments that do not align with identical values still exhibit the same sign and are closely matched in scale. A clear correspondence between visually similar patches and similar moment values is concluded.

Figure 6.19 depicts a pair of patches that exhibit a certain degree of similarity, with distinct features on each side. It is evident that both patches show the right side of an elongated pattern print, composed of smaller discontinuous spots. However, upon closer examination, subtle discrepancies emerge between these patches. In particular, when scrutinizing the individual smaller spots that constitute the entire line, variations become apparent. These differences include the number of spots, their shapes, and precise locations in relation to other spots. Other dissimilarities displayed in Figure 6.19 lie in the image moments' bar charts. Specifically, the central moments indicated by the yellow bars for both patches display significant disparities. Consequently, we can deduce that image moments possess the capability to discern intricate nuances within pattern patches.

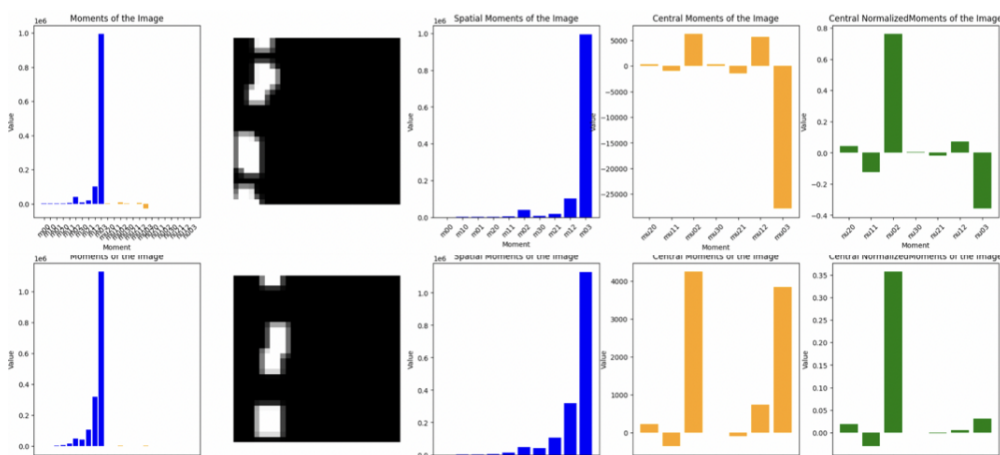


Figure 6.19: Visualization of image moments for a pair of patterns.

Neighbors visualization

In order to verify how neighborhoods are formed in our feature space, one visualization experiment thought of consists in displaying some of the nearest neighbors selected by the trained KNN model for a chosen patch. This operation is possible when using the `KNeighborsClassifier` from `sklearn` library [32]. The module includes a *kneighbors* function which returns the indices and distance of the nearest neighbors selected by the classifier, for a given sample. This visualization uncovered that the use of all pixel intensities of the patch, which contribute to 1024 values of the feature vector for a 32x32 patch size, against 24 for the image moments and 2 for the patch coordinates, introduced a bias towards patches with a high background to foreground ratio, as can be seen in Figure 6.3.3.

Figure 6.3.3 is composed of three rows of patches. Each row starts on the far left with the patch introduced to the KNN classifier followed by its nearest neighbors. Further experiments revealed that the discriminative power lies within the 24 values which constitute the image moments. These experiments showed that neighborhoods collected from patches expressed through their image moments yielded better results. Indeed as we can see in Figure 6.3.3, the visualization of the samples selected alongside their nearest neighbors shows clear similarity of the patterns. We can first notice across all four rows of Figure 6.3.3 that the use of image moments alone allows for the grouping of patches of same body parts without the use of patch coordinates. For example the neighbors of a head patch, in row four, are other head patches (we recognize head spots of a salamander by their a semi-circular form that can be likened to two halves of an oval shape sometimes divided along the middle). More importantly, aside from recovering equivalent patches in terms of body location, it is interesting to see that the nearest neighbors of a given patch show similar spot characteristics. For instance, both the first and second rows in Figure 6.3.3 display elongated lines of spots that correspond to the left half of a dorsal pattern. However, within the neighborhoods, the first row presents nearly flawless lines of spots with minimal separation, while the second row showcases clearly separated lines.

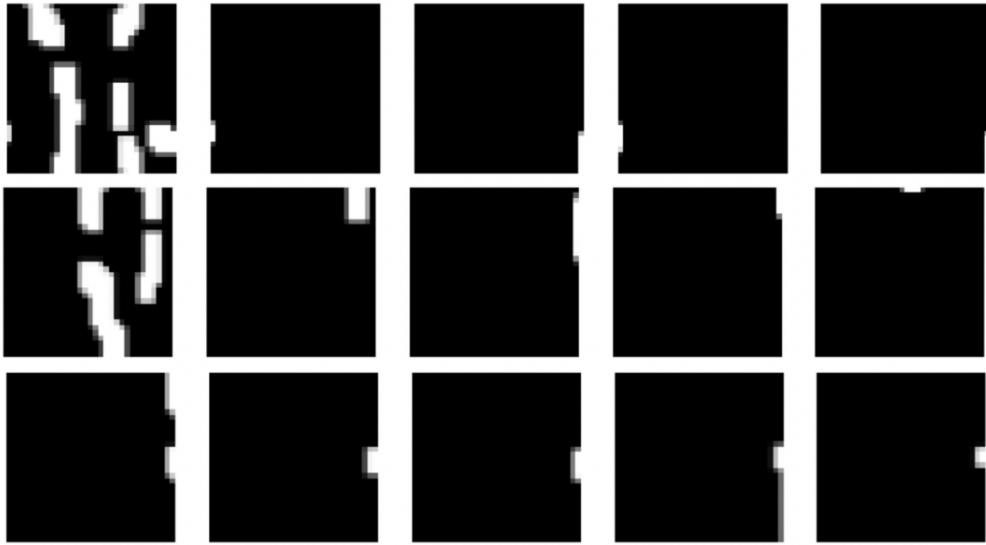


Figure 6.20: Visualization of patch neighbors with pixel features.

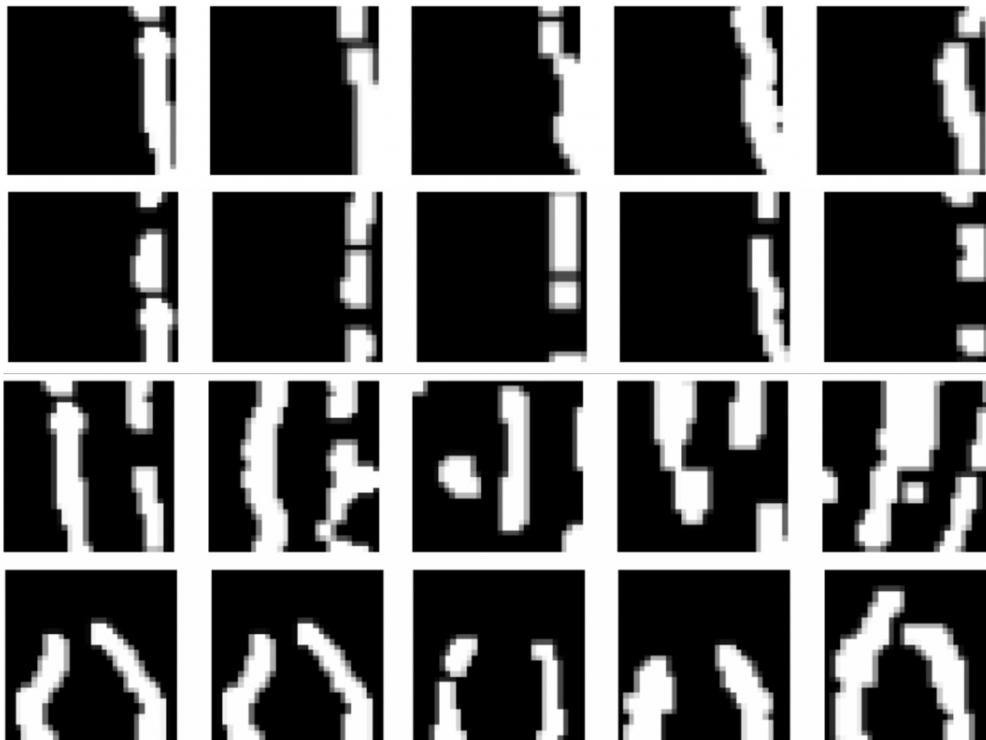


Figure 6.21: Visualization of patch neighbors based on image moments features only.

6.4 Conclusion

In this chapter, the focus was on the identification task that lies at the heart of the population tracking problem for animals. We began by painting a general picture of the techniques used by researchers in the field of animal individual identification and defined the specific category of approaches that our work aligns with.

Our first attempt at this task relied on Local Binary Patterns (LBPs). The motivation behind the use of LBPs originated from the known ability of its circular variant (CLBP) for the description of blob objects. However, the histograms produced by this method were not distinctive enough to accurately distinguish between the patterns presented.

Following the limitations encountered by the LBP approach, we decided to introduce better standardisation in our data by conducting body alignment on the segmented salamanders. We presented the different transformations applied on the data, mainly affine transformations, to reach a straight and centered format of uniform scaling.

Our second attempt, using aligned pattern images, relied on a divide and conquer strategy. It consists of a patch-wise approach that primarily relies on image moments to summarize the crucial information of the spots, followed by a classifier trained on the generated patches. Multiple choices are possible with this approach; the first involves engineering the feature vector to better describe the patches. Alongside image moments, positioned at the core of this approach, we mentioned pixel intensities and coordinates of the patch in its original image as interesting options. The second choice concerns the classifier. Two algorithms were discussed in this context: (i) K-Nearest Neighbors (KNN) and (ii) Support Vector Machines (SVM). The latter was quickly dismissed due to the imbalanced nature of the salamander identification data, leading to a strong bias toward the class with the most occurrences.

First assessment of the patch-wise approach through visualization yielded positive outcomes. Indeed, visualizing image moments via bar charts for selected pairs of pattern patches revealed indications of discriminative capability, successfully highlighting subtle distinctions between similar patterns exhibiting fine differences. Furthermore, visualizing the nearest neighbors of patches selected by a KNN model trained on image moment vectors further solidified the confidence in the viability

of this handcrafted feature as it grouped highly similar patches is the formed neighborhoods.

In the next chapter, we will further discuss the results of the patch-wise identification approach based on individual identification metrics.

Chapter 7

Results and analysis

In this penultimate chapter, we share formal results of the patch-wise approach for the task of salamander individual identification. We begin by presenting the set of evaluation metrics used in the field. Next, we share the results obtained in the two testing scenarios we defined. The first scenario focuses on recurrent identities while the second concerns identities characterized by one unique encounter. Finally, we conclude by discussing the observed results which unveiled the limitations of the approach.

7.1 Evaluation metrics

Building upon the accuracy metric defined in chapter 3, we now introduce a set of commonly used metrics in the context of animal re-id and recognition systems.

The *top 5* accuracy consists in checking for each image to predict, called query, if any of the top five identities returned by the identification model correctly correspond to the query individual. Analogically, *top 1* accuracy verifies if the predicted identity is the correct one. This metric can be generalized to *top m* accuracy where m is the rank. The popularity of this metric in this field aligns with the *suggestion* feature implemented in identification software. This feature consists in suggesting up to ten similar gallery images for the query image. The user can then select the corresponding identity if present among the batch or dismiss the suggestion.

7.2 Closed set evaluation and discussion

A closed set describes a scenario in which the query image has a corresponding identity within the available dataset of identities. In this scenario, the goal is to

verify if the system can assign the correct class to a non-novel individual. Closed set tests, implemented for individuals with at least two appearances, work as follows: For an individual with k appearances, 1 image is selected as a query image and $k-1$ remaining images are included in their reduced patch form; in the training data of the KNN classifier. The operation is repeated k times.

In the described closed set evaluation method, the patch-wise approach reached 52.50% top-5 accuracy where the rank is defined by the decreasing order of vote frequency. These disappointing results are discussed in this section. One important observation to make regarding the presented results is that vote frequencies for closed set query images reached surprisingly low values for such context. Indeed, max vote frequencies which define the predicted identity revolved around 10 votes for query images composed of approximately 50 patches. Additionally, inspection and comparison of the correctly identified identities in this closed set evaluation showed a common trait of near perfect alignment of bodies.

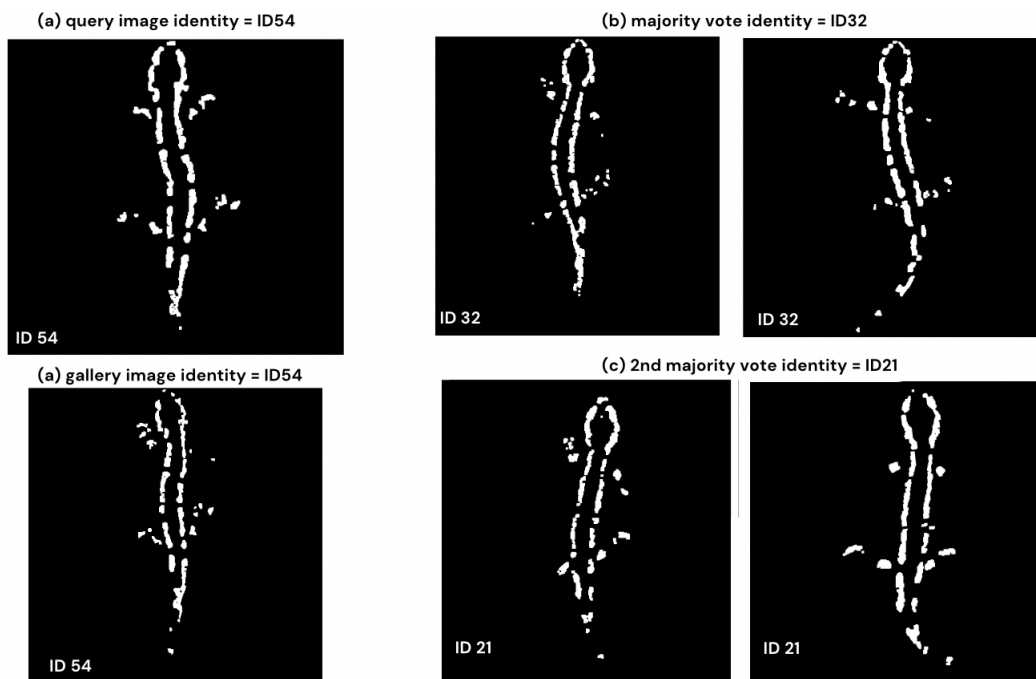


Figure 7.1: Comparison of query image to voted identities.

Figure 7.1 illustrates one flaw unveiled by the closed set evaluation. Query image corresponding to ID54 was voted as ID32 with ID21 as a second comer in majority votes. Clear similarity between the patterns is visible. Unfortunately, in the context

of a closed set evaluation, the query’s true identity is represented in the training data by only one remaining image post-holdout. In this context, when identical patches are clustered together in a KNN neighborhood, the remaining true-identity patch is outnumbered by other identities that contain an identical patch as part of their similar pattern.

Despite the promising outcomes showcased in chapter 6 concerning the image moment feature, it has become evident that employing the image moments representation of the generated patch through a KNN is sufficient solely for the clustering of visually similar entities. An important aspect overlooked during the design of this approach is as follows: A pattern distinct from the query image can accommodate an identical patch signifying a shared spot within the complete identity print. This single, akin patch could receive votes at the patch level. Although a single misassigned vote might appear inconsequential in theory, closed set evaluation has unveiled a consistent pattern: across all introduced query images, regardless of correct identification or not, the highest vote frequency for all patch predictions remains at one vote.

7.3 Open set evaluation and discussion

An open set describes a scenario in which the query image has no corresponding identity within the available dataset of identities. In this scenario, the system is presented with a novel individual.

To facilitate a final comparative assessment between the current method and the LBP approach, we revisited the experiment outlined in Figure 6.9. This experiment involved the introduction of modified copies by incrementally obscuring various segments of the spot patterns. The replication of this experiment is visually presented in Figure 7.2. The query image (a) along with the three modified copies (b), (c) and (d) were submitted to the patch-wise approach. In contrast to the distance scores calculated for each pair in the LBP scenario, we now judge the behavior of the method based on the number of votes. Given the perfect alignment of all variants of the query image, prediction of the true id was preserved for all variants. Vote frequencies reflected the incremental obscuring of the identity. They dropped from 44 votes out of 62 for the perfect duplicate of query image (a), to 37 for variant (b), 33 for variant (c) and 31 for variant (d).

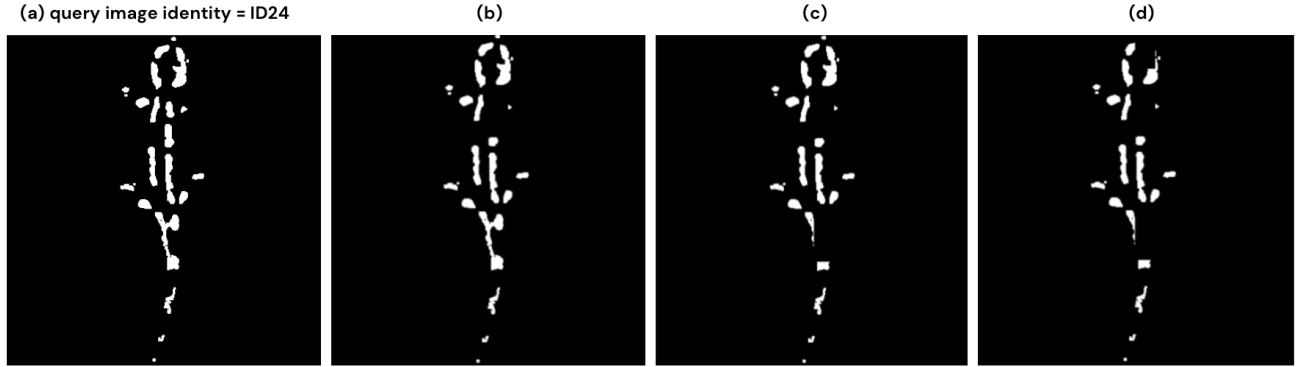


Figure 7.2: Variant of comparison experiment.

7.4 Conclusion

In summary, this chapter has conducted an examination of the patch-wise approach as an overall recognition system for salamander individual identification. This investigation has hinged on formal evaluation metrics to illuminate the performance of the system under both closed and open set scenarios.

Initially, our exploration was underpinned by a promising discovery in our analysis of the patch-wise approach detailed in Chapter 6. We found that image moments had a remarkable ability to effectively group together closely resembling patches.

However, as our study delved deeper into the systematic assessment of this approach, it revealed its limitations and flaws. One pivotal observation emerged: the accurate identification of individual salamanders hinges significantly on the alignment quality of body patterns within the image gallery. This factor, evident in our closed set evaluations, exposes the vulnerabilities of our approach.

Another discovery concerns the divide and conquer strategy that initially aimed at enabling a robust approach for photographs of the same individual that show discrepancies in some isolated parts of the pattern, for instance a missing tail. Unfortunately, it appears that as we divide all individual patterns a side effect is that commons spots to different individuals all become suited voting options at the patch level.

Indeed, the observed percentages for top-5 accuracy, as reported in our findings, have fallen notably below our initial expectations. These results, while perhaps not

aligning with our initial optimism, offer valuable insights into the intricate nature of salamander identification.

Chapter 8

Conclusion

The main goal of this master’s thesis was to conduct experimental research regarding the individual identification of salamanders, employing handcrafted features from their extracted identity patterns. Within this framework, our initial contribution involved enriching the available salamander dataset for segmentation and constructing a tailored identification dataset. From a technical standpoint, our contribution is twofold: first, we concentrated on the extraction aspect of the hypothesis by training a semantic segmentation model using a U-net architecture, followed by color segmentation using K-Means clustering. Second, we explored two promising paths of identification approaches.

The segmentation process of the salamander’s body hinged on the success of our semantic segmentation model. This accomplishment was underpinned by the model’s training on the meticulously formed dataset. Subsequently, the model underwent testing and evaluation, with valuable feedback from an observation volunteer, elaborated in Appendix B. The following color segmentation exhibited notable effectiveness. A subset of hand-selected images underwent additional processing, as presented in Appendix A. This segmentation journey empowered our work, laying the foundation for the investigation of the identification methods that followed.

Two distinct identification methods were conceived, implemented, and scrutinized. The first approach centered around Local Binary Patterns (LBP) as a descriptor of the salamander’s identity patterns. In this process, the segmented pattern was transformed into an encoded image using the chosen LBP variant. The encoded image was then partitioned into neighborhood blocks, each yielding a local occurrence histogram. The combination of these local LBP histograms formed the global feature vector. Identification was conducted based on one-to-one similarity score calculation grounded in a distance measure between feature vectors. However,

for this initial method, a clear inadequacy for the specific case of salamander patterns was quickly recognized upon initial explorations, despite the utilization of different LBP variants and distance measures. Considering the challenges encountered in the identification task, we made the decision to incorporate a body alignment step into our work.

As we delved into our second identification method, the instinct was to craft a patch-wise strategy inspired by recognition approach found in OCR systems. Guided by this idea, our primary focus was to identify an appropriate distance measure and discover a more fitting feature vector for the representation of the generated patches. The Mahalanobis distance was selected as the preferred measure for quantifying distance. Concerning the feature space of these patches, among the explored options, image moments emerged as an exceptionally good contender. They demonstrated the capacity to capture subtle differences, effectively discerning fine differences within similar patterns. However, as our investigation systematically probed deeper into this approach, it unveiled inherent limitations and flaws. We now conclude that while image moments indeed embody the optimal choice for the feature space within this patch-wise methodology, which relies on patch classification, it is clear that patch classification with a KNN model might not be the most effective means to leverage the discriminative potential of image moments.

Building on the insights from the literature and the expertise gained throughout this project, along with the results of the patch-wise approach, we envision a potential path for further exploration. This path involves a revised iteration of our patch-wise approach. The suggested modification consists in implementing a one-to-one similarity score calculation between the query image and all gallery images, followed by a descending sorting of scores to facilitate match selection—in contrast to the one-to-many concept we previously tested. Our underlying hypothesis stems from the belief that executing a one-to-one comparison between the query image and gallery images can effectively counteract the numerous identities contributing to the one-vote frequencies. This frequent occurrence is a result of the presence of a single identical patch within otherwise distinct patterns. By shifting to a one-to-one comparison strategy, we anticipate an enhancement in the selection process. This adjustment could lead to more accurate identification matches, as it gives genuine identity matches a better chance of emerging amidst the selection.

Bibliography

- [1] Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer International Publishing, 2018. DOI: 10.1007/978-3-319-94463-0. URL: http://ndl.ethernet.edu.et/bitstream/123456789/88552/1/2018_Book_NeuralNetworksAndDeepLearning.pdf.
- [2] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. “Face Recognition with Local Binary Patterns”. In: *Machine Vision Group, Infotech Oulu* (2004). URL: <http://www.ee.oulu.fi/mvg/>.
- [3] Yuhan Bai. “RELU-Function and Derived Function Review”. In: *SHS Web of Conferences* 144 (Aug. 2022), p. 02006. DOI: 10.1051/shsconf/202214402006.
- [4] Biodiversité en Wallonie. *Salamandre tachetée (Salamandra salamandra)*. 2023. URL: <http://biodiversite.wallonie.be/fr/salamandra-salamandra.html?IDD=50334413&IDC=587>.
- [5] David Crouse et al. “LemurFaceID: a face recognition system to facilitate individual identification of lemurs”. In: *BMC Zoology* 2 (2017), p. 2. DOI: 10.1186/s40850-016-0011-9.
- [6] François Duchêne. “Exploration de nouvelles techniques d’identification pour le recensement d’espèces protégées : application pour le suivi de la population de salamandres”. Master’s thesis. Louvain-la-Neuve, Belgium: Université catholique de Louvain, 2022. URL: <http://hdl.handle.net/2078.1/thesis:35556>.
- [7] Margherita Grandini, E. Bagli, and Giorgio Visani. “Metrics for multi-class classification: an overview”. In: *ArXiv*, abs/2008.05756 (Aug. 2020). URL: <https://arxiv.org/abs/2008.05756>.
- [8] J. A. Hartigan and M. A. Wong. “Algorithm AS 136: A K-Means Clustering Algorithm”. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. URL: <https://doi.org/10.2307/2346830>.

- [9] M. Hassaballah, Hammam A. Alshazly, and Abdelmgeid A. Ali. “Ear recognition using local binary patterns: A comparative experimental study”. In: *Expert Systems With Applications* 118 (2019), pp. 182–200. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.09.037. URL: <https://www.sciencedirect.com/science/article/pii/S0957417418309373>.
- [10] Lex Hiby et al. “A tiger cannot change its stripes: Using a three-dimensional model to match images of living tigers and tiger skins”. In: *Biology letters* 5 (Apr. 2009), pp. 383–6. DOI: 10.1098/rsbl.2009.0028.
- [11] Benjamin Hughes and Tilo Burghardt. “Automated Visual Fin Identification of Individual Great White Sharks”. In: *International Journal of Computer Vision* 122.3 (2017), pp. 542–557. DOI: 10.1007/s11263-016-0961-y.
- [12] Shruti Jadon. “A survey of loss functions for semantic segmentation”. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE. 2020, pp. 1–7.
- [13] Renaud Lafourt. “Les facteurs abiotiques déterminant la répartition de la Salamandre tachetée (*Salamandra L. 1758*) dans les massifs forestiers de Stambruges et de Beloeil”. Orientation : forêt et nature. Bachelor thesis. Ath, Belgium: Condorcet, 2022.
- [14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440.
- [16] Vincent Miele et al. “Revisiting giraffe photo-identification using deep learning and network analysis”. In: (Mar. 2020). DOI: 10.1101/2020.03.25.007377.
- [17] Natagora. *La salamandre terrestre*. 2023. URL: <https://www.natagora.be/la-salamandre-terrestre>.
- [18] Natagora. *Mode d’emploi pour l’encodage des résultats des inventaires salamandre*. Online document. 2023. URL: <https://docs.google.com/document/d/1hwV8KTSKHjtx7Uu5zZtJARaRcwZq5Kfsg0AvYuY0-eo/edit>.
- [19] Natagora. *Protocole de terrain - Enquête Salamandre et Bsal en Wallonie*. Google Docs. URL: <https://docs.google.com/document/d/18XTfie4RxJaJMVj1NaDw2S4GZj/edit>. 2023.
- [20] Natagora. *Sauvetage des batraciens*. 2023. URL: <https://www.natagora.be/sauvetage-des-batraciens>.

- [21] Natagora Bas-Meuse. *Recensement des salamandres et tritons*. 2023. URL: <https://bassemeuse.natagora.be/recensement-des-salamandres-et-tritons>.
- [22] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. “Learning deconvolution network for semantic segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. IEEE. 2015, pp. 1520–1528.
- [23] Chigozie Nwankpa et al. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. Dec. 2020.
- [24] T. Ojala, M. Pietikainen, and T. Maenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 971–987. DOI: 10.1109/TPAMI.2002.1017623.
- [25] OpenCV. *OCR of Hand-written Data using kNN*. Accessed on 2023-04-05.
- [26] Janakkumar Baldevbhai Patel and R. S. Anand. “Color Image Segmentation for Medical Images using L*a*b* Color Space”. In: *IOSR Journal of Electronics and Communication Engineering* 1.2 (2012), pp. 24–45.
- [27] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (Jan. 2012).
- [28] *Position Statement: Toe-clipping in Amphibians*. PDF URL accessed on Day Month Year. Association of Reptilian and Amphibian Veterinarians. 2018. URL: <https://arav.org/wp-content/uploads/2018/04/Toe-clipping-position-statement.pdf>.
- [29] Selva Prabhakaran. *Mahalanobis Distance – Understanding the math with examples (python)*. Website: machinelearningplus. 2023. URL: <https://www.machinelearningplus.com/statistics/mahalanobis-distance/>.
- [30] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An Introduction to Machine Learning*. Cham: Springer, 2019. DOI: 10.1007/978-3-030-15729-6. URL: <https://link.springer.com/content/pdf/10.1007%2F978-3-030-15729-6.pdf>.
- [31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: vol. 9351. Oct. 2015, pp. 234–241. ISBN: 978-3-319-24573-7. DOI: 10.1007/978-3-319-24574-4_28.
- [32] scikit-learn contributors. *scikit-learn: Machine Learning in Python*. <https://scikit-learn.org/stable/index.html>. 2023.
- [33] *SciPy: Open source scientific tools for Python*. SciPy Developers. URL: <https://www.scipy.org/>.

- [34] Nahian Siddique et al. “U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications”. In: *IEEE Access* PP (June 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3086020.
- [35] Jeroen Speybroeck and Koen Steenhoudt. “ManderMatcher: a free toolbox for study of fire salamanders”. In: (2017). URL: <https://jeroenspeybroeck.shinyapps.io/mandermatcher/>.
- [36] utaywi and Nezamoddin N. Kachouie. “Author to whom correspondence should be addressed”. In: *Entropy* 23.6 (2021). Received: 18 April 2021 / Revised: 31 May 2021 / Accepted: 4 June 2021 / Published: 16 June 2021, p. 759. DOI: 10.3390/e23060759. URL: <https://doi.org/10.3390/e23060759>.
- [37] Anusha Vupputuri and Sukadev Meher. “Facial Expression Recognition using Local Binary Patterns and Kullback Leibler Divergence”. In: Apr. 2015. DOI: 10.1109/ICCSP.2015.7322904.
- [38] Wikipedia. *Bag-of-words model*. Accessed on 2023-08-12. URL: https://en.wikipedia.org/wiki/Bag-of-words_model.
- [39] *WildID: Online detection and identification of Southern African wildlife species in camera trap images using machine learning software*. URL: <https://www.wildid.app>.

Appendix A

Additional processing

Various image processing techniques have been discussed within this thesis. Some of these techniques were employed as post-processing measures, addressing specific scenarios to rectify potential errors. These corrective steps were automatically applied to all images in the dataset.

Additionally, a last post-processing approach, was employed selectively for particular images. This technique revolves around mitigating glare caused by flashlight reflections on the animal's body which, at times, introduces noise during the color segmentation phase of our work, even with the use of the CIELAB color space.

The additional *anti-flash* processing conducted on the selected images includes the following steps : first, the image is converted to the HSV color space. In this new color representation, the saturation component is examined to identify areas of low saturation, which correspond to less vivid colors. Next, a modified version of the value component is created, with the pixels in regions of low saturation set to zero. This process effectively diminishes the impact of those areas in the subsequent steps. Subsequently, regions of glare are identified by detecting pixels with high values in the modified value component. Finally, these glare-affected regions are inpainted using OpenCv's inpaint method.

Inpainting techniques fill-in missing areas based on their surrounding, sometimes introducing modifications in the image. Given the sensitivity of the task, introducing modifications in the spot pattern would compromise the identification data. Figure A.1 shows two examples of glare correction. Figure A.2 shows a case of a pattern modified as a result of the *anti-flash* method. We can see that the corrected image has connected head spots while the original image does not.

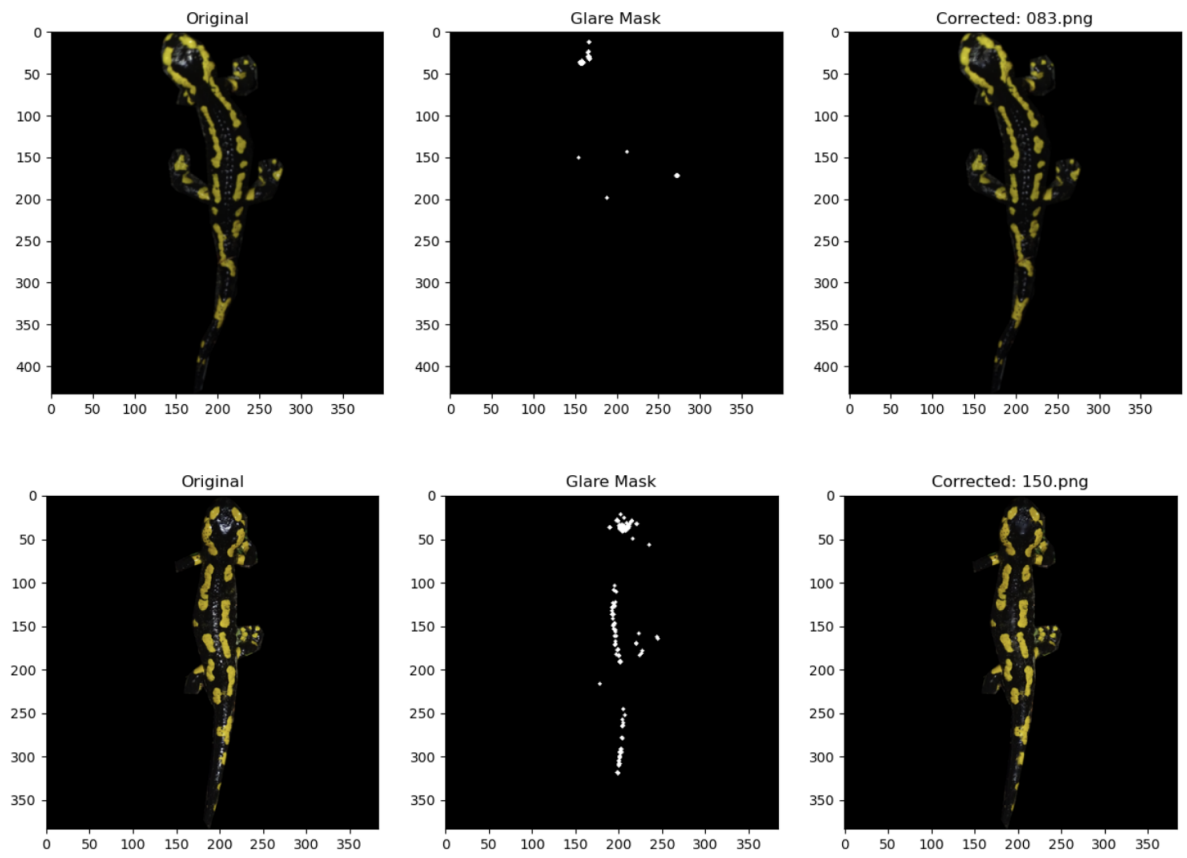


Figure A.1: Examples of glare correction.

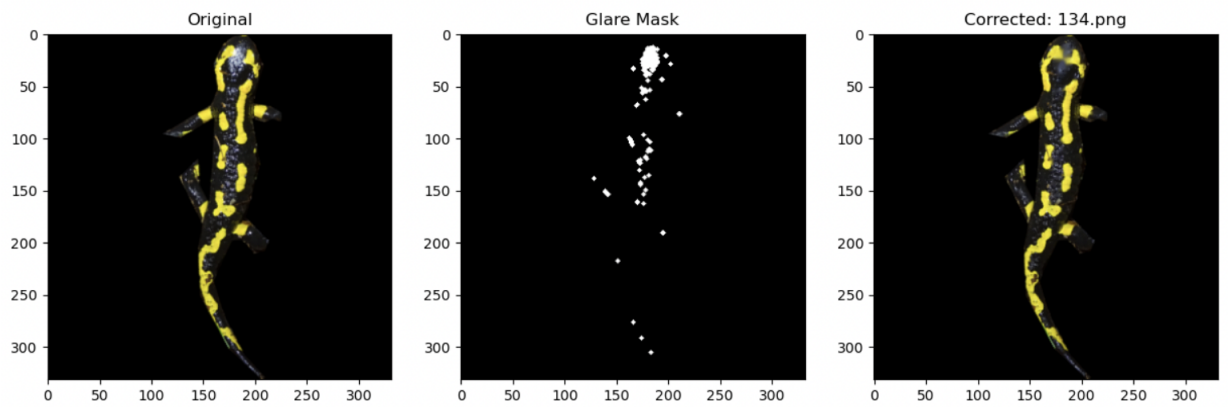


Figure A.2: Example of glare correction introducing spot alterations.

Appendix B

Volunteer Evaluation and Insights: Testing the Segmentation Model

B.1 Context

The segmentation model underwent testing by Adrien Goffin in conjunction with his salamander observation work. Adrien follows a workflow using the WildId giraffe identification software, as highlighted in chapter 1. However, since this software requires pattern shots as input, the collected salamander images need to be manually cropped to achieve more precise spot bounding and to eliminate background noise. This cropping step is often the most time-intensive phase of the process.

B.2 Segmentation results

A trained version of the segmentation model presented in chapter 5 was made available online through a Colab notebook¹. Adrien Goffin’s test made on his personal salamander data evaluated the model’s success rate to 83%. More importantly, this evaluation yielded valuable insights. Adrien’s feedback addressed edge cases and suggested potential enhancements for the real-world utilization of the trained model in volunteer work. One of such cases is displayed in Figure B.1. Such type of fire-salamanders with sheer elongated spots that fade partially into the animal’s black body were not considered in our custom dataset despite being present in many Walloon regions.

¹https://colab.research.google.com/drive/1Q3xjqLYdt_zeg6lKigr6A8zB94_igxHS?usp=sharing



Figure B.1: Fire-salamander category.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl