Automated salamander recognition using deep neural networks and feature extraction

Jørgen Bakløkken*, Felix Schoeler*, Hugo Nørholm*,
Marius Pedersen*, Sony George*, Børre Dervo+
* Department of Computer Science, NTNU
+ Norwegian Institute for Nature Research

Abstract

This paper presents a study conducted to recognize salamanders by using their unique body markings based on images. The detection and matching of unique patterns in a salamander's body can be complex due variability in individual animals size, shape, orientation and also influence from the external enviornment. While traditional methods require time intensive manual image corrections of the salamanders to achieve accurate recognition, in this work we propose a fully automatic techinque for straigthening. We also propose a matching technique based on the corrected images. The convolutional neural network ResNet50 and dense scale-invariant feature transform (DSIFT) are used for belly pattern localization, and matching for salamander recognition.

1 Introduction

To understand the trend of animal life and migratory patterns, it is important to systematically quantify their population in an accurate way. Salamanders are amphibians which experience declination in their population. In Norway there are two different species of salamander, the great crested newt (Triturus crustatis) and the smooth newt (Lissotriton vulgaris). As the great crested newt (Triturus cristatus), has the status near threatened in Norway's national red list, it is crucial to know changes in their population. In order to evaluate the effectiveness of the actions taken in order to increase the number of salamanders, it is important track their population in an accurate way.

There are different ways to count the population of species. One of the most commonly used methods is pit tagging. A pit tag is a small transponder containing a material that is harmless to the animal to which this transponder is the placed. Later the count is performed by scanning its tag ID. However, to implement pit tagging it is necessary to catch the animal, inject a pit tag and document this in the database. This is very time consuming, need careful handling and also stressful for the animal. Image based pattern recognition methods becomes very common for performing population measure, monitoring migration, habitat usage etc. for several animal species because of it is less time consuming and less stressful for the animal. There are notable works in this topic [1].

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

Image based recognition methods solves the issues with pit tagging. However, there exists little efforts towards using automatic image-based pattern recognition for salamanders. Some of the commercial or open source softwares available requires user interventions and manual preprocessing [2, 3, 4, 5]. The great crested newt and smooth newt have body markings of small black patches that are individual-specific [1] (see Figure 1 and Figure 2). These individual-specific body markings can serve as a natural mark for capture–recapture models for estimating population demography. Accurate recognition requires good quality images of the salamanders. This includes proper illumination of the animal while the images are captured, correct orientation of the salamander so that the patterns are included in the images used for recognition. In some cases preprocessing is needed in order to correct the above mentioned factors.

One of the fields of research at the Norwegian Institute for Nature Research (NINA) is salamanders. NINA has interest to learn more about salamander's life cycle, living area and population. To gain more knowledge on this, NINA started a pit tagging project in 2010. In this research, NINA captures images of the salamander and is interested to develop an application that takes an input image of salamander, recognizes the pattern, and identifies whether the same animal is in the database. The objective of the current study is to propose a solution which identifies salamanders based on pattern recognition with a high accuracy, that is comparable to existing solutions. In addition, the solution should reduce the amount of manual work required to match salamanders by automating the workflow. In the current paper, we present results of a study conducted to develop an automated salamander recognition algorithm using deep neural networks and feature extraction.

The paper is organized as follows; first we present the dataset, then further the proposed method, after this we present our results, conclusion and future research.

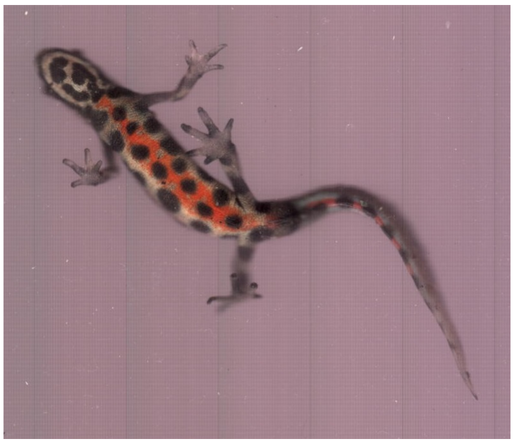


Figure 1: Example image of a salamander.

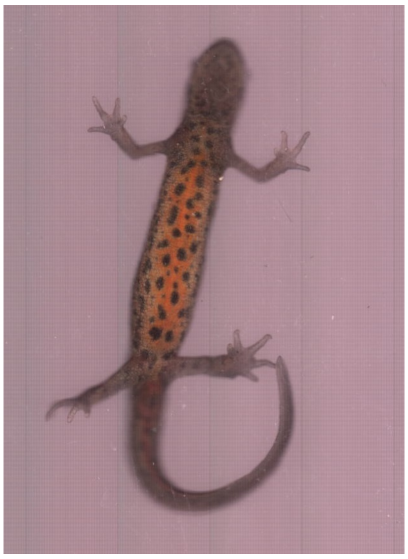


Figure 2: Example image of a salamander.

2 Dataset

The dataset captured by NINA contains a total of 6700 images. These images were of the northern crested newt (Triturus crustatis) and the smooth newt (Lissotriton vulgaris).

The resolution of the images is 1240 x 1748 in JPEG format. Pit tagged information was also available and verify that the images were from the same salamander. Images in the dataset were of different quality levels: this includes images with with partially or fully occluded belly pattern, dark images with low contrast, curved pose, dirt and noisy, varying background, out of focus, overexposed etc. Images with no possibility to extract any information even after preprocessing were excluded from the dataset. Figure 1 and Figure 2 show example images from the dataset.

3 Proposed method

The workflow of the proposed method for salmander recognition is presented in Figure 3.

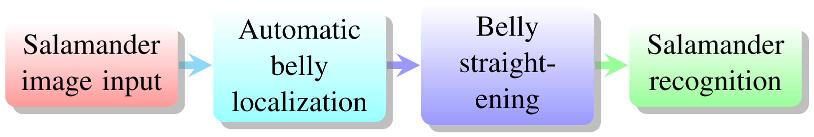


Figure 3: Workflow of the proposed method.

Images of the salamander have been recorded in different conditions and need to be preprocessed before they go to the recognition stage. Preprocessing steps include belly localisation and straightening. In most of the currently available recongition softwares, the preprocessing part is performed manually. One of the main contributions of the presented research is the automated processing of the complete workflow from input image to the recognition.

Belly pattern localisation

Salamander belly pattern

The salamander´s unique patterns is in the body area between the front and back legs. The animal pose is not the same for all the images and in several cases, this needs to be corrected in order to be able to compare patterns accurately. According to A.J. Ijspeert [6], the salamander can be modeled as having three joints in between the front legs and back legs, as demonstrated in Figure 4.

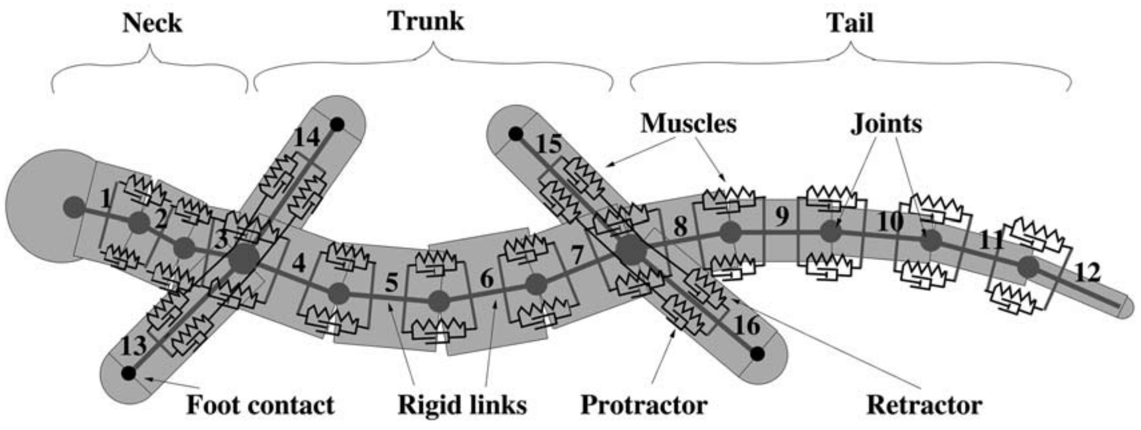


Figure 4: Mechanical model of a salamander, reproduced from [6].

This gets us to a total of five joints required to localize the trunk and belly pattern of the salamander. Based on this, we defined the belly pattern to be from the point in between the front legs to the point in between the back legs. We tried to label these five points equidistantly along the spine. The spine will then be estimated using bicubic interpolation [7] of these five points. The five points were defined as seen in Table 1 and Figure 5.

Table 1: Labeled points on salamander

Point	**Description**	**Color**
Front legs	Point between the front legs	Orange
Belly point 1	First intermediary point along the spine	Yellow
Belly point 2	Second intermediary point along the spine	Green
Belly point 3	Third intermediary point along the spine	Turquoise
Back legs	Point between the back legs	Blue

The simplest method to locate the belly pattern is by manual labeling. In currently available systems, a human labeler goes through the images and selects a set of points along the spinal cord. With a large dataset, this can take a large amount of time. Accurate manual labeling by the authors took about 30 seconds, which coincides with the findings of Matthé et al. [1]. To do this for a dataset with 10000 images, it requires more than 83 hours of manual labeling. Additionally, the human labeler is subject to bias and error rates may also increase when working on labeling for a prolonged amount of time.

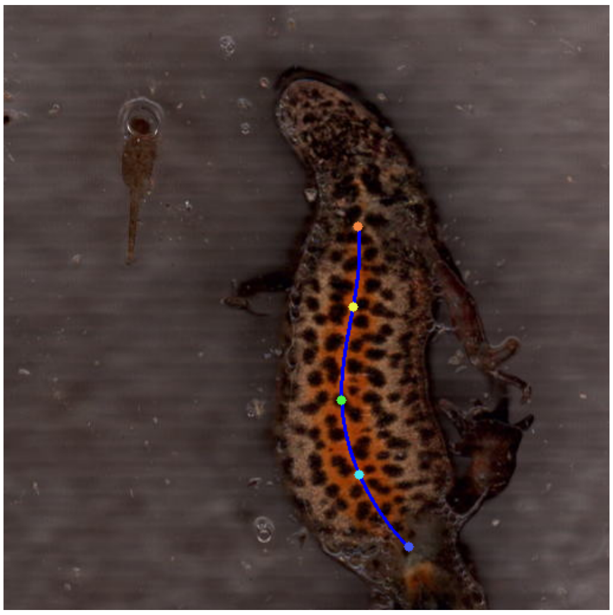


Figure 5: Example of a manually labeled salamander. The color points indicate the different joints of the salamander. The figure is best viewed in color.

Belly localization

Three different methods for belly localization have been tested and they are presented in the following paragraphs.

Segmentation and skeletonization: This method works by first segmenting out the salamander from the background. Further, it applies skeletonization, also referred to as a medial axis transform to estimate the salamander's spinal cord. This method is intended to be used by having user input making the initial line near the object you want to create a contour, however this method proved unsuitable for two reasons: (1) requires longer time to process (2) mixing of background information into the area of interest. After the salamander is segmented out a medial axis transform is performed [8]. This constructs the skeleton image. After the skeleton image is constructed it is converted to a skeleton network [9]. Additionally, we grouped vertices that are close to each other in proximity together to form the skeleton seen in Figure 6. Getting the belly line then simply consists of extracting the longest edge that is not connected to any leaf vertex.

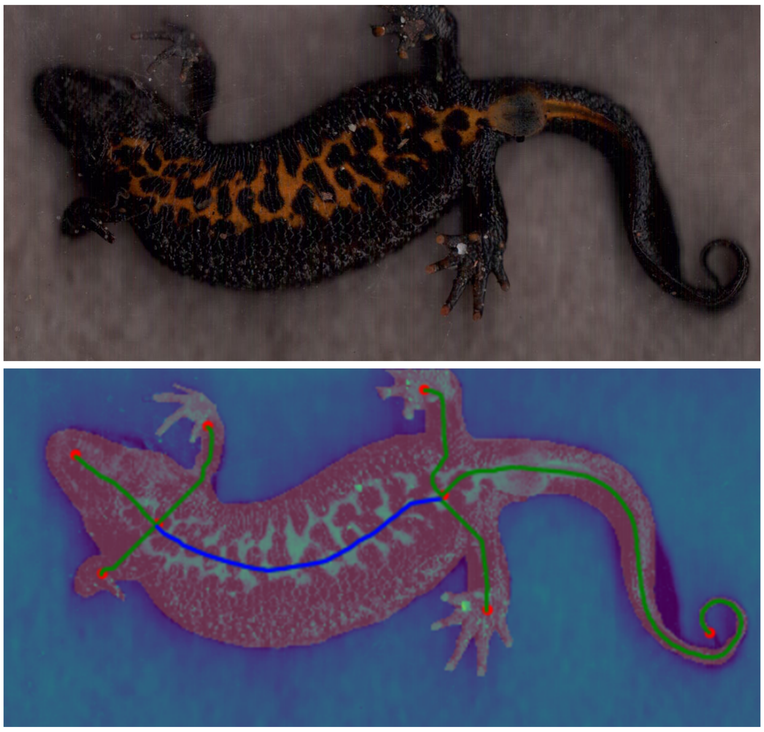


Figure 6: Salamander after manual segmentation and skeletonization.

The segmentation and skeletonization method has shown to quite error-prone due to a number of reasons. One problem with the method is that it requires the segmentation to be very accurate. If there are small errors in the segmentation, often caused by shadows or other artefacts in the image, this causes the skeletonization to create additional branches as seen in Figure 7. In this case a part of the tail, marked in blue, is actually erroneously identified as the belly pattern. If we would be able to attain perfect smooth segmentation as in the example of Figure 6, the same problem also occurs when the salamander does not have its feet perpendicular to body. This causes misalignment of the point between the feet which is used as the respective start- and end-point of the belly pattern. In Figure 6 we see that the salamander is bending and at the same time its's laying down further to one side. This causes the spinal cord and the belly pattern to not align with the medial axis of the segmented salamander. It also causes the pattern to differ depending on how the salamander poses on the image, which in turn negatively affects matching.

Pose estimation using LEAP: In order to more accurately extract the belly pattern, we tested a convolutional neural network (CNN) developed to estimate pose quickly called LEAP [10]. This is to be used in real-time applications. In order to make the images for network, images are converted to grayscale with a resolution of 196 x 196 or 256 x 256 as recommended [11]. Labeled images are split into a training set 90% and test

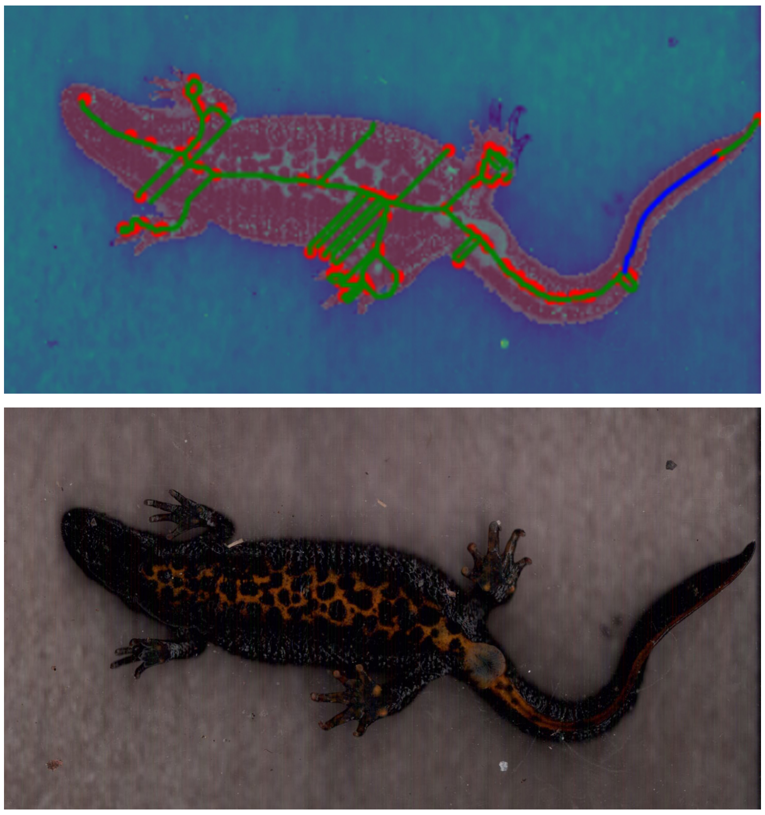


Figure 7: Consequences of minor segmentation errors. Original image on bottom, and segmented image on top. Best viewed in color.

set 10% with a batch size of 16, and rand 50 batches per epoch, for a total of 15 epochs. The dataset is augmented using random rotations (+-45) and mirroring. We have used 10% of the images in each batch for batch-level validation and used 10 of the 50 batches for epoch-level validation. The training finished in less than 5 minutes using hardware mentioned in Table 2.

Table 2: Hardware system used for testing in this work.

Component	Model
GPU	Geforce GTX 1060 6GB
CPU	Intel Core i5-2310@2.9 GHz
RAM	DDR3 8GB

The error is calculated by taking the Euclidean distance between the point predicted by LEAP and the human labeled points. The system was evaluated on 10 randomly selected images and is presented in Table 3.

It is observed that error rates are relatively high, with an average Euclidean error of 66px which is about the width of a typical salamander in our images. LEAP was able to generate useful results in only 1 out of our 10 testing images. It is able to predict the location of the belly pattern on some basic images, but fails quickly if there is rotation and/or bend in the images. Based on the results, we concluded that this neural network is not good enough for the present study.

Pose estimation using DeepLabCut: DeepLabCut [12, 13] is another tool for markerless pose estimation of user-defined body parts with deep learning. DeepLabCut

Table 3: LEAP testing results.

Metric	Error mean	Standard deviation
Front legs	106.9 px	103.8 px
Belly point 1	68.7 px	45.8 px
Belly point 2	37.2 px	26.3 px
Belly point 3	57.0 px	61.0 px
Back legs	65.4 px	96.4 px
All	**66.0 px**	**72.8 px**

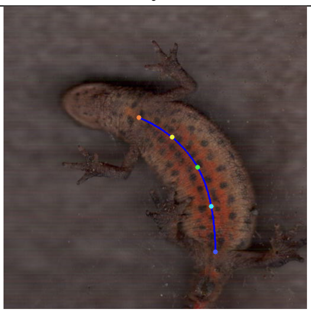
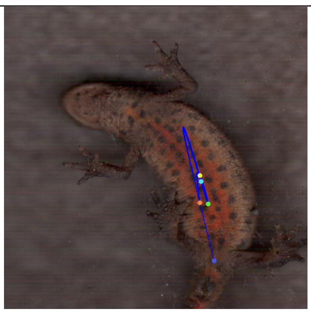
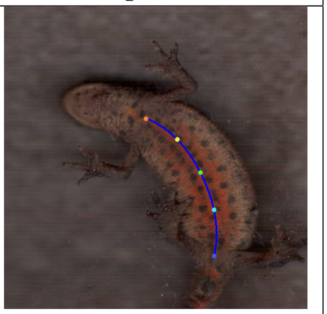
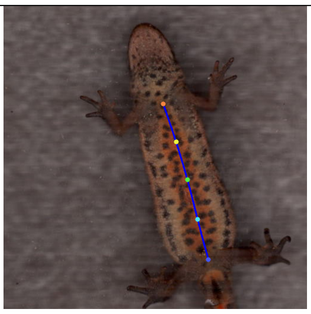
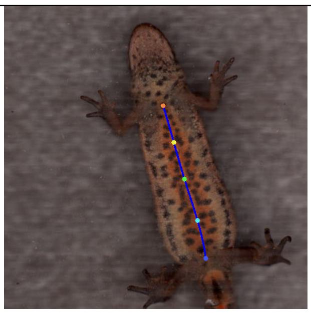
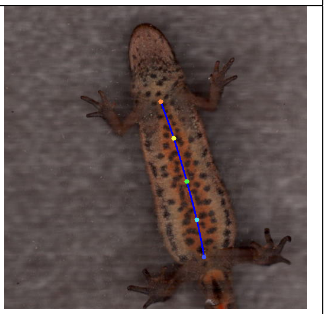
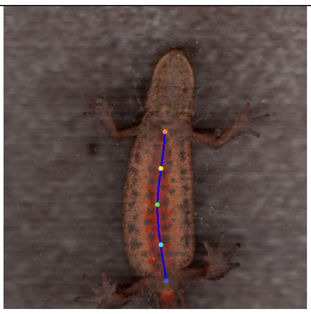
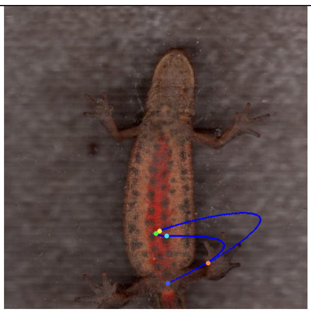
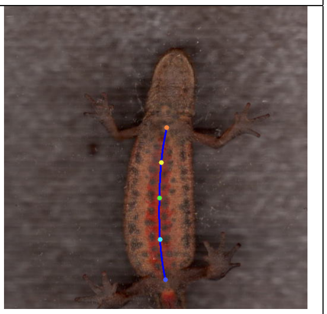
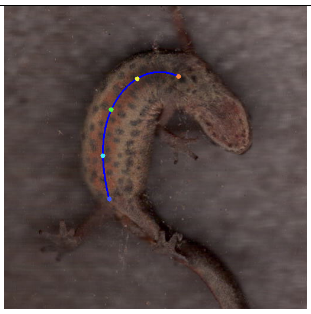
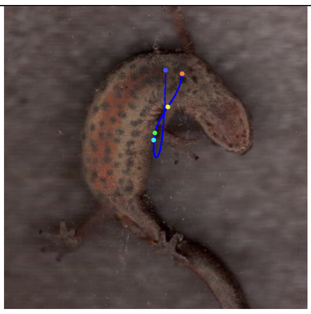
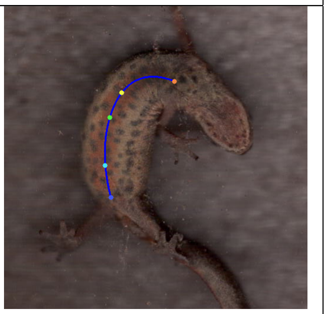
uses Huber loss as it's loss function and is less sensitive to outlier data than the mean-squared-error loss that is used in LEAP [10], as it is linear instead of quadratic for large errors. In order to test the accuracy of DeepLabCut, we provided it with the 200 labeled images. We provided the images in a 512 x 512 RGB format. The labeled images were split into 95% training and 5% testing sets. Data augmentation was done by mirroring the images, applying random rotations (+-30) and resizing (+-25%) the images and labels. Additionally, we used pretrained weights for the backbone, which is a deep residual network, ResNet [14], that has run 5000 iterations on the ImageNet [15] dataset, which according to it's authors makes it able to accurately predict pose with only 200 labeled images [12]. It is recommended to train the network for at least 200,000 iterations in order to arrive at a plateau of the loss function [13]. We trained the network for 350,000 iterations which took 12 hours using our hardware.

The error is calculated by taking the Euclidean distance between the point predicted by DeepLabCut and the human labeled points. The system was evaluated on 10 randomly selected images, and the results are shown in Table 4. DeepLabCut was able to predict images at a rate of 5.2 FPS.

Table 4: DeepLabCut testing results.

Metric	Error mean	Standard deviation
Front legs	6.4 px	2.9 px
Belly point 1	9.3 px	9.8 px
Belly point 2	10.5 px	5.8 px
Belly point 3	8.1 px	4.4 px
Back legs	3.3 px	1.9 px
All	**7.5 px**	**6.2 px**

Table 5: Comparison of belly localization methods.

#	Manually labeled	LEAP	DeepLabCut
1			
2			
3			
4			

Although DeepLabCut at 5.2 FPS is considerably slower than LEAP at 107.3 FPS, we also have a better accuracy with a significant margin for DeepLabCut. From the results in Table 4 there is a mean error of 7.5 px. We also observe that the endpoints have a considerably lower mean error than the three belly points. It is therefore reasonable to assume that the labeled points where not perfectly equidistantly aligned although they were relatively accurately placed on the spinal cord. This causes some fluctuations along the spinal cord. This does increase the mean error but does not affect the resulting bicubic interpolation as much. We could also validate this by simple visual inspection by comparing the bicubic interpolation of the manually selected points to the bicubic interpolation of the points selected by the neural network in Table 5.

As the input images were required to be a straight image, all the images are converted into a rectangular image, with the spinal cord of the salamander in the center. This

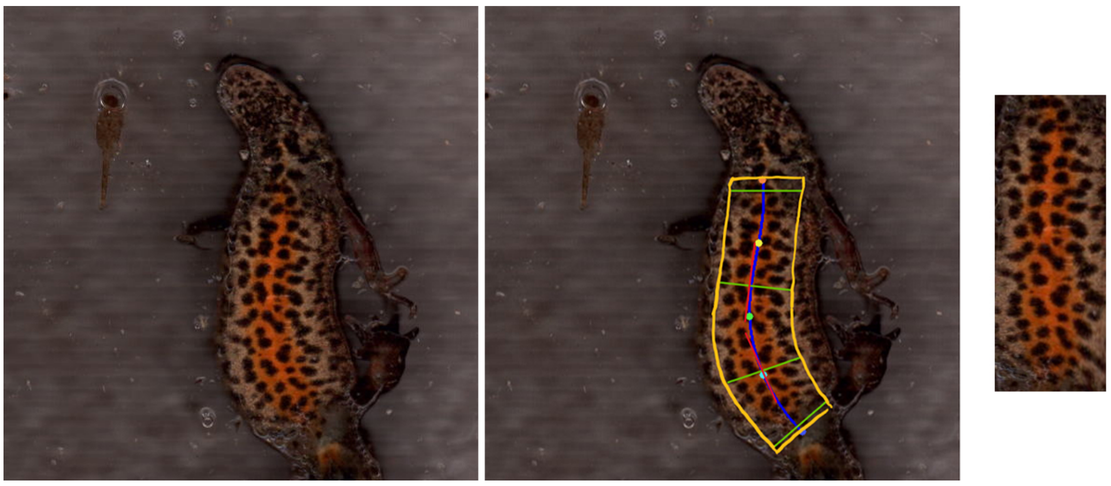


Figure 8: Straightening using DeepLabCut and the derivative of the interpolated spinal curve.

has been implemented using a nonlinear transformation based on the gradient of the interpolated curve. An example is shown in Figure 8.

Salamander recognition

Two different methods for recognition have been tested; a pixel based method and a method based on brute force matching.

Pixel based method

The method itself revolves around comparing the images pixel by pixel and looking at the difference to make an average pixel difference between the images to find matches. In order to make the images suitable for pixel-based comparison, it is necessary to perform few pre-processing steps: resizing and thresholding. Images are resized into 80 x 280 using bilinear interpolation. In addition to making the images the same size this also shrinks them by roughly 75% which makes the comparison algorithm faster.

To increase the speed of the algorithm the images are binarized. Doing this step does not cause any loss of information because only the difference between the images is the presence of a spot from the pattern or not. Compared to other thresholding methods, Otsu binarization [16] worked very well across the images we tested. Images in the present study are bimodal, which makes it easy for the algorithm to calculate a good threshold value. In order to reduce the noise after binarization, the images were blurred to smooth out the noise without compromising the pattern quality in the salamander body. Out of different blurring methods tested, median blur removed the most noise and resulted in least effect on the details of the pattern.

The average pixel difference adding up the differences of all the pixels and dividing by the dimensions of the image. One drawback with this method is that even a minor shift in the animal and thus resulting pattern, the images will not match and can drastically increase the difference between the images. To solve this careful image translation in each direction is performed and followed by comparison algorithm over and over again and picks the best score. In the present experiment, we have shifted images 20 pixels left and right with a step size of 2 and up and down 10 pixels and picking the best result.

This method showed a decent performance to find matches for ten images from a pool of another ten images consisting of of pairs of the original ten. While testing for bigger pool trying to find the same 10 matches in a pool of 48 images the best match of each image was still a real match, however when we looked at all the match numbers there were plenty of images that gave potential matches meaning that if there had been no match in the system it would have given a false match as a result. These false positives varied in severity and we consider increasing the threshold in order to filter more of them out. However, some of the false positives reached as high as 92% and changing the threshold that high would result in most of the true matches as not good enough.

Brute force matching

This is a very simple but fast matching method. It takes two set of data, in this case two descriptors. The function will use the first descriptor as a "template" and then try to find the closest one in the second set and result in an image the closest to the template. In these two objects there's something called distance, and this will tell us the distance between the two matched points. If the distance parameter is within a certain number (in the present case 0.75, this number is commonly used for ratio tests) it's a match. In this work we set a minimum number of matches in order to make sure it's a match. This is because of the images are very similar so it's possible to get a match even though the two descriptors are from two different animals. We tested many different numbers and with some bad images we will get a low number of good matches. Since the lowest number in the present test set was 15 and is been chosen to use this as a minimum. This implies the two descriptors need at least 15 matches before we can be sure the pattern belongs to the same individual salamander.

We tested four popular feature based methods, ORB (Oriented FAST and rotated BRIEF) [17], SURF (Speeded up robust features) [18], SIFT (Scale-invariant feature transform) [19] and DSIFT (Dense SIFT). DSIFT showed the best performance, and the results for this is shown.

Evaluation

Table 6 shows the results obtained using a brute force matching with DSIFT and the pixel based method. As shown we have achieved a high accuracy with brute force matching and DSIFT.

Table 6: Match rates on DSIFT and pixel based method.

	Correct result	Match not found	False positive	Potential false positives
DSIFT	45	4	0	0
%	91.84%	8.16%	0.00%	
Pixel	21	9	19	511
%	42.86%	18.37%	38.78%	

We have analysed the results particularly the failed cases. In some cases, the belly pattern is occluded by the salamander's feet, and this led to the failure of pose estimation by DeepLabCut. Other reasons for false detection are the poor image quality mainly by the low lighting when the images were taken.

The main contribution of our system, contrary to existing software, is that the system provides a fully automated workflow as shown in Table 7.

Table 7: Comparison between existing software and our solution.

Software	Matching	Straightening	Licensing
Aphis	Pixel-based method and SURF[18]	No	Open Source
AmphIdent	Pixel-based method	Manual	Commercial
Wild-ID	SIFT [19]	No	Open Source
I3S Pattern	SURF [18]	Manual	Open Source
Proposed	**DSIFT**	**Automatic**	**Open Source**

4 Conclusion and future work

The goal of this work was to propose a solution which identifies salamanders based on pattern recognition with a high accuracy comparable to existing software's and to reduce the amount of manual work required to match salamanders by automating the workflow. We have reviewed the existing methods and software solutions available and outlined their differences, individual advantages and disadvantages. Based on the state-of-the-art methods and techniques used, we have identified an optimal solution with regard to the research objective. On the basis of this, a set of methods were selected for implementation and evaluated for this project. The methods selected were the neural network DeepLabCut for pose estimation and Dense SIFT for pattern recognition. The selected methods were later implemented as software solution. One main contribution of the work is a system that uses a matching method with a high accuracy on the current dataset. To the best of our knowledge, no efforts has produced any similar results which is been to the direction of automated solution for salamander recognition.

Recent advancements in deep neural networks allowed us to create this automated solution. With improved cameras, the images capture in future would have better image quality and expect increase in performance of the matching. NINA is already moving ahead in this direction. It is possible to extend the proposed methods for performing recognition tasks in other animals.

References

[1] Matthé, M., Sannolo, M., Winiarski, K., Sluijs, A. S. V. D., Goedbloed, D., Steinfartz, S., & Stachow, U. 2017. Comparison of photo-matching algorithms commonly used for photographic capture-recapture studies. *Ecology and Evolution*, 7(15), 5861–5872. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5552938/, doi:10.1002/ece3.3140.

[2] Management, R. URL: http://www.reijns.com/i3s/download/I3S_download.html.

[3] Bolger, D. T. URL: https://home.dartmouth.edu/faculty-directory/douglas-thomas-bolger.

[4] Amphident. URL: http://www.amphident.de/en/pages/download.html.

[5] d'Estudis Avançats, I. M. URL: http://imedea.uib-csic.es/bc/ecopob/index.php?option=com_content&view=article&id=73&Itemid=89&lang=en.

[6] Ijspeert, A. 06 2001. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological cybernetics*, 84, 331–48. doi:10.1007/s004220000211.

[7] scipy.interpolate.interp1d.

[8] Scikit image documentation morphology.skeletonize. Available at https://scikit-image.org/docs/0.15.x/api/skimage.morphology.html#skimage.morphology.skeletonize, Accessed 08:58, 12 May 2019.

[9] Build network from skeleton image (2d-3d). Available at https://github.com/yxdragon/sknw, Accessed 09:05, 12 May 2019.

[10] Pereira, T., Aldarondo, D., Willmore, L., Kislin, M., Wang, S., Murthy, M., & Shaevitz, J. W. 2018. Fast animal pose estimation using deep neural networks. *bioRxiv*. URL: https://www.biorxiv.org/content/early/2018/05/25/331181, arXiv:https://www.biorxiv.org/content/early/2018/05/25/331181.full.pdf, doi:10.1101/331181.

[11] Opencv documentation - cv2.resize.

[12] Mathis, A., Mamidanna, P., Abe, T., Cury, K. M., Murthy, V. N., Mathis, M. W., & Bethge, M. 2018. Markerless tracking of user-defined features with deep learning. *arXiv preprint arXiv:1804.03142*.

[13] Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. 2018. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *bioRxiv*. URL: https://www.biorxiv.org/content/early/2018/11/24/476531, arXiv:https://www.biorxiv.org/content/early/2018/11/24/476531.full.pdf, doi:10.1101/476531.

[14] He, K., Zhang, X., Ren, S., & Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

[15] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

[16] Otsu, N. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1), 62–66.

[17] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. Nov 2011. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, 2564–2571. doi:10.1109/ICCV.2011.6126544.

[18] Bay, H., Tuytelaars, T., & Van Gool, L. 2006. Surf: Speeded up robust features. In *European conference on computer vision*, 404–417. Springer.

[19] Lowe, D. G. Sep. 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, 1150–1157. doi:10.1109/ICCV.1999.790410.