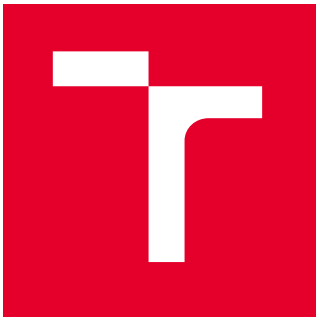


BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ÚSTAV AUTOMATIZACE A INFORMATIKY

COMPENSATION OF DISTORTIONS CAUSED BY MOVEMENTS OF OBJECTS SCANNED BY A LINE SCAN CAMERA

KOMPENZACE ZKRESLENÍ VZNIKLÝCH POHYBEM OBJEKTŮ SNÍMANÝCH ŘÁDKOVOU KAMEROU

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Michal Szabó**

AUTOR PRÁCE

SUPERVISOR — **Ing. Pavel Škrabánek, Ph.D.**

VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Master's Thesis Assignment

Institut:	Institute of Automation and Computer Science
Student:	**Bc. Michal Szabó**
Degree programm:	Applied Computer Science and Control
Branch:	no specialisation
Supervisor:	**Ing. Pavel Škrabánek, Ph.D.**
Academic year:	2022/23

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Compensation of distortions caused by movements of objects scanned by a line scan camera

Brief description:

Two basic types of image sensors used in industrial cameras are area and line scan sensors. To get a digital image of a rectangular section of a scene, an area scan sensor scans a section of a projection of the scene at once. A line scan sensor, however, must scan the projection line-by-line to get the image of the section. The line-by-line scanning requires movement either of a camera (or of its part) or of the scene. If an unwanted movement of the camera or of the scene occurs during the line-by-line scanning, a distortion appears in the image. Such distortions are usually unwanted, and it is desirable to suppress them to the highest possible extent.

Master's Thesis goals:

The student will study the issue of line scan cameras and of digital image restoration, with a special emphasis on compensation of motion distortions. The student will use the gained knowledge to develop and implement a method aimed at compensation of distortions in images captured using line scan cameras, where the distortions are caused by unwanted movements of a camera or of the scene. The unwanted movements are expected to be approximately perpendicular to the direction of the scanning. The student will also propose a methodology for evaluating the quality of the image restoration and will apply the methodology to the proposed method. If necessary, the student will create a dataset on which the development and evaluation of the method will be carried out.

Recommended bibliography:

SU, Jingwen, Boyan XU a Hujun YIN. A survey of deep learning approaches to image restoration. Neurocomputing [online]. 2022, 487, 46-65 [cit. 2022-09-05]. ISSN 09252312. Dostupné z: doi:10.1016/j.neucom.2022.02.046.

BATCHELOR, Bruce G., ed. Machine Vision Handbook [online]. London: Springer London, 2012 [cit. 2022-09-05]. ISBN 978-1-84996-168-4. Dostupné z: doi:10.1007/978-1-84996-169-1.

Students are required to submit the thesis within the deadlines stated in the schedule of the academic year.

In Brno, 18. 10. 2022

L. S.

doc. Ing. Radomil Matoušek, Ph.D.	doc. Ing. Jiří Hlinka, Ph.D.
Director of the Institute	FME dean

ABSTRACT

This master's thesis focuses on compensating for distortions in data of lizards obtained via a line scan camera (LSC) that captures ultraviolet (UV) wavelengths. Breathing movements during scanning cause distortions in the lizard's trunk width, affecting the final LSC image. The thesis proposes image processing methods to adjust these distortions, including contour extraction, interpolation, and evaluation using a reference image. The methodology aims to minimize the difference between the adjusted LSC image and the reference image.

ABSTRAKT

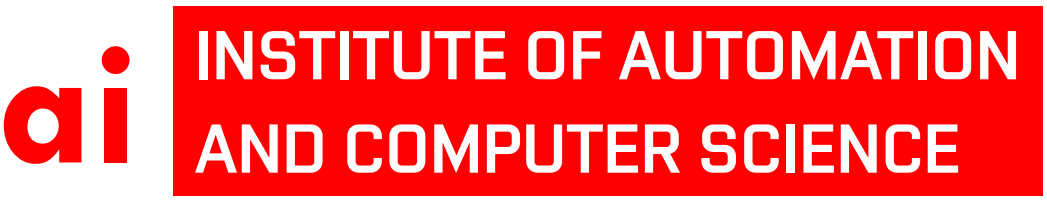
Tato diplomová práce se zaměřuje na kompenzaci zkreslení v datech ještěrek získaných pomocí řádkové skenovací kamery (LSC), která snímá ultrafialové (UV) vlnové délky. Dýchací pohyby během skenování způsobují zkreslení šířky trupu ještěrky, což ovlivňuje konečný snímek LSC. Práce navrhuje metody zpracování obrazu pro úpravu těchto zkreslení, včetně extrakce kontur, interpolace a vyhodnocení pomocí referenčního snímku. Metodika má za cíl minimalizovat rozdíl mezi upraveným LSC snímkem a referenčním snímkem.

KEYWORDS

Line scan camera, distortions, compensation, image processing, contour extraction, interpolation, evaluation, reference

KĽÚČOVÉ SLOVÁ

Řádková skenovací kamera, zkreslení, kompenzace, zpracování obrazu, extrakce kontur, interpolace, vyhodnocení, reference



2023

BIBLIOGRAPHIC CITATION

SZABÓ, Michal. *Compensation of distortions caused by movements of objects scanned by a line scan camera*. Brno, 2023. Available at: `https://www.vut.cz/studenti/zav-prace/detail/149175`. Master's Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Supervised by Ing. Pavel Škrabáne, Ph.D.

AUTHOR'S DECLARATION

I honestly declare that this master's thesis on the topic "Compensation of distortions caused by movements of the objects scanned by a line scan camera" was developed independently by me under the guidance of Ing. Pavel Škrabánek, Ph.D. and that I have listed all the sources and literature used.

As the author of this work, I declare that I have not violated any third-party copyrights or infringed upon them illegally while creating this work. I am fully aware of the consequences that may arise from violating § 11 and the following sections of Copyright Act No. 121/2000 Coll., which could potentially lead to criminal charges.

In Brno on 21. 5. 2023 .....

Michal Szabó

ACKNOWLEDGEMENT

I would like to thank Mr. Ing. Pavel Škrabánek, Ph.D, for his valuable advice and guidance during the completion of this master's thesis.

I would also like to extend my thanks to my family for their continuous encouragement and support throughout my academic journey. Additionally, I am grateful to my classmates and friends for their assistance, the enjoyable memories we shared, and everything else that occurred during my university studies.

CONTENTS

1 Introduction

Ultraviolet (UV) vision and UV color patches have been reported in a wide range of animal species and are increasingly appreciated as an integral part of vertebrate visual perception and communication systems. Previous studies with Lacertidae, a lizard family with diverse and complex coloration, have revealed the existence of UV-reflecting patches that may function as social signals [1]. New studies are currently in process on a various species of lizards, but in order to study them, it is necessary to have a good data for analysis.

This master's thesis aims to address the issue of compensating for distortions in collected data resulting from the movements of lizards scanned by a line scan camera (further only as LSC). The problem arises due to the selection of a line scan hyperspectral camera for capturing the reflection of the lizard in ultraviolet wavelengths. This type of camera is preferred because of its high spectral resolution, which a regular snapshot camera cannot achieve due to its multiple dispersive elements that separate incoming light into different wavelengths.

The scanning process with the LSC involves placing the living object on a moving table located under the hyperspectral camera. The table's speed is synchronized with the camera's frame rate, and another camera captures a snapshot of the scene for reference purposes. After scanning, the camera outputs an image that is combined from each scanned row. As the object being scanned is a lizard, it can move during the scanning process, resulting in various image distortions depending on the type of movement. This thesis specifically deals with compensating for breathing movement, which causes a change in the lizard's trunk width during scanning, resulting in saw-like edges in the final hyperspectral image.

Various image processing methods are used to extract information about the lizard's body contours and to adjust these distortions. Multiple algorithms are performed on these contours to locate the area that needs adjustment. The image is adjusted through interpolation by calculating the natural contours of the lizard's trunk area and then interpolating the degenerated body into a naturally-looking one.

After the interpolation, the compensation is evaluated. The user selects a reference image where the lizard looks natural. The evaluation method involves calculating the difference between the original image contours from LSC and reference image contours. The same difference is then calculated with the adjusted image, and the evaluation result is presented as a percentage, describing the degree to which the difference was minimized in comparison to the original image.

2 Problem Analysis

To analyze the UV color patches, the image of a lizard needs to be taken in the UV wavelength range. The traditional color camera will not be sufficient for this, as it can only get the reflection of the object in three bands that are visible for the human eye (red, green and blue).

Spectral imaging divides the spectrum into many more bands. This technique of dividing images into bands can be extended beyond the human visible range. A type of camera that uses this technique for capturing light across a wide range of wavelengths, from ultraviolet to near-infrared is called hyperspectral camera.

2.1 Hyperspectral Cameras

Hyperspectral cameras work by using a spectrometer, which splits the incoming light into spectral bands and measures the intensity of each band. The dispersed light falls onto the charge-coupled device (CCD) sensor array, where each pixel detects a specific spectral band. The number of pixels in the array determines the spatial resolution of the resulting image, while the number of spectral bands captured determines the spectral resolution. The data from the hyperspectral camera are formed into a hyperspectral image, which is a three-dimensional data cube where each pixel contains information about the intensity of light across hundreds of different wavelengths. This data can be analyzed to identify materials or substances based on their unique spectral signatures.

There are several types of hyperspectral cameras, each designed for specific applications. The most common types of hyperspectral cameras are (Fig. 1):

- Whiskbroom (Point scan) hyperspectral camera: This type of hyperspectral camera uses a scanning mirror to sweep across the scene and capture data in a series of narrow strips. This type of camera is often used in laboratory settings.
- Push-broom (Line scan) hyperspectral camera: This type of hyperspectral camera captures an image line by line as it moves across a scene. The camera is typically mounted on a platform that moves over the area being imaged.
- Imaging spectrograph (Spectral Scan): This type of hyperspectral camera uses a dispersive element to separate the incoming light into its different wavelengths, which are then focused onto a detector array. The imaging spectrograph can be configured for different spectral resolutions and bandwidths.
- Snapshot hyperspectral camera: This type of hyperspectral camera captures all spectral bands simultaneously in a single snapshot. It can be used for

real-time imaging applications such as medical imaging and industrial quality control.

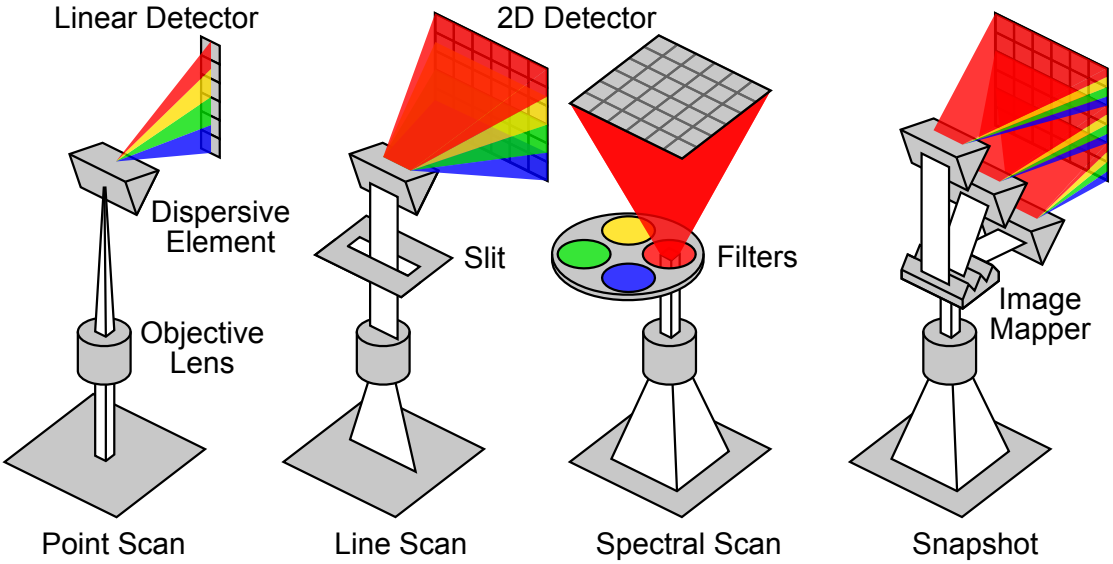


Fig. 1: Types of acquisitions of a multi-/hyperspectral imaging systems [2].

Each type of hyperspectral camera has its own set of advantages and disadvantages depending on the specific application. In this case, data were collected using a LSC due to its high spectral resolution and adequate scanning area.

2.2 Technical Equipment

This section provides a brief description of the technical equipment used to collect the data. It should be noted that the selection of technical equipment was not a part of this master's thesis.

2.2.1 Hyperspectral Camera Pika NUV2

The Pika UV (formerly Pika NUV2) is a line-scan hyperspectral camera that covers the near ultraviolet and visible spectral range (330 – 800 nm). It can be used with Resonon's Reflectance benchtop system, outdoor, and airborne systems, standalone with Resonon's software development kit, and integrated into machine vision systems [3]. The Pika NUV2 camera is displayed in Fig. 2.

FEATURES:

- Spectral Range: 330 – 800 nm
- 1500 Spatial Pixels Per Line
- 255 Spectral Channels Per Line

Fig. 2: Hyperspectral camera Pika NUV2 [3].

2.2.2 Reference Camera Basler acA2500-60um

The Basler acA2500-60um USB 3.0 camera with the ON Semiconductor PYTHON 5000 CMOS sensor, that delivers 60 frames per second at 5 MP resolution. This camera was used to take reference images during the scanning of the lizard, for comparison purposes [4]. The Basler camera is displayed in Fig. 3.

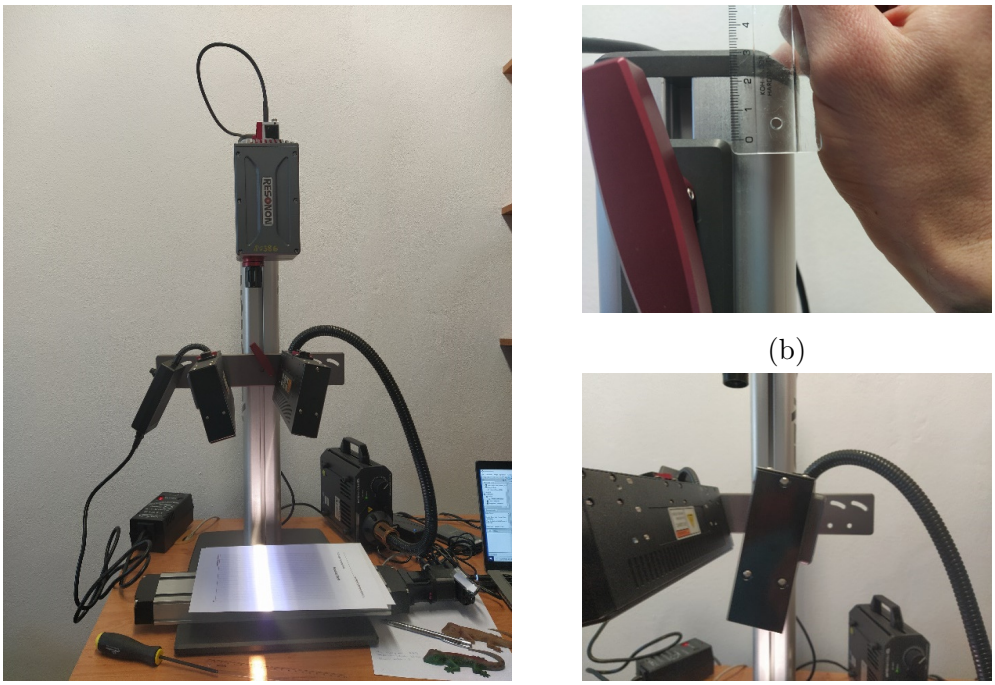
Fig. 3: Basler acA2500-60um camera [4].

2.2.3 Lens Kowa LM12SC

This is a 1" C-mount lens with a 16 millimeter image circle, built with a metal body and mount. It has a high resolution of 6 megapixels and features an adjustment screw for aperture and focus [5]. The lens was mounted on a Basler camera (2.2.2) and adjusted to ensure good image quality.

2.3 Data Collection Setup

The hyperspectral line-scan camera was mounted on the Resonon's desktop system (Fig. 4 (a)) approximately 3,2 cm from the top of the camera stand (Fig. 4 (b)). The NUV lights are positioned as close to each other as possible, mounted at the first position on a holder, with the UV light located on the right side (Fig. 4 (c)).



(a) (b) (c)

Fig. 4: Camera setup (a) with camera position detail (b) and UV lights positions (c).

The angle of the lights was adjusted individually in live camera mode by a line test, that was placed on the moving board. The width of the visible light beam was set to approximately 2.3 cm using a rotating knob, while the width of the UV light remained unchanged and was aligned symmetrically with the visible light.

In order to maintain optimal image clarity throughout the entire height of the scanned object, the camera's focus was adjusted to the middle height of the lizard. The camera was also calibrated to reduce dark noise from the sensor and to account for the current intensity of light radiation. The following settings were configured for the hyperspectral line-scan camera:

- Frame rate was set to 27 fps.
- Integration time was set to 19,6 ms.
- Gain was set to 13.

The speed of the moving board, or scanning speed, was set to approximately 0.3209 cm/s. The accuracy of the speed was verified through a circle test.

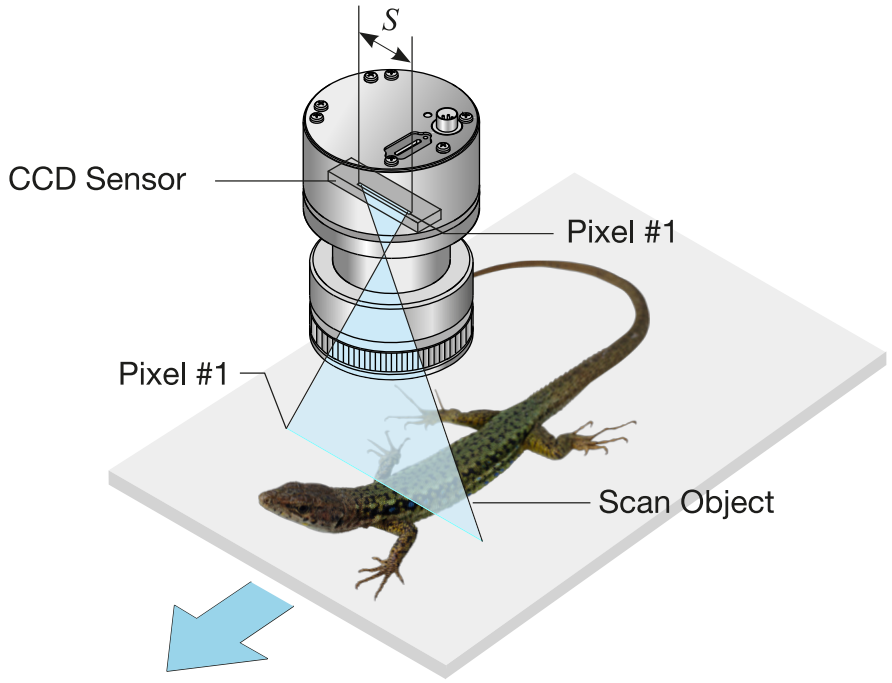


Fig. 5: Data collection scheme with a line scan camera [6] and a lizard [7] positioned on the moving board.

2.4 Problem Description

The process of scanning a living object can result in various image distortions, which depend on the movement of the object being scanned. Fig. 7 shows some examples of color images constructed from hyperspectral images. If the scanning is not interrupted, the resolution of these images is 1100x1500 pixels.

The data were collected in April 2022, when the lizards were coming out of winter hibernation and their metabolism was slowed down as a result. Due to this fact, the lizards moved less, and distortions in the data were minimized. Nevertheless, some lizards were still active and had to be restrained during the scanning process.

Fig. 6 showcases four types of distortion that can occur during the scanning process. The first type occurs when an object obstructs the camera's view, as shown in Fig. 6 (a). The second type results from the object's movements, particularly in the legs and tail. Another distortion arises from breathing movements in the lizard's trunk area, causing saw-like edges along the body. Moreover, the lizard's twisting movements can also cause distortion, as illustrated in Fig. 6 (d). Additionally,

improper speed settings of the scanning table can result in longer and thinner lizards in the images than in real life. Collisions with the camera or table can also create discontinuities in the captured images, as observed in Fig. 7 (b).

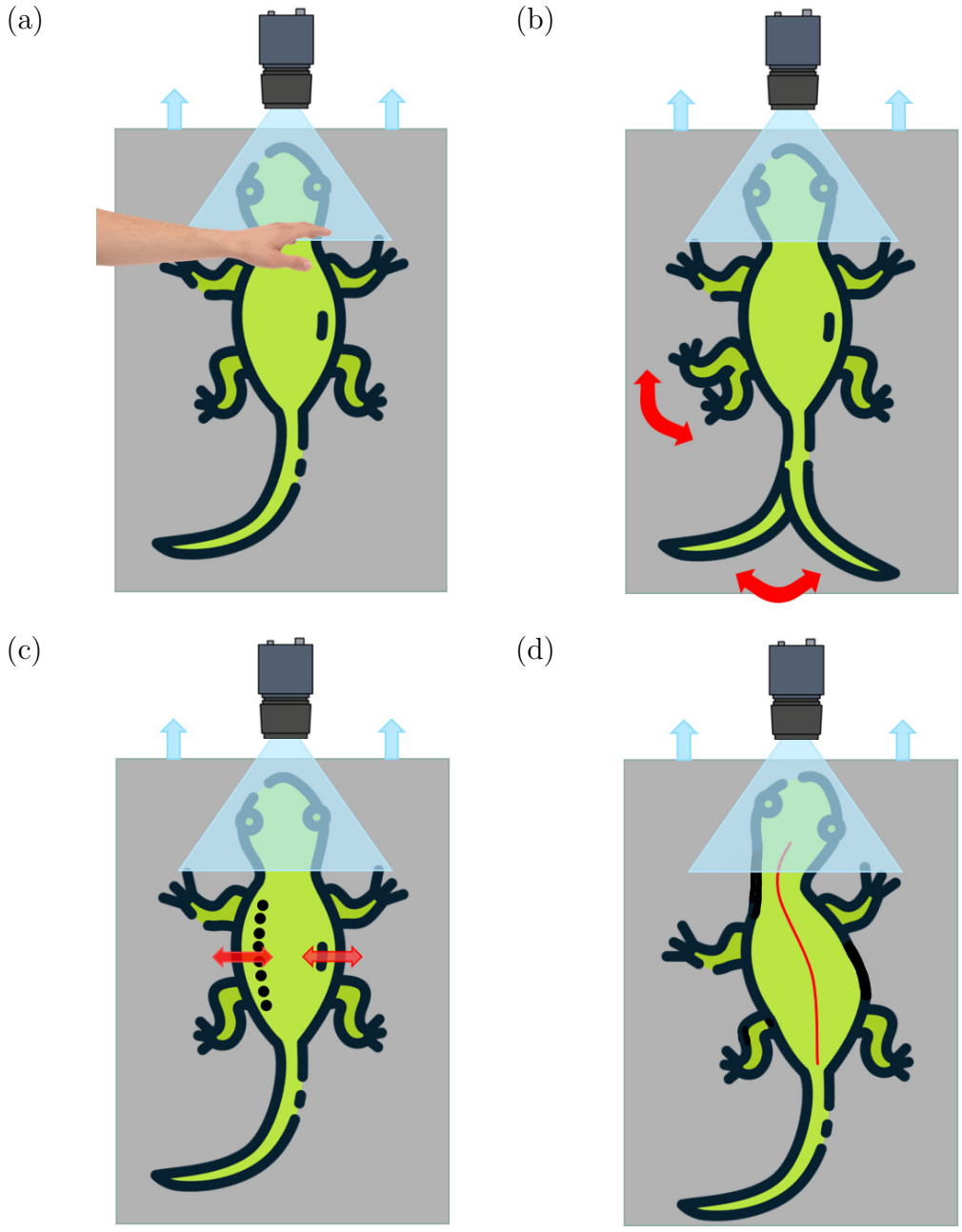
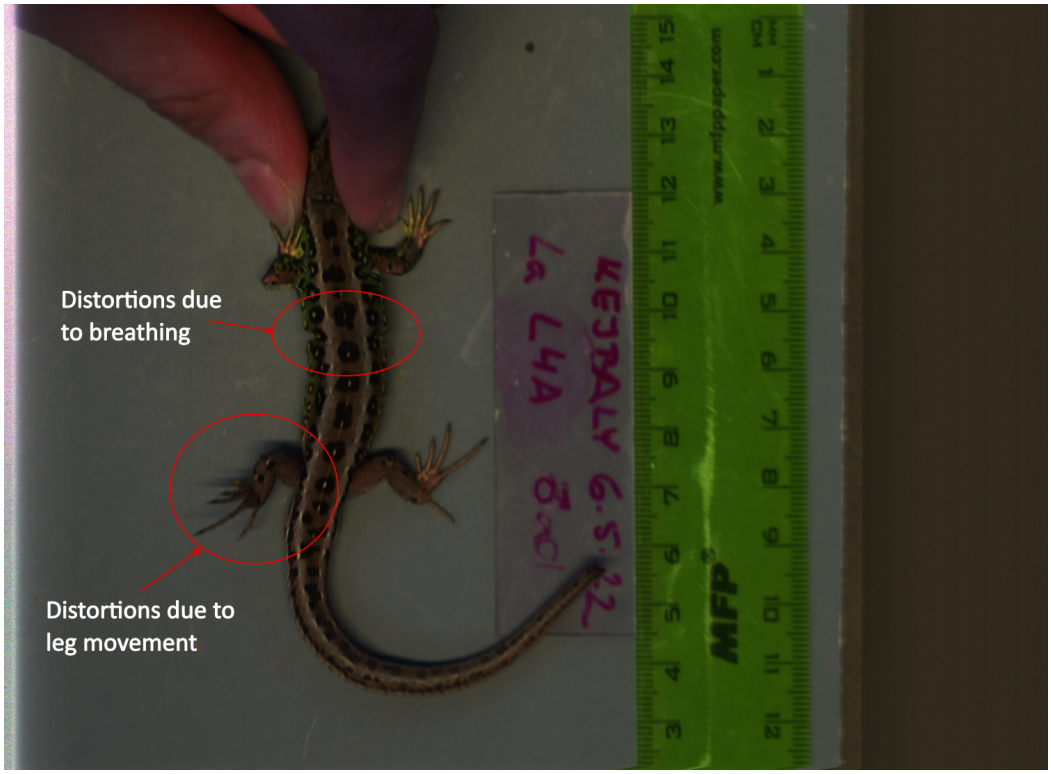


Fig. 6: Types of distortions: (a) object in camera's view, (b) leg and tail movements, (c) breathing movement and (d) twisting movement of the lizard [8].

Lighting can also cause image distortions. The images were captured outside, with a significant amount of ambient lighting creating uneven shadows on each side of the lizard. Furthermore, the ambient lighting conditions may have changed over time, resulting in different lighting conditions for each image.

(a)



(b)

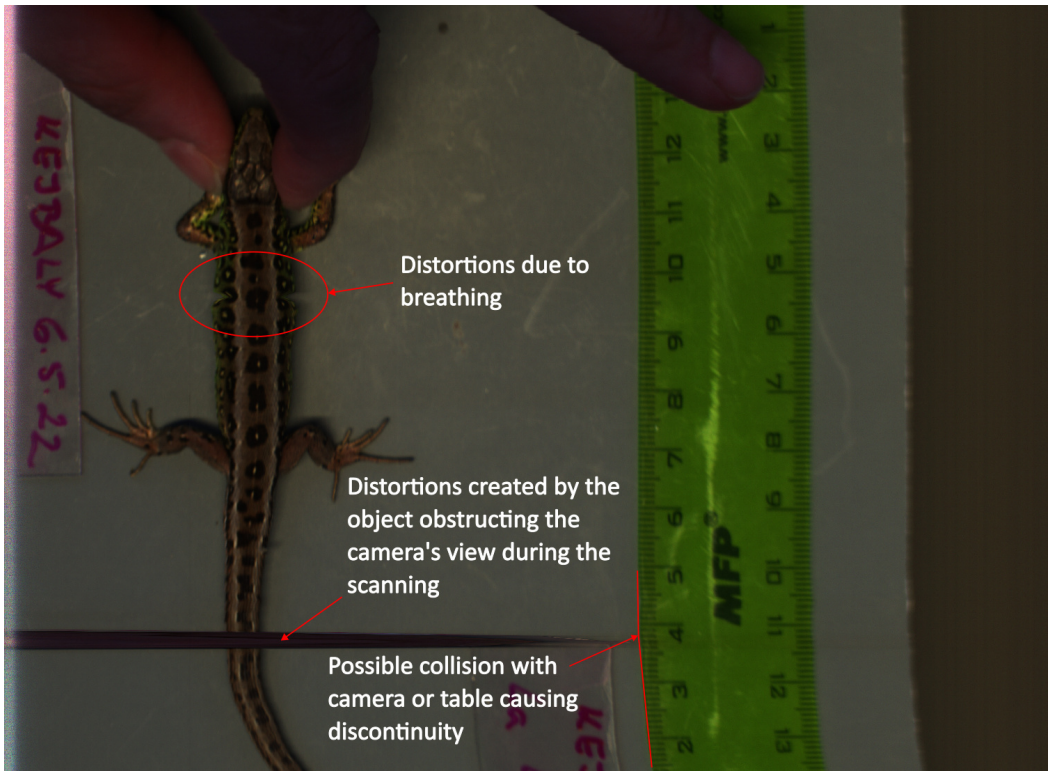


Fig. 7: Full resolution LSC images in RGB with names (a) L4ADORSAL3-RGB and (b) L4ADORSAL1-RGB.

2.5 Reference Images

The reference images taken by a reference camera during the scanning are capturing the whole lizard's body. These images can be used to explain lizard's respiration cycle and for the evaluation purposes. The reference camera was positioned as close to hyperspectral camera lens as possible. An example of full size (2048x2592) reference image is in Fig. 8. Detailed look on how the lizard's body changed during the respiration cycle is displayed in Fig. 9.

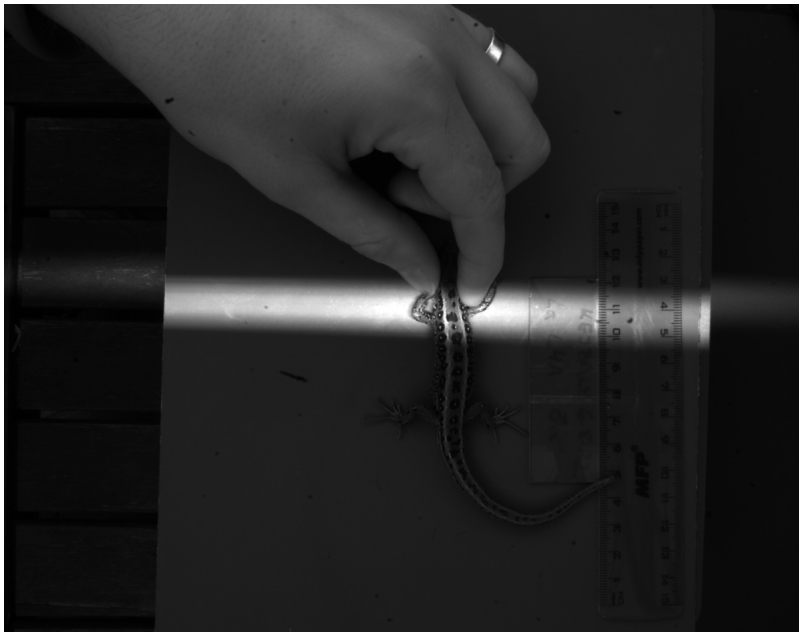


Fig. 8: Example of the full-size reference image.

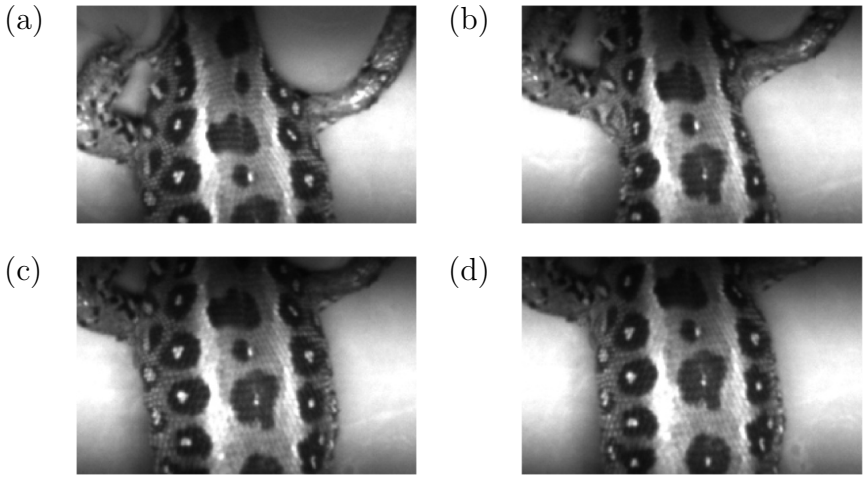


Fig. 9: Detailed images of the lizard's respiration cycle with states of (a) first lizard's inhale, (b) large exhale, (c) large inhale, and (d) in normal resting state.

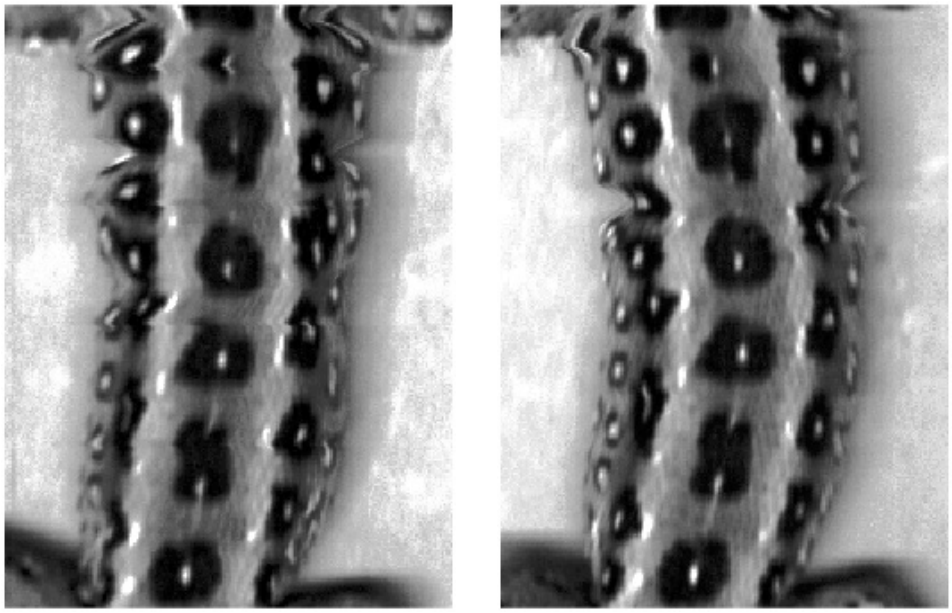
There are around 500-900 images for each scanning run. The frame rate of the reference camera was not exactly constant as there are significant differences in change of the lizard's position in between the frames.

2.6 Focus of Thesis

The main objective of this thesis is to investigate how the trunk area of a lizard can be adjusted to compensate for breathing movement. Specifically, the focus will be on the mechanisms by which the lizard can modify its trunk area with the aim to maintain a consistent volume of air in its lungs during respiration.

It's worth noting that the compensations for body or leg movements will not be addressed. Focus will be solely on the trunk area and its relationship to respiration. Furthermore, changes in light conditions will not be considered as a factor in this study.

The area of the lizard that will be modified is between the front and back legs of the lizard, displayed in Fig. 10.



(a) (b)

Fig. 10: Detailed view of trunk deformation in equalized grayscale images. The lizard in (a) corresponds to the LSC image named L4ADORSAL2, and the lizard in (b) corresponds to LSC image named L4ADORSAL3.

3 Image Processing Methods

Image processing involves converting an image into a digital format and then applying various operations to extract valuable information from it. Typically, image processing systems treat images as 2D signals and apply predetermined signal processing methods to analyze and manipulate them. This allows for the extraction of meaningful data from the images.

This section contains all the methods that were used to adjust distortions in lizard's images. Each method is briefly explained, along with a description of how it can be applied and what is it used for.

3.1 Binarization Methods

Image binarization, also known as image thresholding, is a technique used to convert a grayscale or color image into a binary image where each pixel is assigned either a foreground (object) or background (non-object) value based on a certain threshold value. In the resulting binary image, foreground pixels are typically represented as white (or 1) and background pixels as black (or 0).

Image binarization serves various purposes in image processing and computer vision. In this thesis it was used as a preliminary step for image segmentation. By converting an image to binary form, it simplifies the subsequent analysis and processing by separating objects of interest from the background. This is particularly useful in tasks such as object detection, object recognition, and image-based measurements.

Binarization techniques are generally categorized as Global and Local, where Global applies a single threshold to the entire image, but may fail for complex backgrounds, while Local chooses different threshold values for each pixel based on its neighborhood, but may not perform well with background noise. Thresholding methods can also be based on criteria such as histogram, clustering, entropy, and local adaptive methods, or use decision trees, combinations of techniques, or iterative methods [9]. An example of good binarization is displazed in Fig. 11.

Fig. 11: Example of good binarization on degraded sample image.[9]

3.2 Histogram Equalization

Histogram equalization is a technique used to enhance the contrast and improve the overall appearance of an image. It redistributes the pixel intensities in an image to utilize the full available range of values, thereby stretching the histogram and increasing the distinguishability of different intensities.

The `histeq` MATLAB function [10] is used in this thesis, to perform histogram equalization on an image. The function performs the following steps:

1. Compute the histogram: Calculate the histogram of the input image, which represents the frequency of occurrence of each pixel intensity level.
2. Compute the cumulative distribution function (CDF): Calculate the cumulative sum of the histogram values.
3. Normalize the CDF: Divide each value in the CDF by the total number of pixels in the image to obtain a normalized CDF.
4. Create the intensity mapping function: Multiply the normalized CDF by the maximum intensity value (e.g., 255 for an 8-bit image) to obtain the intensity mapping function. This function maps the original pixel intensities to new intensity values.
5. Apply the intensity mapping function: Map each pixel in the original image to its corresponding new intensity value using the intensity mapping function.

The `histeq` function returns the output image, which is the input image with histogram equalization applied. The resulting image will have enhanced contrast and a more uniformly distributed histogram, utilizing the full intensity range.

3.3 Image Filling

Image filling refers to the process of filling in missing or incomplete parts of an image to create a visually complete representation. This technique is often used in image editing and computer vision applications to repair damaged or obscured areas, remove unwanted objects, or generate realistic content.

The MATLAB `imfill` function is used in this thesis, to fill holes or gaps in binary or grayscale images. The function uses an algorithm based on morphological reconstruction [11], which operates by filling regions of an image based on connectivity and the specified filling criteria.

The connectivity parameter specifies the neighborhood connectivity used for determining which pixels should be considered as connected. It can take a value of 4 or 8, representing 4-connectivity and 8-connectivity, respectively. In 4-connectivity, pixels that share an edge are considered connected, while in 8-connectivity, pixels that share an edge or corner are considered connected.

The filling criteria parameter determines the criterion for filling the regions. It can take two main values: 'holes' or 'objects'.

- 'holes' is used to fill interior holes within objects. These holes are regions of background enclosed by foreground pixels.
- 'objects' is used to fill the exterior of objects. It fills regions of background that are not connected to the image boundary.

The `imfill` function returns the output image, which is the input image with the specified regions filled according to the chosen criteria. The filled regions will typically have pixel values that match the foreground or object of interest.

3.4 Edge Detection

Edge detection is a fundamental technique in image processing used to identify boundaries or edges between different regions or objects within an image. It plays a crucial role in various computer vision tasks such as object detection, image segmentation, and feature extraction. Several methods have been developed for edge detection, and here are some that are used in this thesis.

3.4.1 Morphological operators

Morphological operators can be used for edge detection by leveraging the properties of morphological transformations to highlight boundaries or edges in an image [12]. The two commonly used morphological operators for edge detection are:

1. **Erosion:** Erosion is a morphological operator that shrinks or erodes regions of an image. It works by sliding a structuring element (a small binary matrix) across the image and replacing each pixel with the minimum value within the neighborhood defined by the structuring element [13]. Erosion accentuates boundaries and decreases the size of foreground regions. By subtracting the eroded image from the original image, the resulting image will highlight the edges.
2. **Dilation:** Dilation is the counterpart of erosion and expands or dilates regions in an image. Like erosion, it uses a structuring element to slide across the image, replacing each pixel with the maximum value within the neighborhood defined by the structuring element. Dilation strengthens the boundaries and increases the size of foreground regions. Subtracting the original image from the dilated image results in an edge-highlighted image.

By combining erosion and dilation, more advanced morphological operators can be created for edge detection. Two examples that are used in this thesis are:

1. **Opening:** Opening is a sequence of an erosion followed by a dilation. It helps in removing noise and small details while preserving larger connected edges. The process involves applying erosion to the image to remove small features, followed by dilation to restore the remaining edges. Subtracting the opened image from the original image highlights the edges.
2. **Closing:** Closing is a sequence of dilation followed by erosion. It helps in closing small gaps or holes in edges while preserving the overall shape. The process involves applying dilation to the image to close gaps, followed by erosion to smooth the edges. Subtracting the original image from the closed image highlights the edges.

Morphological operators are effective for edge detection in scenarios where the edges are well-defined and the background noise is relatively low. However, they may produce thick or fragmented edges, and their effectiveness depends on the choice of the structuring element and the specific characteristics of the image. Adjusting the size and shape of the structuring element can have a significant impact on the detected edges.

3.4.2 Laplacian of Gaussian

The Laplacian of Gaussian (LoG) method is an edge detection technique that combines the use of the Laplacian operator and Gaussian smoothing. It is designed to detect edges by highlighting regions of rapid intensity changes in an image [12]. The LoG method involves the following steps:

1. Gaussian smoothing: The image is convolved with a Gaussian kernel. This step helps to reduce noise and eliminate small-scale details that may lead to spurious edge detections. The size of the Gaussian kernel determines the scale at which edges are detected. A larger kernel size captures broader edges, while a smaller size focuses on finer details.
2. Laplacian operator: The Laplacian operator is applied to the smoothed image. The Laplacian is a second-order derivative operator that measures the rate of change of intensity. By convolving the smoothed image with the Laplacian operator, regions with rapid intensity changes, corresponding to edges, are enhanced. The Laplacian operator is commonly represented by a 3x3 or 5x5 kernel.
3. Zero-crossing detection: The Laplacian response obtained from the previous step is analyzed to identify edge locations. The zero-crossings in the Laplacian response correspond to the regions of maximum intensity change, which indi-

cate the presence of edges. A zero-crossing occurs when a pixel changes sign in the Laplacian response (i.e., it transitions from a positive value to a negative value or vice versa).

4. Thresholding: Since the Laplacian response may contain noise and weak edge responses, thresholding is often applied to distinguish true edges from noise. A threshold is set, and only the zero-crossings with magnitudes above the threshold are considered as valid edges.

The Laplacian of Gaussian method provides edge detection at multiple scales due to the involvement of Gaussian smoothing. It is particularly useful for detecting edges with varying widths and for identifying edges in noisy images. However, it can also produce thick edges and may be sensitive to the selection of the scale parameter and threshold.

3.5 Interpolation Methods

Interpolation is a technique used in image processing and computer graphics to estimate values or fill in missing data points between known data points. It involves generating intermediate values based on the surrounding known data points to create a smoother and more continuous representation of the data.

In the context of image processing, interpolation is often employed to adjust distortions, enhance resolution, or resize images. When resizing an image, for example, interpolation is used to estimate the color or intensity values of new pixels that are inserted between existing pixels. This helps to maintain the visual quality and smoothness of the image during the resizing process.

There are various interpolation methods commonly used [14], including:

Nearest Neighbor Interpolation
This method assigns the value of the nearest known data point to the new data point. It is a simple and computationally efficient interpolation technique but may result in pixelation and loss of fine details.

Bilinear Interpolation
Bilinear interpolation estimates the new pixel value based on a weighted average of the surrounding four known data points. It produces smoother results compared to nearest neighbor interpolation but may still cause some blurring.

Bicubic Interpolation
Bicubic interpolation considers a larger neighborhood of sixteen known data points to estimate the new pixel value. It uses a weighted average, taking into account

both distance and intensity differences. Bicubic interpolation typically yields higher-quality results with less blurring but requires more computational resources.

Cubic B-spline
Cubic B-spline constructs smooth and continuous curves or surfaces by connecting adjacent control points using cubic polynomial segments. Cubic B-spline interpolation offers visually pleasing results and handles irregularly spaced data well, but may introduce some smoothing effects.

The choice of interpolation method depends on the specific application and desired trade-offs between computational complexity, smoothness, and preservation of fine details. Different methods may be more suitable for different scenarios, and experimentation is often required to determine the most appropriate interpolation technique for a particular task.

3.6 Image Registration

Image registration is the process of aligning different images of the same scene or object to a common coordinate system. It involves finding the spatial transformation that minimizes the differences or maximizes the similarity between corresponding points or features in the images. The goal of image registration is to correct for geometric distortions, such as translations, rotations, scaling, and deformations, as well as to account for differences in sensor geometry, viewpoint, or imaging conditions.

In this thesis the intensity based image registration was used to align images from the hyperspectral camera and the reference camera. The process involves measuring the similarity between the intensities of corresponding pixels in the images and iteratively adjusting the transformation parameters to maximize this similarity.

Typically, intensity-based image registration begins by selecting a similarity metric, such as sum of squared differences (SSD), normalized cross-correlation (NCC), or mutual information (MI). These metrics quantify the agreement between the intensity values of corresponding pixels or regions in the images. The transformation parameters, such as translations, rotations, or deformations, are adjusted to optimize the chosen similarity metric [15].

During the registration process, an optimization algorithm, such as gradient descent or the Levenberg-Marquardt algorithm [16], is used to iteratively refine the transformation parameters. The algorithm searches for the optimal transformation that minimizes the dissimilarity between the intensities of corresponding pixels.

4 Compensation of Image Distortions

This chapter is about compensation of distortions caused by lizard's breathing movement in the trunk area. It's divided into multiple sections where each step of the compensation algorithm is explained.

The algorithm is written in MATLAB programming language with a student license and version R2021b [17]. To run the functions used in the script, it is necessary to install the Image Processing Toolbox [18]. The diagram of compensation algorithm is displayed in Fig. 12.

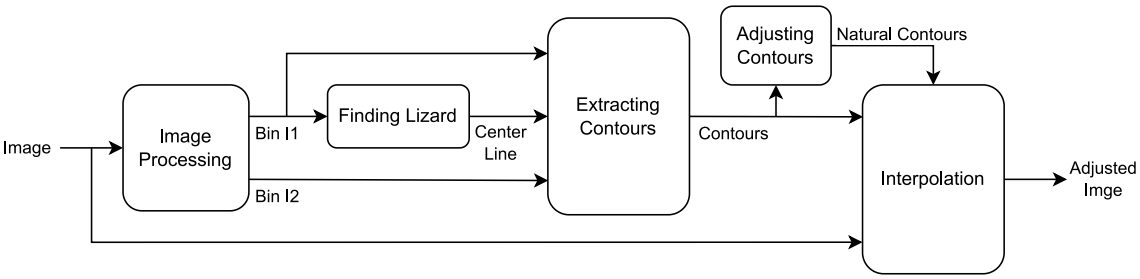


Fig. 12: Diagram of compensation algorithm.

4.1 Image Processing

The images are loaded from the specified `image_path`, which contains the RGB image captured by the hyperspectral camera and a folder with reference images. The LSC image is loaded into the workspace using the `imread` function, and then it is cropped to a resolution of 1000 × 1000 pixels. The resulting cropped image is displayed in Fig. 13.

4.1.1 Splitting image into RGB channels

The LSC image is then split into its RGB channels. As shown in Fig. 14, each channel exhibits a different reflection of the scanned object. The red and green channels successfully capture the white back patterns (small white dots inside the black circles) of the lizard, while the blue channel does not reflect them. Consequently, the blue channel image is more suitable for subsequent image processing as it facilitates the separation of the lizard's body from the background.

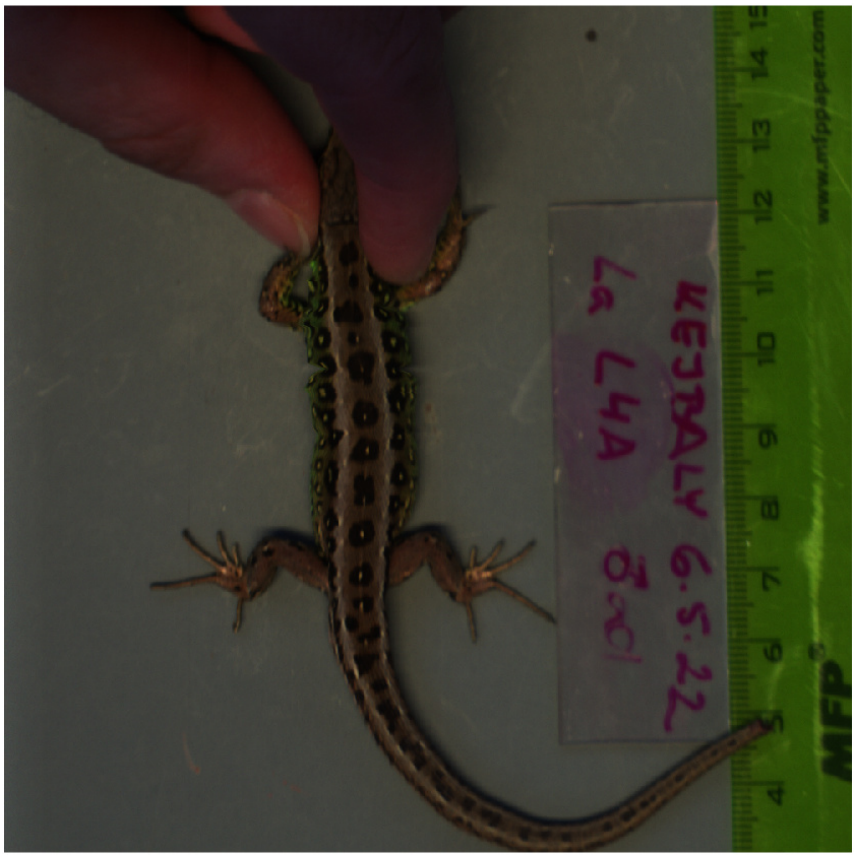
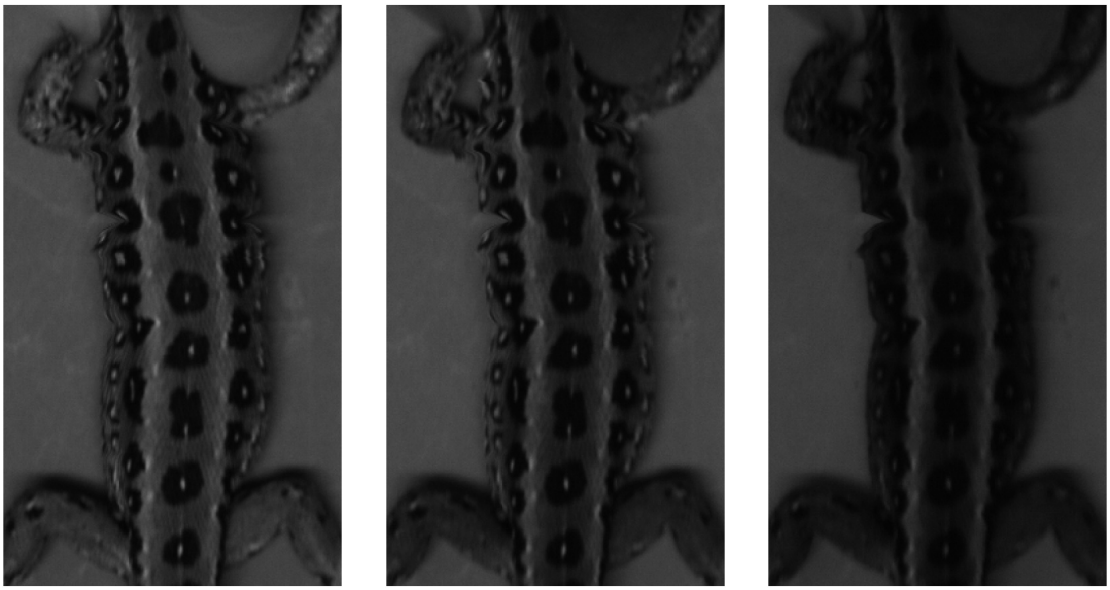


Fig. 13: Cropped LSC image in RGB colors with a resolution of 1000×1000 pixels.

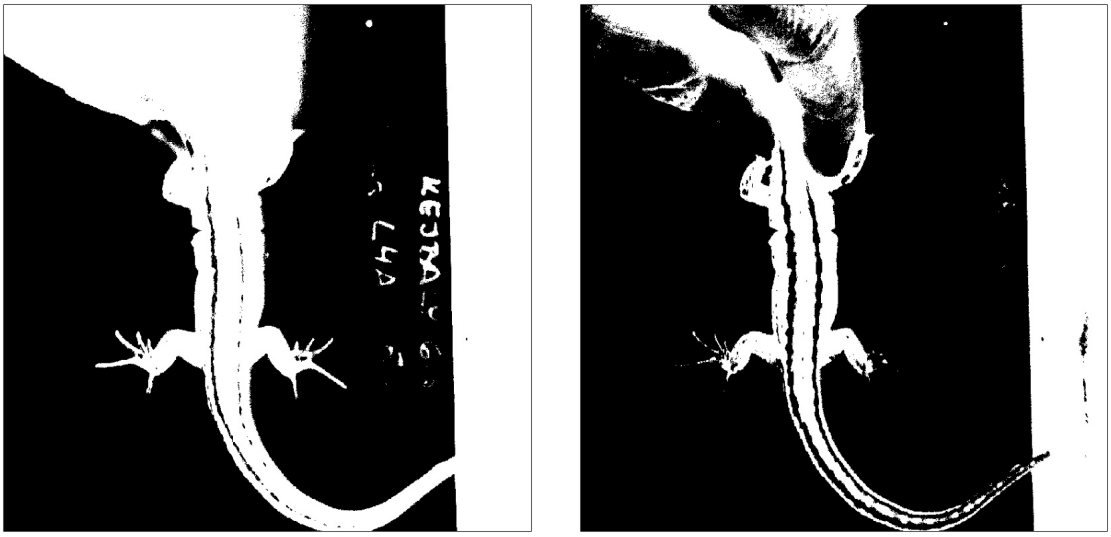


(a) (b) (c)

Fig. 14: Detailed view of the cropped trunk areas of the (a) red, (b) green, and (c) blue channels from the LSC image.

4.1.2 Binarization

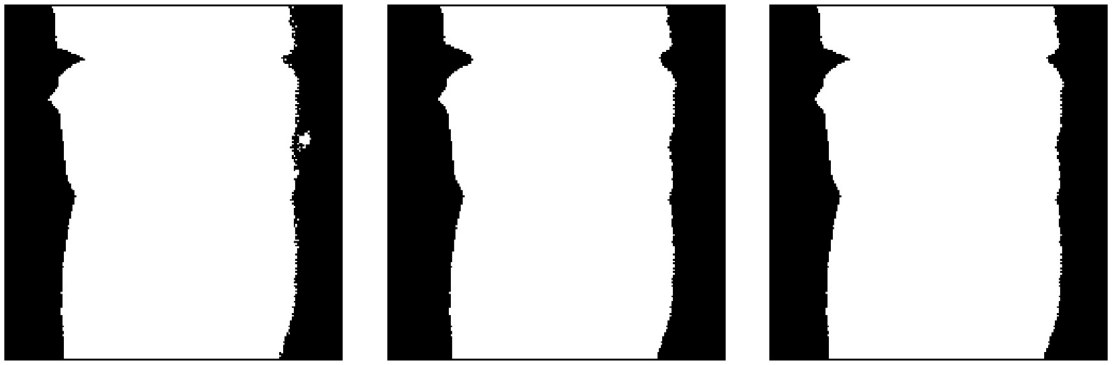
Binarization is performed using a simple thresholding method on the blue channel image. The threshold values of 40 and 25 were experimentally determined to effectively detect the edge pixels on the left and right sides of the lizard's body, respectively. In this method, a threshold value is set, and all pixels below this threshold (the darkest pixels) are assigned a foreground value of 1, while the remaining pixels are considered background and assigned a value of 0. The double thresholding is necessary to account for shadows present on the right side of the lizard's body (threshold 25), which could otherwise lead to false edge detection in the binarized image. Both resulting binarized images are displayed in Fig. 15.



(a) (b)

Fig. 15: Binarized images were obtained for threshold values of 40 (a) and 25 (b), respectively.

After binarization, it is necessary to remove unwanted noise from the images and prepare them for the next step, which involves detecting the edges of the lizard and identifying the trunk area. Firstly, the `imfill` function (3.3) is applied with default settings to fill any enclosed black areas with white pixels. Subsequently, erosion and dilation operations are performed on the image using a disk-shaped morphological structuring element of size 3. This structuring element was used with `imerode` and `imdilate` MATLAB functions to perform these operations. The complete sequence of steps applied to the first binarized image (Fig. 15 (a)) is illustrated in Fig. 16. The same steps were also applied to the second binarized image.



(a) (b) (c)

Fig. 16: The binarized image from Fig. 15 (a) is subjected to a filling operation to fill any enclosed regions within the lizard's body. Subsequently, the black pixels inside the body are removed in (a). The resulting filled image is then eroded, which eliminates a group of white pixels near the right edge of the lizard in image (a), but also causes the body of the lizard to become thinner in image (b). To restore the original width of the lizard, the eroded image is dilated, resulting in image (c) with the same width as the original.

4.2 Location of Lizard in Image

The precise location of the lizard within the image is unknown, and it is crucial to determine its position for further processing. To accomplish this, the image is scanned using algorithm 1 within the typical region (between row 200 and 700) where the lizard's body is expected to be.

The algorithm is designed to detect the edges of the lizard and record their locations. However, there are instances where it may not trace the entire body of the lizard. This typically occurs between the lizard's fingers and legs, as well as in the presence of smaller white areas that are not part of the lizard's body. The scanning path (shown in red) and the skeleton line (shown in blue) can be observed in Fig. 17.

4.2.1 Center Line

In Fig. 17, it is evident that the center line (blue) is not connected throughout the whole lizard's body. To detect the edges effectively, it is necessary to create a connected line in which each row contains a pixel inside the lizard's body. This approach allows for edge detection, starting from the center line. The reason for focusing on the center line is that in certain areas of the image, it is challenging to detect the trunk's edge from the outside. An example of this can be seen in Fig. 17, above the lizard's back left leg.

Algorithm 1 Finding center of the lizard

CL $\leftarrow$ zeros(size(**I**, 1),size(**I**, 2)) ▷ Where **CL** is the center line image
▷ and **I** is the first binarized image
$idx \leftarrow 0$ ▷ Initialized indexing variable
for $i = 200 : 700$ **do**
  $idx \leftarrow idx + 1$
  $pix \leftarrow 0$ ▷ Initialized pixel value to 0 (background)
  $j \leftarrow 100$
  while $pix \neq 1$ **do** ▷ While we don't encounter the lizard
    $j \leftarrow j + 1$
    $pix \leftarrow \mathbf{I}(i, j)$
  end while
  $L_{edge}(idx, :) \leftarrow [i, j]$ ▷ Save this index as the left edge of the lizard
  while $pix \neq 0$ **do** ▷ While we don't encounter the end of the lizard
    $j \leftarrow j + 1$
    $pix \leftarrow \mathbf{I}(i, j)$
  end while
  $R_{edge}(idx, :) \leftarrow [i, j]$ ▷ Save this index as the right edge of the lizard
  $center \leftarrow$round$(\frac{R_{edge}(idx,2)-L_{edge}(idx,2)}{2})$ ▷ Find the center
  CL$(i, center) \leftarrow 1$ ▷ Set the center pixel to 1
end for

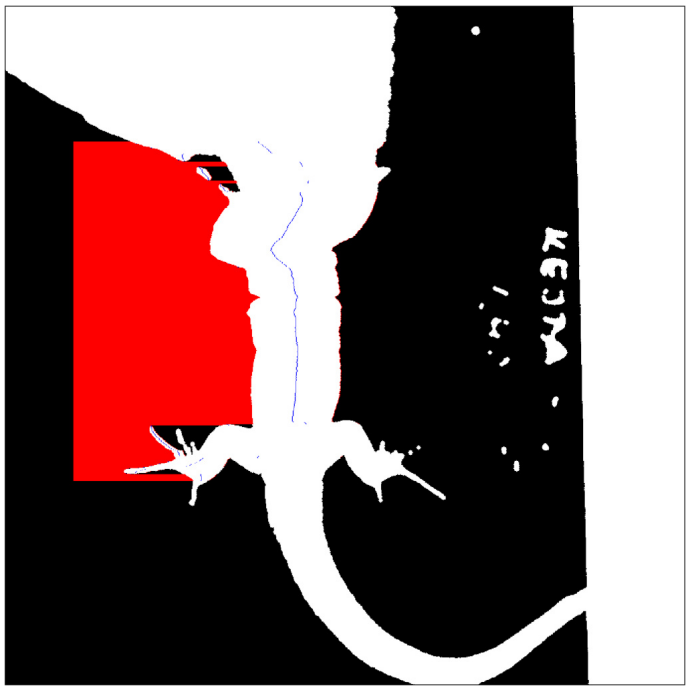


Fig. 17: Location of lizard's body in the LSC image.

Obtaining the center line, or skeleton line, cannot be achieved solely through skeletonization algorithms. This is because the lizard image may exhibit significant distortions caused by breathing, leading the algorithms to mistakenly identify additional body parts within the trunk area. Instead, the center line calculated in algorithm 1 is utilized. The pixels are connected through dilation using a disk-shaped structuring element of size 7. Subsequently, the largest white object is selected, and a thinning operation is performed using the `bwmorph` function. These steps are illustrated in Fig. 18 (not all white pixels are visible in Fig. 18 (a) and Fig. 18 (d)).

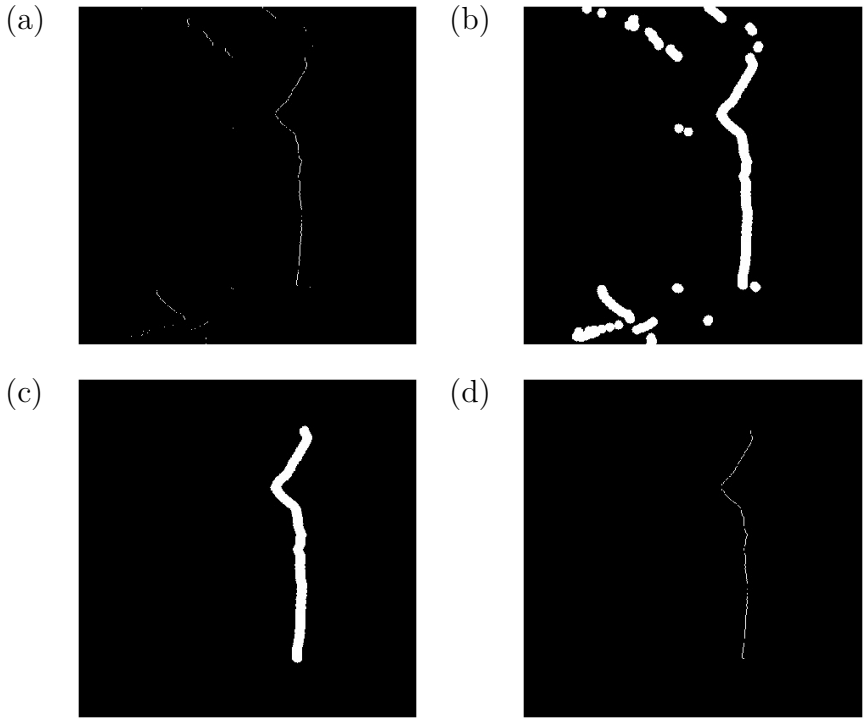


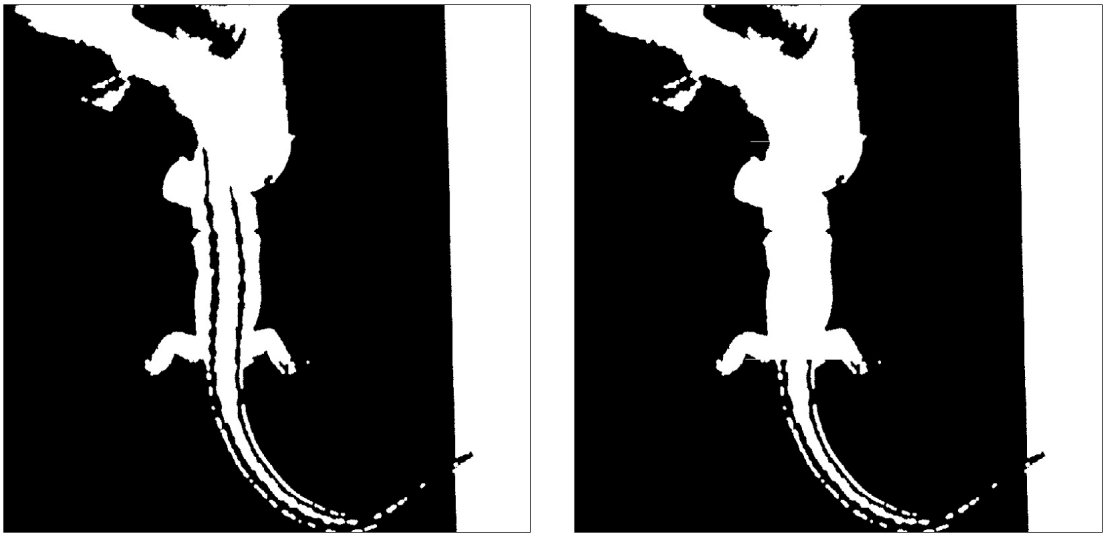
Fig. 18: Extracting center line: (a) center line from Fig. 17, (b) dilation performed, (c) extracting largest white area, (d) thinning performed.

However, it should be noted that in Fig. 18 (d), an issue may arise during thinning, resulting in two white pixels in a single horizontal image row. Since only one starting pixel is necessary, the `unique` function was employed to remove the extra pixels from the center line. Additionaly, a line of pixels is added to the start and the end of this center line, to ensure the detection of the whole trunk area.

4.3 Extracting Contours

Now that the center line has been obtained, the starting pixel locations are known, enabling the detection of edges from the center line. In section 4.1.2, the `imfill` function was utilized to remove black pixels from the lizard's trunk area. While this

approach was successful on the first binarized image (Fig. 15 (a)), it did not work on the second image (Fig. 15 (b)) due to the absence of closed black pixel areas caused by a lower threshold value. To address this, a stripe of white pixels is added to the end of the center line, effectively closing the black pixel regions and allowing the `imfill` function to fill the black pixels in lizard's trunk area (see Fig. 19).



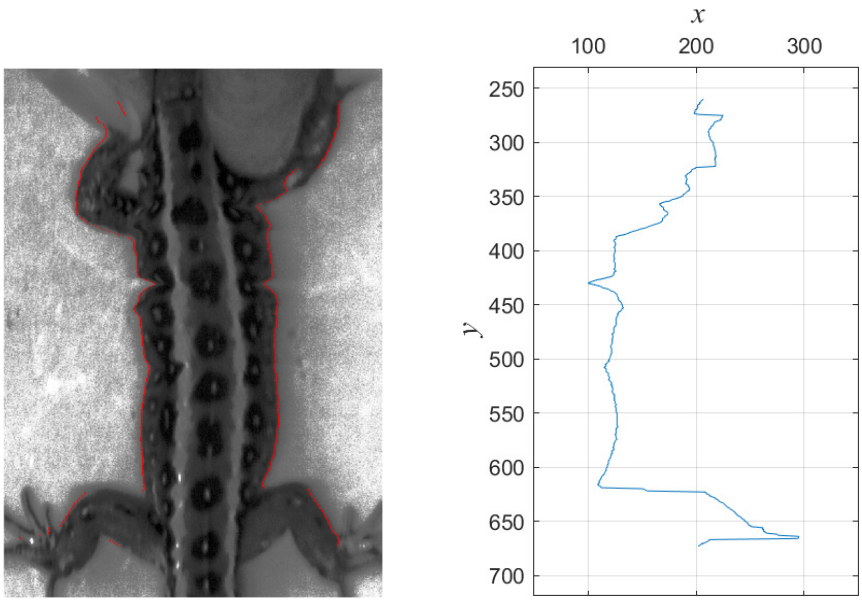
(a) (b)

Fig. 19: Second binarized image from Fig. 15 (b) after denoising, is displayed in (a) before adjusted filling and (b) after adjusted filling.

Both binarized images are now prepared for detection of the edges from the center line. In the first image, scanning is conducted in the left direction starting from the center line, resulting in the detection of the left contour of the lizard. In the second image, scanning is performed in the right direction from the center line, yielding the right contour of the lizard. The detected contours are highlited in Fig. 20 (a) with red lines. The calculated area between the edges is displayed in Fig. 20 (b).

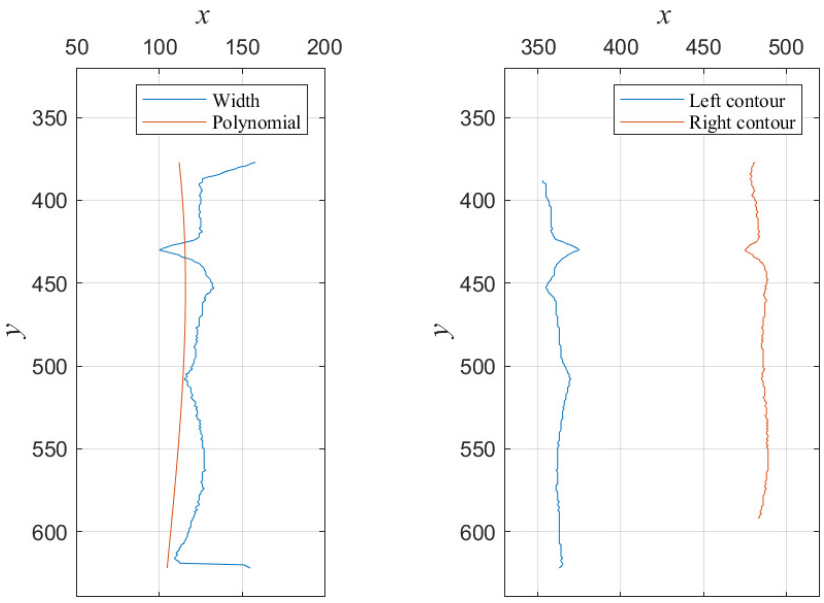
4.3.1 Finding Trunk Area

In Fig. 20 (b), it can be observed that the most consistent width (small change in the x-axis) along the lizard's trunk area lies between the front and back pair of legs. This information is utilized to locate this specific area through cross-correlation with the experimentally derived polynomial function, which represents the average length and width of the lizard's trunk area. The trunk area, displayed in Fig. 21 (a), corresponds to the location of the curve with the minimal correlation value. This indicates the part of the lizard that deviates the least from the calculated polynomial curve.



(a) (b)

Fig. 20: Detected edges from the center line are highlighted with red in (a). Calculated area between the edges plotted on the x-axis, with the y-axis corresponding to the image rows displayed in (b).



(a) (b)

Fig. 21: Finding the trunk contours of the lizard's body. Graph (a) shows a location where the calculated polynomial (red) has the highest similarity with the trunk width of the lizard (blue). Graph (b) shows the lizard's trunk contours with eliminated leg pixels.

In Fig. 21 (a), the contours still include some pixels that belong to the legs (at the top and the bottom of the blue curve), resulting in a wider width compared to the nearly constant width of the trunk. To ensure accurate processing, it is necessary to remove these pixels from the contours since they would interfere with the calculation of natural contours using a polynomial in the subsequent step. The algorithm 2 is employed to delete these pixels and the here is a brief explanation on what it does:

- It examines the first 11 pixels of the contour and calculates the standard deviation along the x-axis.
- If the standard deviation exceeds 1, the algorithm continues by considering the 2nd pixel through the 12th pixel and performs the same calculation.
- This process continues until the standard deviation falls below 1.

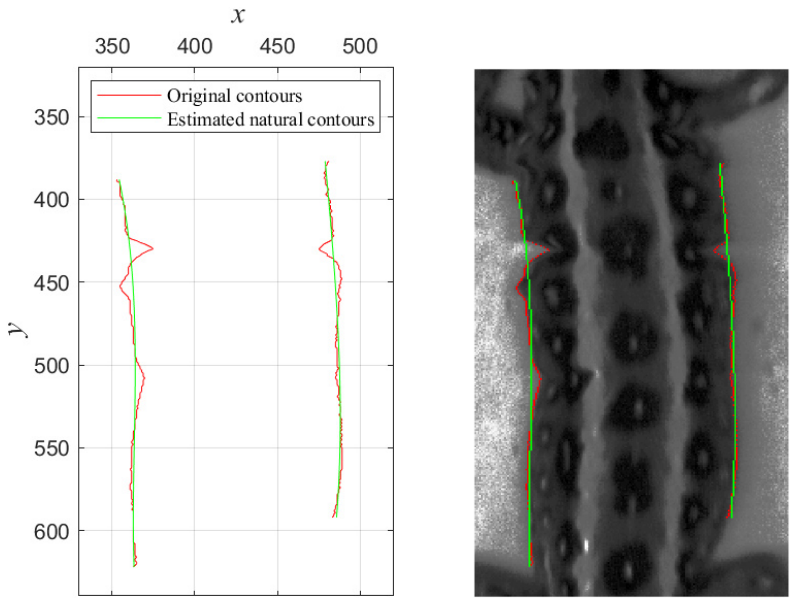
The algorithm 2 is performed in a top-down direction on the starting pixels of both the left and the right contours, eliminating the pixels from the front pair of legs. With the same idea a similar algorithm is performed in the bottom-up direction, eliminating the pixels from the back pair of legs. The contours without the leg pixels are shown in Fig. 21 (b).

Algorithm 2 Deleting leg pixels from contours.

$i \leftarrow 1$
while std($L\_trunk\_edge(i : i + 10, 2)) > 1$ **do**
    $i \leftarrow i + 1$
end while
$L\_start \leftarrow i$

4.4 Calculating Natural Contours

The main idea of this thesis is to compensate for breathing distortions to restore a normal appearance of the lizard's body. To achieve this, the calculation of new natural contours of the lizard's body is performed. This calculation is done by fitting the obtained lizard contours with a low degree polynomial, so the quick changes in the lizard's width are neglected and the final contours are smooth. A 3rd degree polynomial is selected as it creates gentle curves in the contours. The results of this fitting process are displayed in Fig. 22.



(a) Obtained and estimated contours.

(b) Contours of the lizard's body.

Fig. 22: Estimating the natural contours of the lizard's body. Graph (a) shows the original contours (red) and estimated natural contours of the lizard (green). Image (b) shows both of these contours highlighted in a cropped grayscale LSC image.

4.5 Interpolation

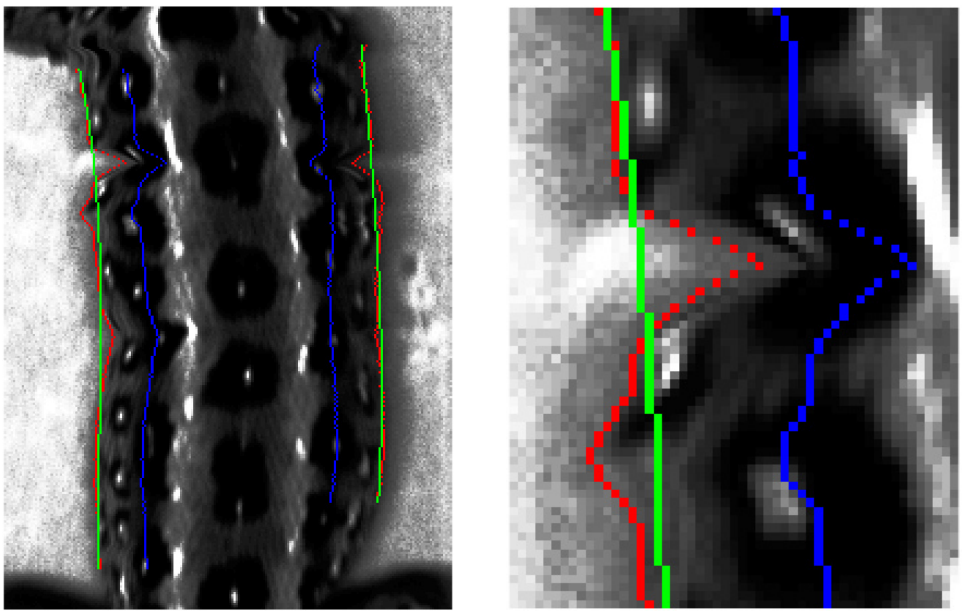
Before the interpolation, it is necessary to define the interpolation ranges to determine which pixels should be interpolated and by how much. The two borders that have already been set in Fig. 22 are as follows:

- Red pixels - representing the actual contours of the lizard's body.
- Green pixels - representing the estimated natural contours of the lizard's body.

The missing border is the one that determines where the interpolation should start. This border is calculated based on the actual contours of the lizard and is then multiplied by a factor of 0.8, which smooths out the peaks of the original contours. This factor has been experimentally estimated to improve the interpolation results. The underlying idea is that during the respiration cycle, the lizard's body exhibits the most movement at the edges, which gradually decreases towards the center of the lizard. The starting interpolation pixels for the left side and the right side are calculated using the following code:

$$Lstart \leftarrow Lnatural(:,2) + \text{round}((Lcontour(:) - Lnatural(:,2)) * 0.8),$$
$$Rstart \leftarrow Rnatural(:,2) + \text{round}((Rcontour(:) - Rnatural(:,2)) * 0.8),$$

where *Lcontour* and *Rcontour* are the left and right contours of the lizard, the *Lnatural* and *Rnatural* are the natural contours calculated in section 4.4, and *Lstart* and *Rstart* are the starting pixels of the interpolation. The rounding is done with the MATLAB function `round` to the closest whole number, because the pixel coordinates can't have decimal places. These contours are then shifted in x-axis to the positions that will define the interpolation range. The shifts of the *Lcontour* and *Lnatural* are the same and are defined with a constant variables $U1 = -1$ and $T1 = -1$, which provide a shift to the left by one pixel. The *Lstart* is shifted to the center of the lizard's body with constant variable $F1 = 20$. The same constants are set for the right side, but with the opposite signs. The start of the interpolation is depicted in blue color in Fig. 23.



(a) (b)

Fig. 23: Calculated and shifted interpolation ranges with original contours (red), estimated natural contours (green), and calculated starting contours (blue) are highlighted in the equalized grayscale image (a). Image (b) displays a cropped detail of these interpolation ranges.

4.5.1 Interpolation Algorithm

The last step of the compensation algorithm involves interpolation. Interpolation is performed on each row of the LSC image with defined interpolation range, for each side of the lizard's body and for each RGB channel individually. Consequently, all the adjusted RGB channels are concatenated into a single colorful image. As explained in section 3.5, there are multiple interpolation methods available. In this

thesis, the spline method is utilized, specifically the MATLAB function `spline`, to calculate and fit curves to the pixel intensity values within the specified range.

When interpolating the lizard's body, the interpolation will not be done linearly, but quadratically. The decision to use quadratic interpolation stems from analyzing the lizard's body movement during the respiration cycle. The pixels at the edge exhibit greater movement compared to those closer to the lizard's spine. It is worth noting that some edge pixels move inward to the extent that the edge patterns vanish, making it impossible to interpolate them due to the lack of information. Although the reference pictures could potentially provide the necessary information, implementing such complexity is beyond the scope of this algorithm.

The following algorithm was developed to interpolate the pixel values between the red and blue lines in Fig. 23 into the pixels between the green and blue lines:

1. First, the channel matrix is selected from a set of all RGB channels $\mathcal{X}$, and assigned to a matrix **CH**. The same matrix is then copied to a new matrix **NCH**, on which the adjustments will be performed. The channel matrix is a 1000×1000 matrix containing pixel intensity values.
2. The second step involves defining a for loop that iterates through each row of pre-set interpolation ranges. The loop accounts for each side of the lizard's body, as they may have different rows requiring adjustment. In this example, the focus is on adjusting the left side of the lizard's body.
3. Each iteration selects a row y, where the interpolation is performed and the following variables are defined:
    - *from* - x coordinate in row y where the interpolation should start (blue pixel).
    - *until* - x coordinate in row y, that specifies the end of the interpolation range (red pixel).
    - *to* - x coordinate in row y, that specifies to which range should be the interpolation preformed (green pixel).
4. If the variable *until* and the variable *to* don't match (meaning that the red and green pixels do not have the same x coordinate), the interpolation is performed, and the following variables are defined:
    - X - is an array of x coordinates of the pixels that need to be interpolated between *from* and *until* coordinates.
    - Z - is an array containing the intensity values corresponding to the x-coordinates in array X.
    - R - is an array with the length of interpolation range in a specified quadratic range.

- F - is an array with the squared values of R.
- D - is an array of function values F divided by the value range calculated by subtracting the last array value of F from the fisrt one.
- N - is an array of normalized function values of D between 0 and 1.
- XX - is an array of x coordinates with a length of interpolation range fitted between *from* and *until* coordinates.
- ZZ - is an array of calculated pixel intensity values for XX positions.
- **NCH** - it the new channel matrix where the calculated ZZ values are substituted in between *from* and *to* coordinates.

The pseudocode for interpolation of the lizard's left side of the body is written in algorithm 3. The explanatory graph is displayed in Fig. 24.

Algorithm 3 Interpolation of the left side of the the lizard's body.

CH $= \mathcal{X}\{channel\}$ ▷ Channel selection
NCH $=$ **CH**
for $i = 1$:size$(Lnatural, 1) - skip$ **do** ▷ Interpolation of left edge
$from \leftarrow Lstart(i) + F1$
▷ *Lstart* is an array of x coordinates for the left interpolation start
$until \leftarrow Lcontour(i) + U1$
▷ *Lcontour* is an array of x coordinates of original contour
$to \leftarrow Lnatural(i, 2) + T1$
▷ *Lnatural* is an array of x coordinates of natural contour
if $until \neq to$ **then**
$X \leftarrow until : from$
$Z \leftarrow$double(**CH**$(Lnatural(i, 1), X))$
$R \leftarrow$ linspace$(-7, -6.5, from - to + 1)$
$F \leftarrow R^2$
$D \leftarrow \frac{F}{(F(1)-F(end))}$
$N \leftarrow D - D(1)$
$XX \leftarrow$flip$(from + N * (from - until))$
$ZZ \leftarrow$round(spline(X, Z, XX))
NCH$(Lnatural(i, 1), to : from) \leftarrow ZZ$
end if
end for

The quadratic interpolation starts with specifying a range e.g.[-8,-3], within which the values will be reflected into quadratic function values, that define the differences in the gaps between the interpolation pixels. The array of these values, represented by the variable R, is calculated using the `linspace` MATLAB function.

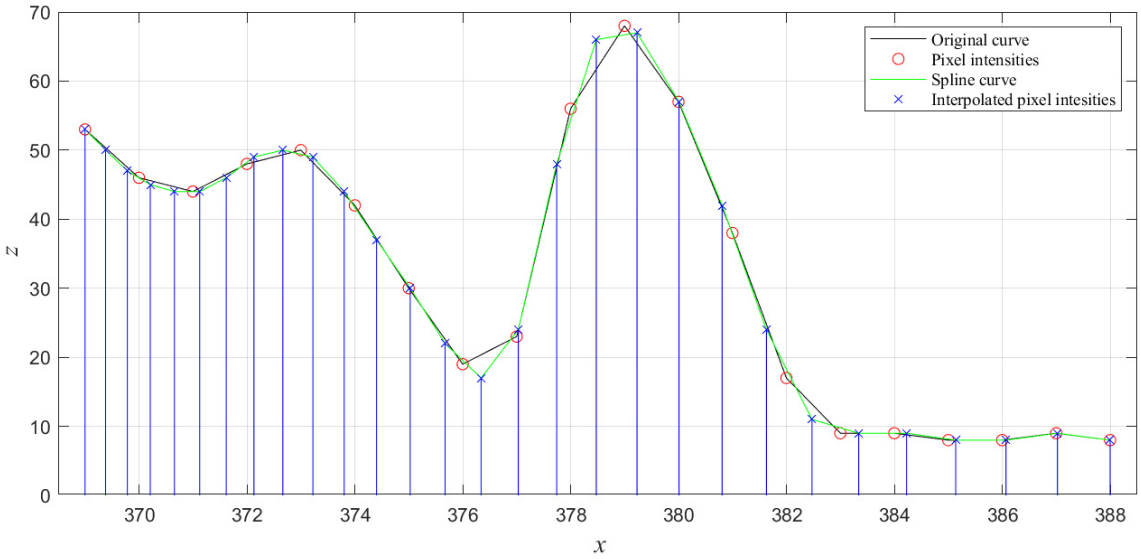


Fig. 24: An example of quadratic interpolation with a range of [-8,-3].

This function takes the specified range and the number of interpolation pixels, calculated by subtracting variable *from* from variable *to*. By squaring this array, the values are reflected on the y-axis of the quadratic function *f(x)*, which defines the array variable *F*. The values in array F do not have a constant gap between them. The calculation of quadratic function values can be seen in Fig. 25. These *f(x)* values are then normalized and multiplied by the number of interpolation pixels. This results in more pixels being interpolated near the edges of the lizard's body and fewer pixels closer to the center of the lizard. The quadratic range [-7, -6.5] used in this algorithm was estimated experimentally to yield the best results, while the range [-8, -3] is used here for explanatory purposes only.

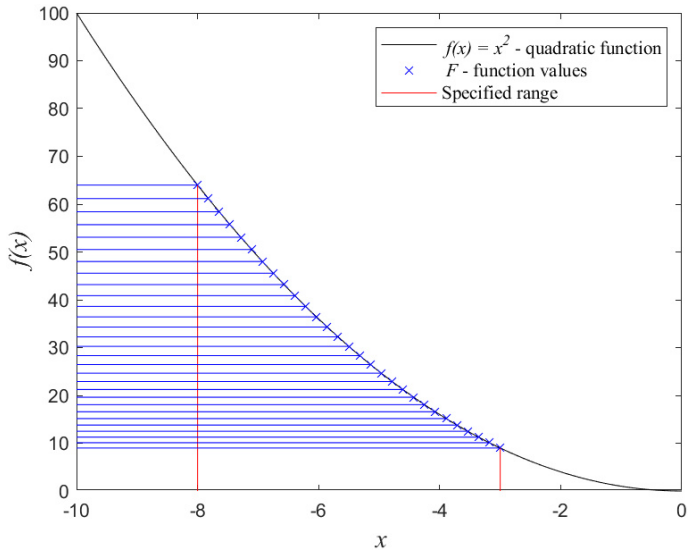


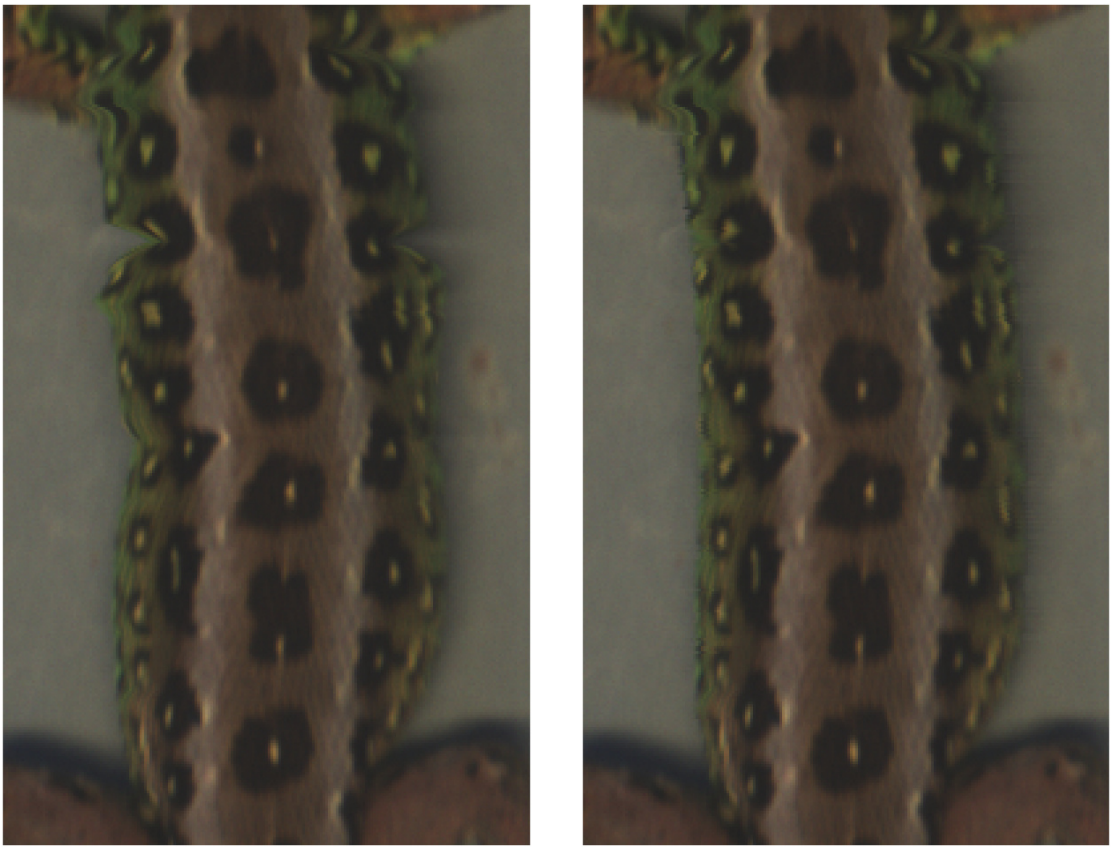
Fig. 25: Calculation of quadratic function values.

For the background, the linear interpolation is used, which helps in creating natural shadows from the lizard. Linear interpolation means that the pixel values are calculated with equal spacing between pixels. This algorithm operates similarly as the algorithm 3, and the pseudocode in algorithm 4 is employed.

Algorithm 4 Interpolation of the background shadows for the left side.

for $i = 1$:size$(Lcontour, 1) - skip$ **do**
$from \leftarrow Lcontour(i) - 30$ ▷ Shifted original contour by 30 pixels further from the body
$until \leftarrow Lcontour(i) - 3$ ▷ Shifted original contour by 3 pixels further from the body
$to \leftarrow Lnatural(i, 2) + T1 - 1$ ▷ Natural contour with multiple shifts
$X \leftarrow from : until$
$Z \leftarrow$double(**CH**$(Lnatural(i, 1), X)$)
$XX \leftarrow$linspace$(from, until, to - from + 1)$
$ZZ \leftarrow$round(spline(X, Z, XX))
NCH$(Lnatural(i, 1), from : to) = ZZ$
end for

After interpolating each side of the lizard and the background, the adjusted channels are concatenated into one colorful image. The interpolation results, which are slightly enhanced for better observation, are displayed in Fig. 26.



(a) (b)

Fig. 26: Comparison of (a) original LSC image and (b) adjusted LSC image after interpolation

5 Evaluation of Compensation Algorithm

A part of this thesis is to evaluate how well the compensation of distortions is done. There are various factors that can be considered in this evaluation, such as the position and size of the back patterns, the normal appearance of the lizard's contours, and so on. The evaluation method used in this thesis is based on comparing the adjusted contours to a reference image chosen by the user. The reference image should represent the normal appearance of the lizard, without any eccentric or concentric distortions in the trunk. The diagram of evaluation algorithm is displayed in Fig. 27.

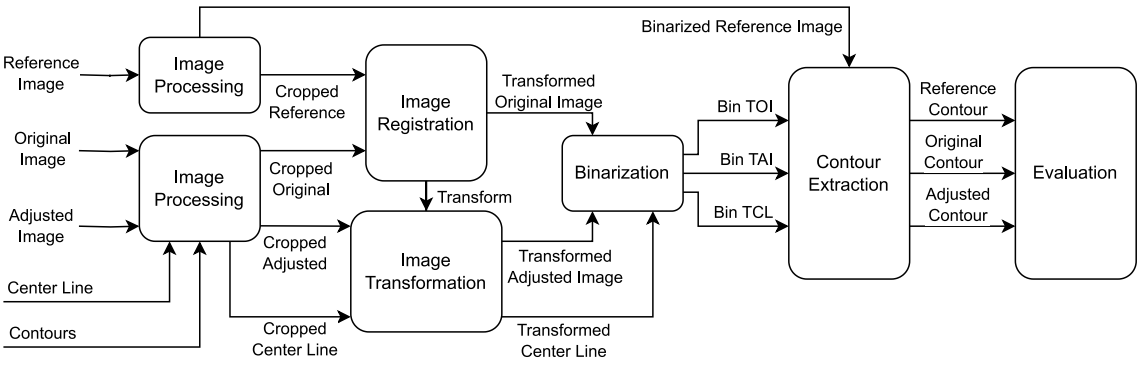
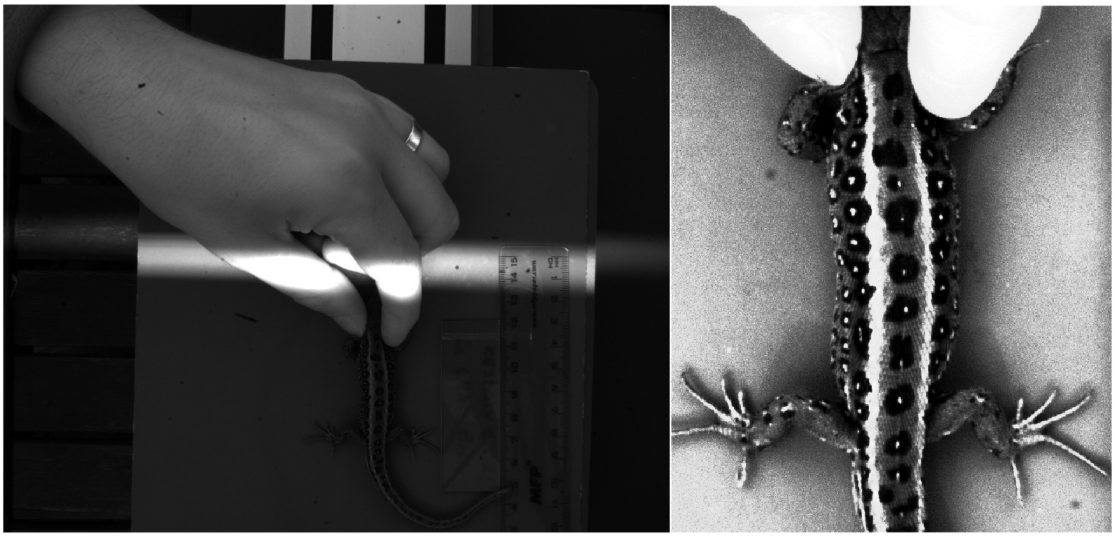


Fig. 27: Diagram of the evaluation algorithm.

5.1 Reference Image Processing

There are several reference images from which a user can choose one for comparison. As the scanning begins, the lizard is not yet positioned under the lighting stripe, which allows us to have the entire lizard's body under similar lighting conditions. The user should select one of these images, as the algorithm is specifically optimized for these very dark images. Otherwise, the algorithm may encounter difficulties in extracting accurate information. An example of a correctly selected image is shown in Fig. 28.

In Fig. 28 (b), we can see that by using the histogram equalization method, we can retrieve a lot of information about the lizard, even though the initial image is very dark. The histogram equalization is performed via `histeq` MATLAB function with default settings. The function takes an already cropped image without the white stripe from the light.



(a) (b)

Fig. 28: Selection of an appropriate reference image for comparison. Image (a) shows a correctly selected reference image in full-size resolution, without a white lightning stripe over the lizard's body. Image (b) displays a histogram equalized cropped reference image.

5.1.1 Cropping the Trunk Area in the Reference Image

As there is no information regarding the location of the lizard in the reference image, it is necessary to first locate the body of the lizard. The binarization method used for the image from the LSC cannot be applied here due to significant noise in the image. Therefore, another edge detection method called Laplacian of Gaussian is used. This method is briefly described in subsection 3.4.2. By using this method, the algorithm is able to identify pixels with the most significant changes in pixel intensity. The results of this method are displayed in Fig. 29.

Fig. 29 (a) already depicts the true appearance of the lizard in real life, but obtaining the lizard's contours is not yet possible. To achieve this, the body of the lizard has to be separated from the background. The Laplacian of Gaussian method has detected numerous edges within the lizard's body. By dilating these lines, the black holes between them are filled, resulting in a filled area of white pixels that resembles the shape of the lizard's body. As dilation increases the area by the size of the mask, performing erosion on the image with the same mask restores the area to its original size, but the pixels within the body are not erased. This process creates a binarized version of the lizard's body. By analyzing the largest white area in the image, the centroid can be calculated, typically located close to the middle of the lizard's trunk area. This centroid is then used to crop a rectangular image for comparison purposes. The cropped area is situated 250 pixels away from the

(a) (b)

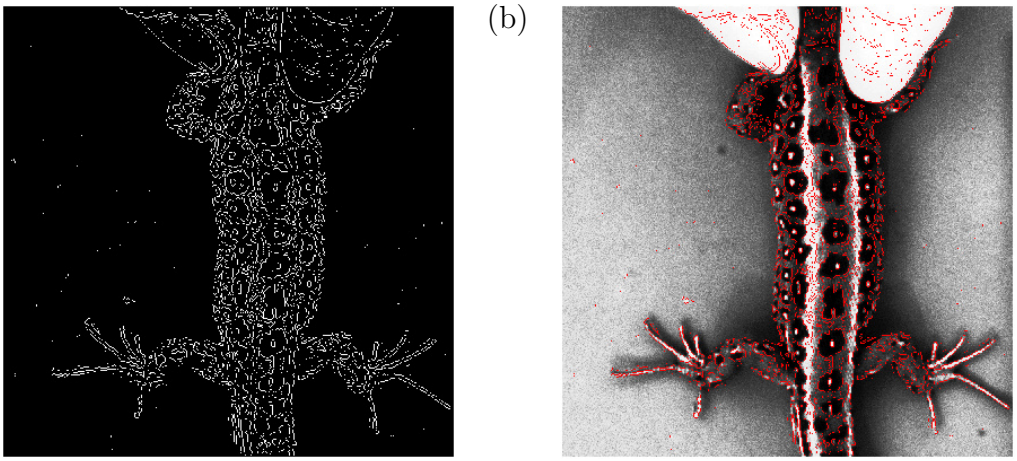
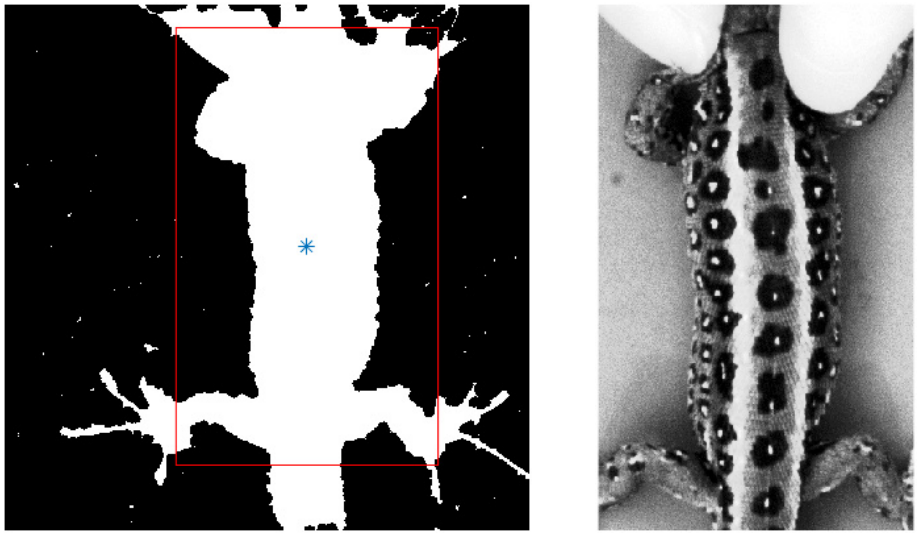


Fig. 29: Laplacian of the Gaussian (LoG) of the reference image. Image (a) shows the thresholded edges detected with LoG as white pixels. Image (b) highlights these edges (red) in a histogram equalized cropped reference image.

centroid in height and 150 pixels away in width, resulting in resolution of 501×301 pixels. These steps and results can be observed in Fig. 30 (a). The final cropped image is displayed in Fig. 30 (b).

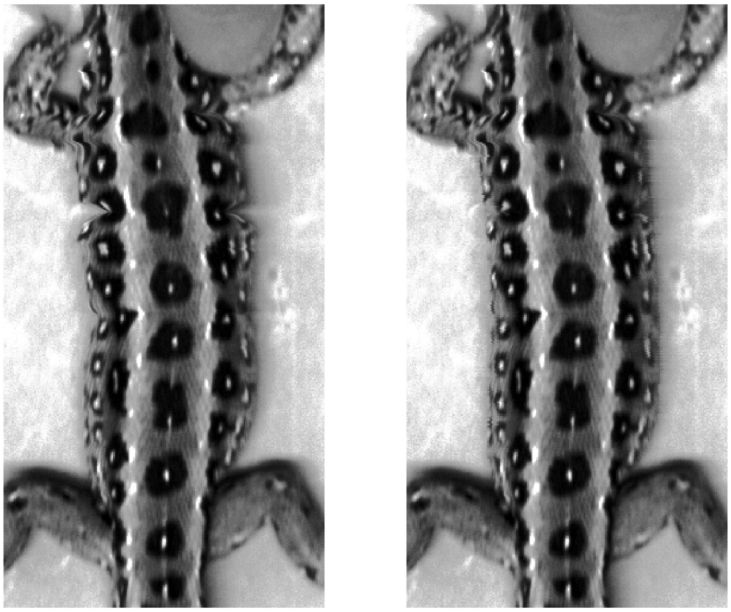


(a) (b)

Fig. 30: Finding and cropping the trunk area of the lizard. Image (a) shows a dilated and eroded LoG image from Fig. 29 (a). This image also highlights the centroid (blue) of the largest white area, along with the calculated rectangle for cropping (red). Image (b) displays the histogram equalized cropped reference image at the location of the rectangle in image (a).

5.2 Processing of LSC Images

In order to assess the quality of the lizard's adjustment, we need to align the same areas of the original image, the adjusted image, and the reference image. It is important to select an appropriate part of these images because the alignment would not work properly otherwise. To determine the area we want to align, we can use information about the location of the lizard's trunk area from detected contours (4.3.1) to crop only the necessary part of the lizard. The marginal pixels of these contours are selected, and with a reserve of 50 pixels in width and 100 pixels in height, this area is cropped. The cropped images are displayed in Fig. 31.



(a) (b)

Fig. 31: Cropped trunk area of the original (a) and adjusted (b) image from LSC.

5.3 Image Registration

In this thesis the `imregister` MATLAB function is used for intesity-based image registration [19]. The optimizer is created via `imregconfig` MATLAB function and is set to multimodal, because the images come from a different sensors [20]. In order to compare the LSC images with the reference image, image registration is performed between the original LSC image and the reference image. The unregistered and registered images are displayed in Fig. 32.

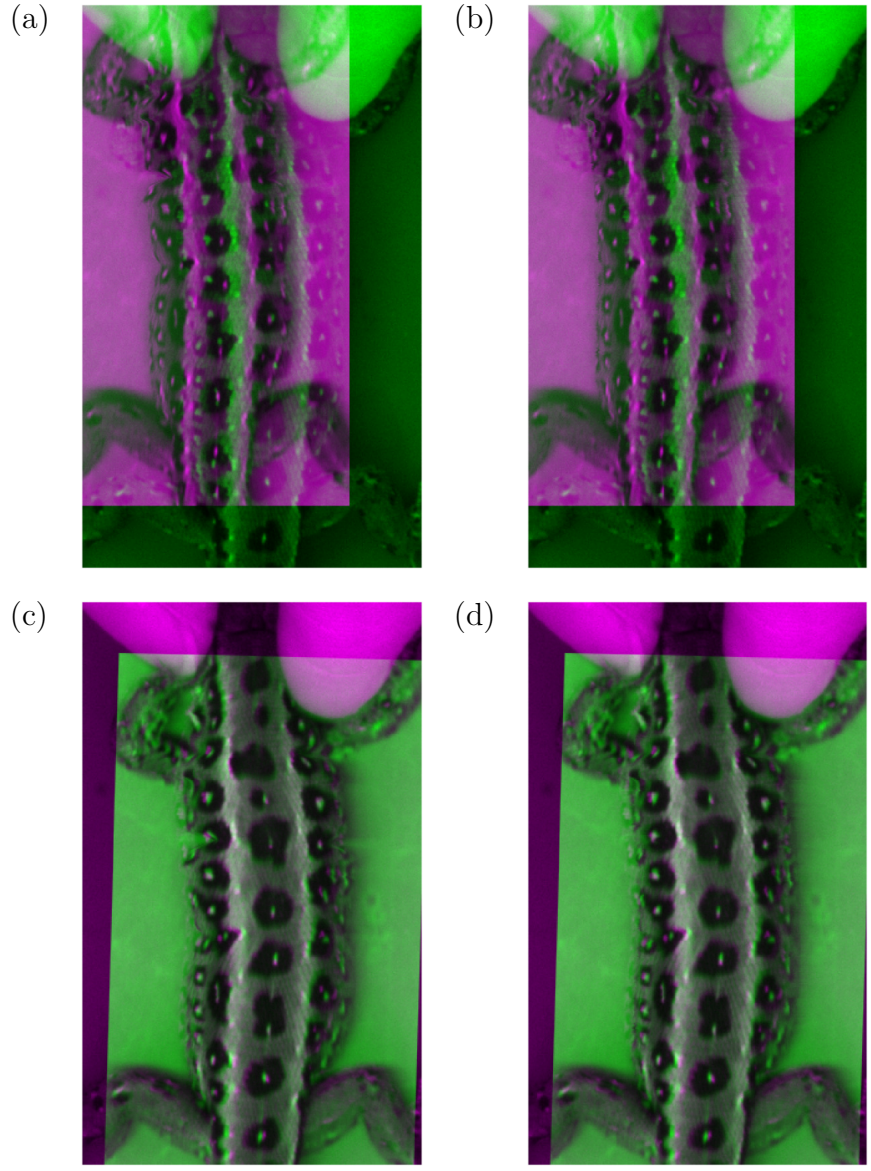


Fig. 32: Transformation of LSC images to coordinates of the reference image. Images (a) and (b) show the unregistered images from the LSC camera on a reference image. Image registration is performed between the original LSC image and the reference image displayed in (a). From the image registration, a transform matrix is obtained and used to transform the original LSC image (a) to the coordinate system of the reference image (c). The same transformation matrix is used on the adjusted LSC image (b), which is transformed to (d).

5.3.1 Image Transformation

The `imregister` function calculates the transformation matrix, which is then applied to all other images that need to be aligned with the reference image. This ensures that the original and adjusted images undergo the same transformation.

5.4 Binarization of Transformed Images

After aligning the images, the same binarization method used in section 4.1.2 will be applied. However, for the reference image, binarization is not necessary as we can directly utilize the image shown in Fig. 30 (a).

5.5 Contour Extraction

In the binarized images, the algorithm specified in section 4.3 can be applied to detect the contours of the lizard in the original, adjusted, and reference images. The algorithm requires the center line pixels to determine the starting point for scanning the end of the lizard's body. The center line was already obtained in section 4.2.1. This image containing the center line is used again, but this time it is transformed to different coordinates using the same transformation process applied to the original and adjusted LSC images. The transformed center line image is then binarized once more to address blurriness caused by the transformation. After binarization, only the unique white pixels in each row of the image are selected as center pixels. The final center line is displayed in Fig. 33.

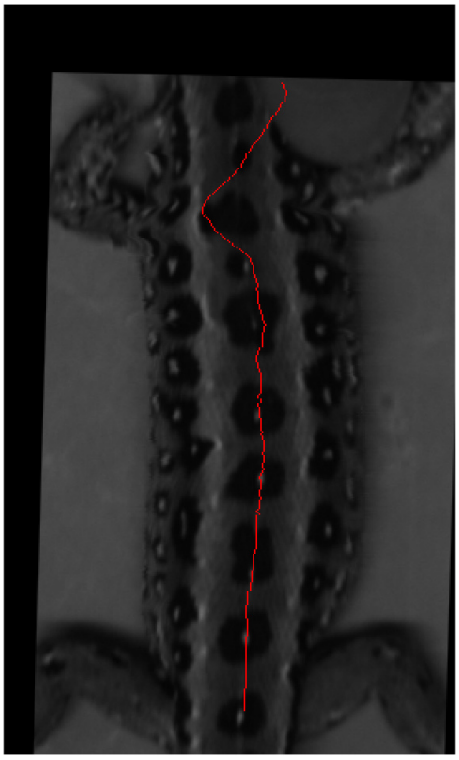
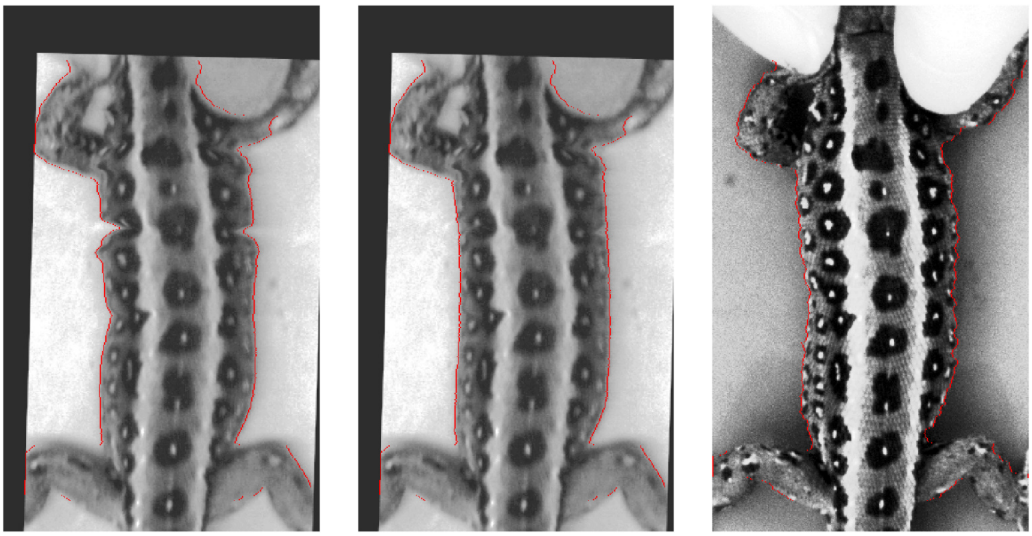


Fig. 33: Transformed center line (red) in the original transformed grayscale LSC image.

Using this center line, it is possible to obtain contours for all three images. The detected contours are displayed in the histogram-equalized images in Fig. 34.



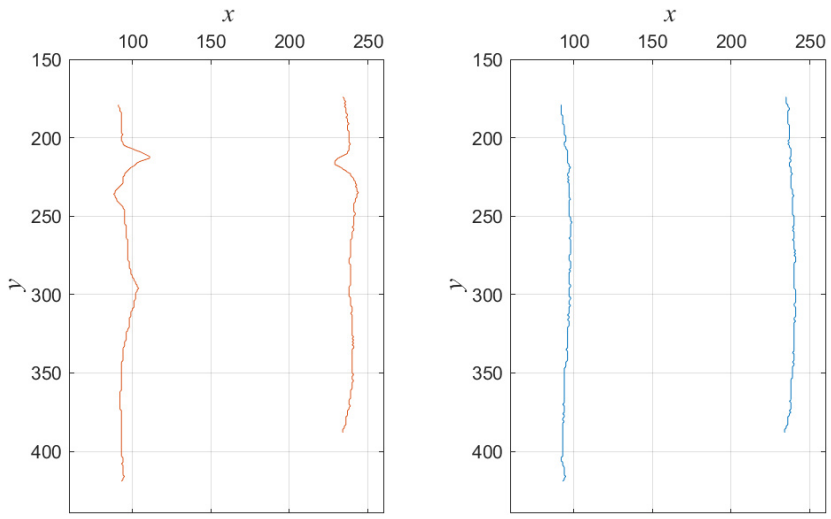
(a) (b) (c)

Fig. 34: Contour detection on transformed images and the reference image. The contours are highlighted in red, where (a) displays the contours of the original transformed LSC image, (b) displays the contours of the adjusted transformed LSC image, and (c) displays the contours of the reference image. All images are histogram equalized for demonstration purposes.

5.5.1 Finding Trunk Contours for Comparison

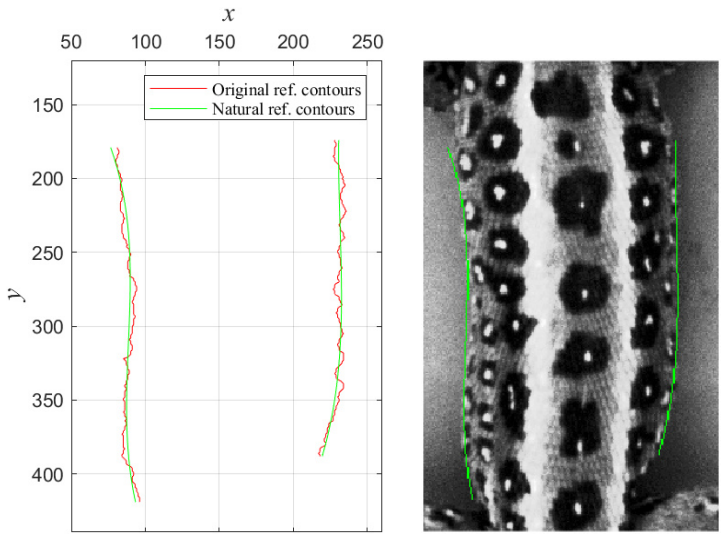
From the edges detected in the previous step, the trunk area of the lizard's body where the adjustments were made needs to be identified. This is done using the same method as described in section 4.3.1. The method is applied to the contours of the referemce image and the contours of the transformed adjusted LSC image. The starting and ending indexes are then selected for the left and right contours, and used for all extracted contours. The resulting contours, which will be compared, are displayed in Fig. 35.

The contours in the reference image are quite bumpy, so a third-order polynomial is calculated and used for the final comparison. The resulting contours of the lizard in the reference image are displayed in Fig. 36.



(a) (b)

Fig. 35: Contours of transformed LSC images. Graph (a) shows original transformed LSC image contours without elimanted leg pixels. Graph (a) shows the same for the adjusted transformed LSC image.



(a) (b)

Fig. 36: Estimating natural contours of the reference image. Graph (a) shows the detected original contours (red) and estimated natural contours (green) of the reference image. The estimated natural contours (green) are highlighted in the histogram equalized reference image in (b).

5.6 Evaluation

Now, with contours available for all three images, the evaluation can begin by calculating the difference between the contours of LSC images and the contours of the reference image. Initially, the alignment of these contours is necessary because the reference contours may be slightly shifted after the polynomial adjustment. This alignment is performed to minimize the difference between the LSC and the reference image contours. The adjusted LSC image is used for alignment to avoid potential issues caused by large distortions in the original image, which could result in an incorrect shift calculation. The shift is only performed on the x-axis, and the optimization algorithm 5 is used to determine the $good\_shift$.

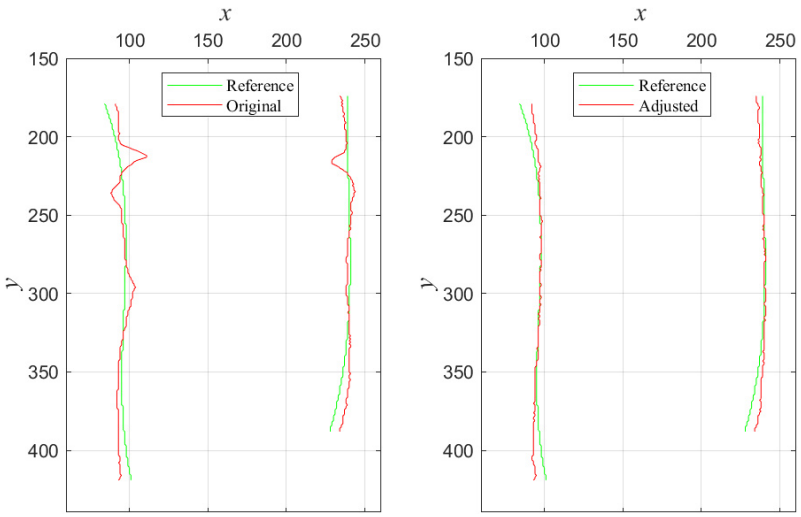
Algorithm 5 Finding shift for good alignment of contours.

$min\_diff \leftarrow inf$ ▷ minimal difference, so far set to infinity
for $shift = -10 : 10$ **do**
  $LAR\_contour \leftarrow \mathcal{C}\{2, 1\}$ ▷ where $\mathcal{C}$ is a set of all contours
  $RAR\_contour \leftarrow \mathcal{C}\{2, 2\}$
  $diff\_L \leftarrow \text{abs}(L\_ref\_contour(:, 2) + shift - LAR\_contour(:, 2))$
    ▷ where $LARcontour$ is Left Adjusted Registered image contour
    ▷ and $L\_ref\_contour$ is Left reference image contour
  $diff\_R \leftarrow \text{abs}(R\_ref\_contour(:, 2) + shift - RAR\_contour(:, 2))$
    ▷ same as before but for the right side
  $total\_diff \leftarrow \text{sum}([diff\_L; diff\_R])$ ▷ sum of left and right difference
  if $total\_diff < min\_diff$ **then** ▷ if new minimum found, variables change
    $min\_diff \leftarrow total\_diff$
    $good\_shift \leftarrow shift$
  end if
end for

Contours that are aligned with the $good\_shift$ are displayed in Fig. 37. The primary objective of this evaluation is to compare, if the difference between the adjusted LSC image and reference image to the one between the original LSC image and the reference image was minimized, and determine the extent of the difference reduction. For the final calculation of the difference, the algorithm 6 was developed, based on algorithm 5.

Algorithm 6 Calculating difference.

$final\_diff = []$ ▷ Initialization of an array to store the differences
for $img = 1 : 2$ **do**
  $LAR\_contour \leftarrow \mathcal{C}\{img, 1\};$ ▷ where $\mathcal{C}$ is a set of all contours
  $RAR\_contour \leftarrow \mathcal{C}\{img, 2\};$
  $diff\_L \leftarrow \text{abs}(L\_ref\_contour(:, 2) + good\_shift - LAR\_contour(:, 2))$
  $diff\_R \leftarrow \text{abs}(R\_ref\_contour(:, 2) + good\_shift - RAR\_contour(:, 2))$
  $final\_diff(img) \leftarrow \text{sum}([diff\_L; diff\_R])$
end for



(a) (b)

Fig. 37: Comparison of LSC image contours with the reference contours. Graph (a) shows a comparison of the original transformed LSC image contours with the estimated natural reference image contours. Graph (b) shows a comparison of the adjusted transformed LSC image contours with the estimated natural reference image contours.

5.7 Results

For the given dataset consisting of 12 LSC images and folders containing reference images, the compensation algorithm was evaluated. The results are documented in Table 1. This table includes the names of the LSC images, the ending number of the reference images chosen for evaluation, the difference between the original LSC images and the reference images, the difference between the adjusted LSC images and the reference images, the improvement results in pixels (px), and the corresponding percentages of improvements. The table highlights the results for the

example used to explain the compensation algorithm (Chapter 4) and the evaluation algorithm (Chapter 5) with a green colored row.

Tab. 1: Evaluation results

LSC Image Name	Reference Image Number	Original Diff. [px]	Adjusted Diff. [px]	Improved by [px]	Improved by [%]
L3ADORSAL1	150436289_0100	774	475	299	38.63
L3ADORSAL2	151429857_0100	1467	1391	76	5.18
L3ADORSAL3	151640816_0100	914	725	189	20.68
L3ADORSAL4	151930772_0100	2949	2807	142	4.82
L3ADORSAL5	152427143_0100	3488	3437	51	1.46
L3ADORSAL6	152714513_0100	3077	2999	78	2.53
L3ADORSAL7	152943185_0100	898	667	231	25.72
L4ADORSAL1	143650760_0100	1642	1186	456	27.77
L4ADORSAL2	144509234_0250	1469	846	623	42.41
L4ADORSAL3	153615062_0170	1859	1564	295	15.87
L4ADORSAL4	153912623_0100	1646	1079	567	34.45
L4ADORSAL5	154150495_0210	2027	1377	650	32.07

6 Discussion

When analyzing the results in Tab. 1, it was observed that all of the adjusted images showed an improvement in the contours of the lizard to some extent. The smaller improvements were noticeable in the images that did not exhibit severe breathing distortions in the trunk area of the lizard, which mainly included the LSC images starting with L3ADORSALx. These images corresponded to a type of lizard that displayed a calmer behavior during the scanning process and did not require physical restraint. In contrast, the LSC images starting with L4ADORSALx showed significant improvements in their trunk contours. These images were associated with a more active lizard during the scanning process, resulting in substantial contour deformations in the captured images (refer to Fig. 10).

What can be seen in the evaluation results (Tab. 1) is, that the percentage values of improvement vary significantly for each image. The evaluation algorithm relies heavily on the choice of the reference image for comparison. If the lizard in the reference image is not in the normal state of the respiration cycle (see Fig. 9), the contours can differ significantly from those of the LSC images. Consequently, the difference between the LSC images and the reference image will generally be larger, and the percentage improvement will be less significant. Thus, selecting an appropriate reference image is a crucial aspect of the evaluation method, which is why a shift of the contours is calculated in section 5.6 to minimize the overall difference between the LSC images and the reference image.

Proper image registration is another critical factor in the evaluation process. If the images are not correctly registered, the entire evaluation is compromised since the lizards' bodies would not be properly aligned, and their sizes may differ. Additionally, the selection of contours for comparison requires careful consideration. The polynomial crossover and algorithm 2 are used to isolate only the trunk area of the lizard for adjustment and comparison. While the polynomial crossover works well for all images, the algorithm based on standard deviation has some limitations. One disadvantage is that it may not accurately identify the trunk area if there is distortion near the legs. Similarly, if the lizard in the image is not perpendicular to the x-axis, the algorithm may struggle to distinguish between the trunk and legs. Although these methods have worked well for the given dataset, there could be scenarios where they may not adequately detect the trunk area of the lizards.

When addressing the compesation, there are multiple ways to compensate for image distortion or, in other words, restore the image. One method that comes to mind is the use of Deep Neural Networks (DNNs). DNNs can be trained to restore or inpaint the affected areas in an image, but their effectiveness heavily relies on the quantity and quality of training data. Examples of how this approach can work

are mentioned in [21, 22, 23]. Although this option was considered in this thesis, it was not employed due to a lack of training data and the importance of preserving as much original information about the lizard in the image as possible. Modifying the LSC image to an extent where the adjusted information no longer reflects the scanned lizard features was undesirable.

Due to this consideration, other compensation methods such as intensity-based image registration were not pursued. Intensity-based image registration was used in this thesis, specifically in section 5.3, to align the LSC images with the reference image. This registration process transforms the image based on the similarity of pixel intensities and aligns them to achieve the highest similarity. Currently, this algorithm only modifies the entire image through rotation and scaling. If this algorithm were extended to also modify certain areas and interpolate the lizard's body to its natural look, it would be theoretically possible to compensate for distortions. However, this is not a sure thing, and due to the importance of keeping as much information untouched, this research led to a simplified interpolation algorithm.

The interpolation method developed in this thesis allows us to store the modifications or reverse them, which is useful as it provides information about the adjustments made to the original LSC image. Currently, it produces visually pleasing lizard images. However, it is necessary to evaluate these results with the assistance of a specialist since the proposed evaluation method only considers the contours and does not include the evaluation of reconstructed back patterns and other elements that may be analyzed in these images.

It is important to note that while the compensation is specifically targeted between the front and back legs of the lizard, breathing distortions can also be observed in other areas of the lizard's body, such as between its legs, as depicted in the Fig. 38. The current method of compensation, developed in this thesis, cannot

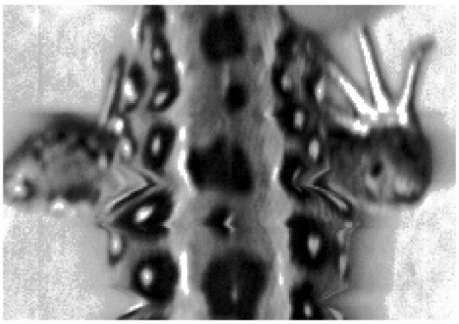


Fig. 38: Distortions between the legs in the equalized grayscale LSC image.

be used for distortions located between the legs as it relies on the lizard's contours. Compensation for distortions between the lizard's legs is generally challenging. If the interpolation method was employed, it would be necessary to determine where

and by how much the lizard's body moved in order to establish the interpolation ranges and perform the interpolation accordingly.

Another thing that is important to note is that the compensation and evaluation algorithm described in this thesis can successfully identify the trunk area, compensate for distortions in this area, and evaluate the compensation within the given dataset of images. While it is possible that it could work for other sets of images, it is not optimized for such scenarios and should be enhanced by incorporating additional checks and making each compensation and evaluation step more robust.

7 Conclusion

The first goal of this thesis was to study line scan cameras and digital image restoration, with a focus on compensating for motion distortions. The description of how line scan cameras work, why they are used in this thesis, and what are the resulting problems that can happen when scanning a living objects are all described in Chapter 2. For digital image restoration an algorithmic method was chosen, which led to a detailed analysis of how the lizard's body moves and how this movement can be compensated. More specifically, the interpolation method was used to adjust the information in the image to better represent a lizards body. Even though there are many ways the lizard can move when scanning, this thesis only addresses the compensation of distortions created by the lizard's breathing movement.

Based on the analysis of the lizard's respiration cycle, a second goal of this thesis was fullfiled by developing a compensation method to address the distortions in the lizard's trunk area between the front and back pair of legs. The compensation algorithm relies on the lizard's body contours, which have saw-like edges in the captured LSC images. The algorithm enhances these contours to achieve a more natural appearance, resulting in a visually pleasing lizard image.

The final goal of this thesis was to develop an evaluation method for assessing the quality of image restoration. This was achieved through a comparison of the original and adjusted contours in the LSC images with the contours in the reference camera image, as well as by calculating the improvement in the adjusted contours.

The compensation and evaluation methods were tested on a given dataset, which comprised 12 LSC images and their corresponding reference images. The results presented in Table 1 show an improvement in the contours of all adjusted images, which validates the effectiveness of this compensation.

The methods proposed in this thesis are user-friendly, requiring only the specification of the path to load the LSC image and the selected reference image. These methods have been proven effective on the provided dataset of lizard images, and they are likely to work for other sets of similar lizard images with similar lighting conditions. However, it is important to note that they can be further improved as they are not specifically optimized for such scenarios.

BIBLIOGRAPHY

[1] Lanuza, G. P. i. de and Font, E. Ultraviolet vision in lacertid lizards: evidence from retinal structure, eye transmittance, SWS1 visual pigment genes and behaviour. *Journal of Experimental Biology*. august 2014, vol. 217, no. 16, p. 2899–2909, [cit. 2023-04-23]. DOI: 10.1242/jeb.104281. ISSN 0022-0949. Available at: https://doi.org/10.1242/jeb.104281.

[2] Lucasbosch. *English: Comparison of four multispectral camera setups* [online]. February 2021 [cit. 2023-04-23]. Available at: https://commons.wikimedia.org/wiki/File:Multispectral_imaging_approaches.svg.

[3] *Resonon Pika UV | Hyperspectral Imaging Camera | Software* [online]. © 2023 Resonon Inc. [cit. 2023-04-23]. Available at: https://resonon.com/Pika-UV.

[4] *Basler ace acA2500-60um - Area Scan Camera* [online]. ©2023 Basler AG [cit. 2023-04-23]. Available at: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2500-60um/.

[5] *LM12SC discontinued, build to order* [online]. ©2023 Kowa Optimed [cit. 2023-04-23]. Available at: https://www.kowa-lenses.com/en/lm12sc-6mp-c-mount-lens.

[6] *Optical resolution of line scan cameras* [online]. ©2023 [cit. 2023-04-23]. Available at: https://www.sukhamburg.com/support/technotes/linescancamera/basics/opticalresolution.html.

[7] *Lizard Reptile Snake Turtle Insect, Fields lizard lizard, animals, scaled Reptile png* [online]. [cit. 2023-04-23]. Available at: https://www.pngegg.com/en/png-bojlz.

[8] *Lizard free icons designed by Good Ware* [online]. [cit. 2023-04-23]. Available at: https://www.flaticon.com/free-icon/lizard_2683090.

[9] Sauvola, J. and Pietikäinen, M. Adaptive document image binarization. *Pattern Recognition*. february 2000, vol. 33, no. 2, p. 225–236, [cit. 2023-04-23]. DOI: 10.1016/S0031-3203(99)00055-2. ISSN 0031-3203. Available at: https://www.sciencedirect.com/science/article/pii/S0031320399000552.

[10] *Adjust Image Contrast Using Histogram Equalization - MATLAB & Simulink* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-23]. Available at: https://www.mathworks.com/help/images/histogram-equalization.html.

[11] Soille, P. *Morphological Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999 [cit. 2023-05-24]. ISBN 978-3-662-03941-0 978-3-662-03939-7. Available at: http://link.springer.com/10.1007/978-3-662-03939-7.

[12] Hildreth, E. and Marr, D. Theory of edge detection. *Proceedings of Royal Society of London*. 1980, vol. 207, 187-217, p. 9, [cit. 2023-05-15]. Available at: http://www.hms.harvard.edu/bss/neuro/bornlab/qmbc/beta/day4/marr-hildreth-edge-prsl1980.pdf.

[13] *Morphological structuring element - MATLAB* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-08]. Available at: https://www.mathworks.com/help/images/ref/strel.html.

[14] Han, D. Comparison of Commonly Used Image Interpolation Methods. In:. Atlantis Press, March 2013, p. 1556–1559 [cit. 2023-05-15]. DOI: 10.2991/iccsee.2013.391. ISBN 978-90-78677-61-1. ISSN: 1951-6851. Available at: https://www.atlantis-press.com/proceedings/iccsee-13/4822.

[15] Klein, S., Staring, M., Murphy, K., Viergever, M. A. and Pluim, J. P. W. Elastix: A Toolbox for Intensity-Based Medical Image Registration. *IEEE Transactions on Medical Imaging*. january 2010, vol. 29, no. 1, p. 196–205, [cit. 2023-05-15]. DOI: 10.1109/TMI.2009.2035616. ISSN 1558-254X. Conference Name: IEEE Transactions on Medical Imaging.

[16] *Levenberg–Marquardt algorithm*. February 2023 [cit. 2023-05-16]. Page Version ID: 1142022327. Available at: https://en.wikipedia.org/w/index.php?title=Levenberg%E2%80%93Marquardt_algorithm&oldid=1142022327.

[17] *MATLAB - MathWorks* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-07]. Available at: https://www.mathworks.com/products/matlab.html.

[18] *Image Processing Toolbox* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-07]. Available at: https://www.mathworks.com/products/image.html.

[19] *Intensity-Based Automatic Image Registration - MATLAB & Simulink* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-16]. Available at: https://www.mathworks.com/help/images/intensity-based-automatic-image-registration.html.

[20] *Create an Optimizer and Metric for Intensity-Based Image Registration - MATLAB & Simulink* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-16]. Available at: https://www.mathworks.com/help/images/create-an-optimizer-and-metric-for-intensity-based-image-registration.html.

[21] Wan, Z., Zhang, B., Chen, D., Zhang, P., Chen, D. et al. *Old Photo Restoration via Deep Latent Space Translation*. arXiv, september 2020 [cit. 2023-05-25]. DOI: 10.48550/arXiv.2009.07047. ArXiv:2009.07047 [cs]. Available at: http://arxiv.org/abs/2009.07047.

[22] Liu, H., Wan, Z., Huang, W., Song, Y., Han, X. et al. *PD-GAN: Probabilistic Diverse GAN for Image Inpainting*. arXiv, may 2021 [cit. 2023-05-25]. DOI: 10.48550/arXiv.2105.02201. ArXiv:2105.02201 [cs]. Available at: http://arxiv.org/abs/2105.02201.

[23] Liu, H., Wan, Z., Huang, W., Song, Y., Han, X. et al. *DeFLOCNet: Deep Image Editing via Flexible Low-level Controls*. arXiv, march 2021 [cit. 2023-05-25]. DOI: 10.48550/arXiv.2103.12723. ArXiv:2103.12723 [cs]. Available at: http://arxiv.org/abs/2103.12723.

[24] *Types of Morphological Operations - MATLAB & Simulink* [online]. ©1994-2023 The MathWorks, Inc. [cit. 2023-05-23]. Available at: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html.

SYMBOLS AND ABBREVIATIONS

CCD	Charge-Coupled Device
CDF	Cumulative Distribution Function
CMOS	Complementary Metal Oxide Semiconductor
DNN	Deep Neural Network
LoG	Laplacian of Gaussian
LSC	Line Scan Camera
MI	Mutual Information
NCC	Normalized Cross-Correlation
NUV	Near Ultraviolet
RGB	Red, Green and Blue
SSD	Sum of Squared Differences
UV	Ultraviolet

LIST OF FIGURES

LIST OF APPENDICES

Source_2023_DP_Szabó_Michal_209479.zip - archive containing:

- compensation.m - MATLAB script for compensation of distortions
- evaluation.m - MATLAB script for evaluation of compensation
- Template_curve.mat - calculated polynomial curve matrix
- getContourCurves.m - function for calculation of natural contours
- imregister2.m - adjusted MATLAB function to get transformation matrix