Introduction 3
Results 5
Chunking videos into usable segments 6
Chromatophore segmentation and color classification 7
Compensating for movement: Registration 9
Stitching across video chunks 10
Tracking chromatophore size 11
Tracking anisotropic chromatophore activity 12
Long-term chromatophore tracking 15
Built-in analysis tools and additional features 16
Performance 18
Discussion 19
Scalability 20
Usability 20
Known limitation 21
Methods 21
Implementation details 21
Training machine learning models to segment chromatophores 22
Data Augmentation 22
Models 22
Registration details 24
Stitching reprojection error 24
Clustering motion correlation data 24
Data availability 26
Acknowledgments 26

CHROMAS: A Computational Pipeline to Track Chromatophores and Analyze their Dynamics

Johann Ukrow#, Mathieu D. M. Renard#, Mahyar Moghimi, Gilles Laurent*
Max Planck Institute for Brain Research, Frankfurt, Germany

#joint first authors
*corresponding author.

Abstract:
Cephalopod chromatophores are small dermal neuromuscular organs, each consisting of a pigment-containing cell and 10-20 surrounding radial muscles. Their expansions and contractions, controlled and coordinated by the brain, are used to modify the animal's appearance during camouflaging and signaling. Building up on tools developed by this lab, we propose a flexible computational pipeline to track and analyze chromatophore dynamics from high-resolution videos of behaving cephalopods. This suite of functions, which we call CHROMAS, segments and classifies individual chromatophores, compensates for animal movements and skin deformations, thus enabling precise and parallel measurements of chromatophore dynamics and long-term tracking over development. A high-resolution tool for the analysis of chromatophore deformations during behavior reveals details of their motor control and thus, their likely innervation. When applied to many chromatophores simultaneously and combined with statistical and clustering tools, this analysis reveals the complex and distributed nature of the chromatophore motor units. We apply CHROMAS to the skins of the bobtail squid *Euprymna berryi* and the European cuttlefish *Sepia officinalis*, illustrating its performance with species with widely different chromatophore densities and patterning behaviors. More generally, CHROMAS offers many flexible and easily reconfigured tools to quantify the dynamics of pixelated biological patterns.

Introduction

Animal skin color can change adaptively thanks to the migration or direct control of specialized pigment-containing cells. In ectothermic animals, these cells are called chromatophores. They often each contain one of several pigments, enabling elaborate patterns, colors (as in fish or reptiles) and in some cases, such as cephalopods, communication and camouflage. Coleoids cephalopods evolved the unique ability to change in an instant the patterning, color and 3D texture of their skin through the neural control of their chromatophores. Cephalopod chromatophores are elaborate neuromuscular organs composed of a cell containing an elastic sacculus filled with pigment granules, surrounded by a set of radial muscles, themselves controlled by motor neurons located in the brain (Clooney & Florey, 1968; Messenger, 2001). The skin of some species of cuttlefish and octopus possesses millions of such chromatophores at a density of 230 per mm^2 in adults. As biological analogs of pixels on a screen, chromatophores operate by varying their size (and thus effective reflectance) rather than photon emission. Chromatophores, together with other specialized cells such as iridophores and leucophores, enable these cephalopods to display a variety of patterns and colours that play crucial roles for camouflage and intra- and inter-specific signaling (Hanlon & Messenger, 1988).

The cephalopod camouflage system operates as a perception-processing-projection pipeline, where visual input from the environment is analyzed by the nervous system and reproduced as a statistical approximation on its skin. For this reason, these skin displays offer a quantifiable readout of the animal's perceptual state and have fascinated scientists for decades. However, studying chromatophore activity over chromatophore populations is challenging because of their small size (ranging from 5 to 100 µm when retracted, depending on the species), their vast numbers (thousands to millions) over large surfaces (hundreds of cm^2), the inherent flexibility and motion of cephalopod skin (due to respiration, for example), and the absence of tools to track their state in real time. While researchers did, in a few studies, quantify skin patterns from still images of the animals, chromatophore resolution and tracking of patterns over time were not achieved (Barbosa et al., 2008; Hanlon et al., 2009; Shohet et al., 2007). Physiologists, however, did study individual or a few chromatophores in action, but this approach rested heavily on non-automated

measurements, with limited precision and scalability (Packard, 1982), and did not address the issue of camouflage directly.

This laboratory recently developed automated and quantitative approaches to circumvent these limitations to study cephalopod camouflage and related behaviors (Reiter et al., 2018; Woo, Liang et al., 2023). The tools developed there have been re-written and expanded in an easy-to-use software suite, which we call CHROMAS, the object of the present paper. CHROMAS is a high-resolution analysis software that aims to provide an objective quantification of chromatophore-population behavior over periods of time as long as months, using video recordings of live cephalopod skin as an input. While CHROMAS retains the foundational concepts introduced in Reiter et al., 2018 and Woo, Liang et al., 2023, the tools have been rewritten to improve performance and usability. New functionalities have been added to extend the temporal range and spatial resolution of the analysis. In particular, we introduce the semi-automated tracking of chromatophore identity over development and the segmentation of individual chromatophores by analysis of deformation anisotropy. Although developed to track and analyze cephalopod skin patterning, these tools can be used with the skin of other species, and more generally, with any comparable time series of images.

CHROMAS reads videos files and extracts chromatophores properties based on known biological features, derived from prior experimental work:

Identity and position: The relative spatial arrangement of chromatophores is anatomically fixed in the skin (Packard, 1982) because they are anchored to a stable and appropriately stiff extracellular matrix. Their neighboring chromatophores, however, change as the animal grows, because new chromatophores are continuously added between older ones, and change color as they age (Reiter et al, 2018). CHROMAS keeps track of chromatophore identity over time despite these changes.

Expansion state: When radial muscles (attached distally to the extracellular matrix and proximally to a chromatophore) contract, the chromatophore's cytoplasmic membrane and internal pigment sack expand; inversely, when those muscles relax, the pigment sack retracts passively due to its elasticity. Because the chromatophore muscles are controlled by motor neurons (Florey & Kriebel, 1969), chromatophore

size variations offer an indirect but objective and quantifiable readout of neural activity (Reiter et al, 2018) at video sampling rate.

Anisotropy of expansion and detailed innervation: The radial muscles controlling a chromatophore can collectively be innervated by more than one motor neuron (Florey, 1969). Chromatophore expansion is then determined by a set of potentially independent forces, which can result in irregular or anisotropic chromatophore deformations. Analysis of this anisotropy can thus inform us of the fine innervation of individual chromatophores. CHROMAS provides this information.

We hope that these tools will become valuable resources to address complex biological questions regarding chromatophore neural control, development, biomechanics and more generally, cephalopod perception and fine motor behavior. More broadly, these tools should be useful also for detailed moving-image analysis in other systems.

Results

CHROMAS provides a versatile framework to investigate a wide range of biological questions. Each tool can be used independently of the others to solve specific tasks, while their combined use enables the construction of elaborate workflows. The subsequent sections provide examples of specific tasks that, when combined sequentially, constitute a powerful workflow (see Figure 1), as demonstrated in Reiter et al., 2018 and Woo, Liang et al., 2023 with *Sepia officinalis*. Figures 1-8 illustrate the main functions with *Euprymna berryi*, because of its small size and limited number of chromatophores. All functions have been developed to work with larger species with greater chromatophore numbers and density, such as *Sepia officinalis*, as shown later in Fig. 9.

We provide a comprehensive tutorial guide along with sample video files to test the pipeline. Links to these resources are available in the ‘Data Availability’ section. The entire pipeline can be run with the command below, and individual task command lines are included as each step of the pipeline is detailed.

```
Command line:
> chromas run /path/to/sepia_example.mp4
```

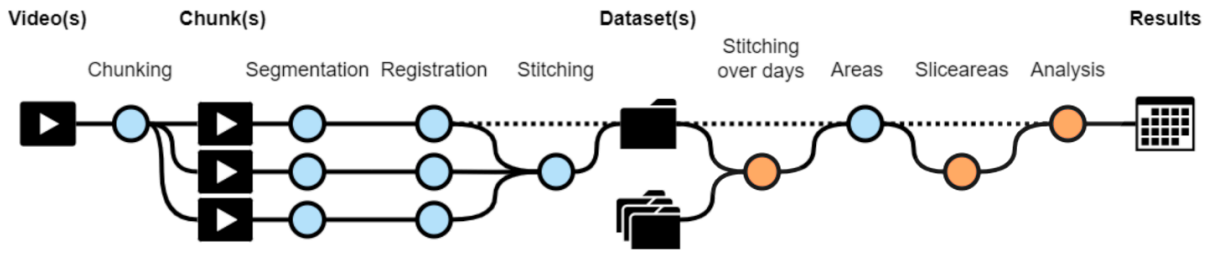

Figure 1: Workflow diagram of CHROMAS. Dotted lines indicate optional steps in the process, allowing flexibility depending on experimental requirements. Orange dots represent new functions that did not exist in the initial versions of the pipeline (Reiter et al., 2018; Woo, Liang et al., 2023). Each step of the pipeline can be run independently, providing users with the flexibility to execute specific stages as needed. Additionally, CHROMAS includes options to generate visual output videos, enabling users to assess and validate the accuracy and quality of the result at every step.

Chunking videos into usable segments

Analyzing long video recordings of behaving-cephalopod skin can be frustrating because of occasional movement blur, de-focusing, and obstruction. To address this, we divide the video into continuous segments which we call "chunks", consisting of consecutive frames where the animal's mantle, or a part of it, is both visible and in focus. An animal, or a field, is considered to be in focus if the edges of its chromatophores are clearly defined. The identification of these chunks can be accomplished using focus statistics, which are numerical measures indicative of the sharpness of an image (Fig. 2, focus score). A method often used is the difference of Gaussians, which involves subtracting a blurred version of an image (produced by convolving the image with a Gaussian kernel with a large standard deviation) from a less blurred version of the original image (using a smaller standard deviation). The resulting image emphasizes areas of steep intensity change, such as edges, while suppressing low-frequency variations, making it well-suited for detecting the sharp boundaries of chromatophores. The standard deviations of the two Gaussian kernels were chosen such that their difference matched the typical size of chromatophores, to increase the visibility of their edges. The input video is then cut into shorter clips — the chunks (Fig. 2, chunking output). This step can be adapted to exclude other types

of unwanted frames based on other parameters, such as low brightness, the absence of a fluorescent tag in the frame, or motion blur.

```
Command line:
> chromas chunk /path/to/sepia_example.mp4
```

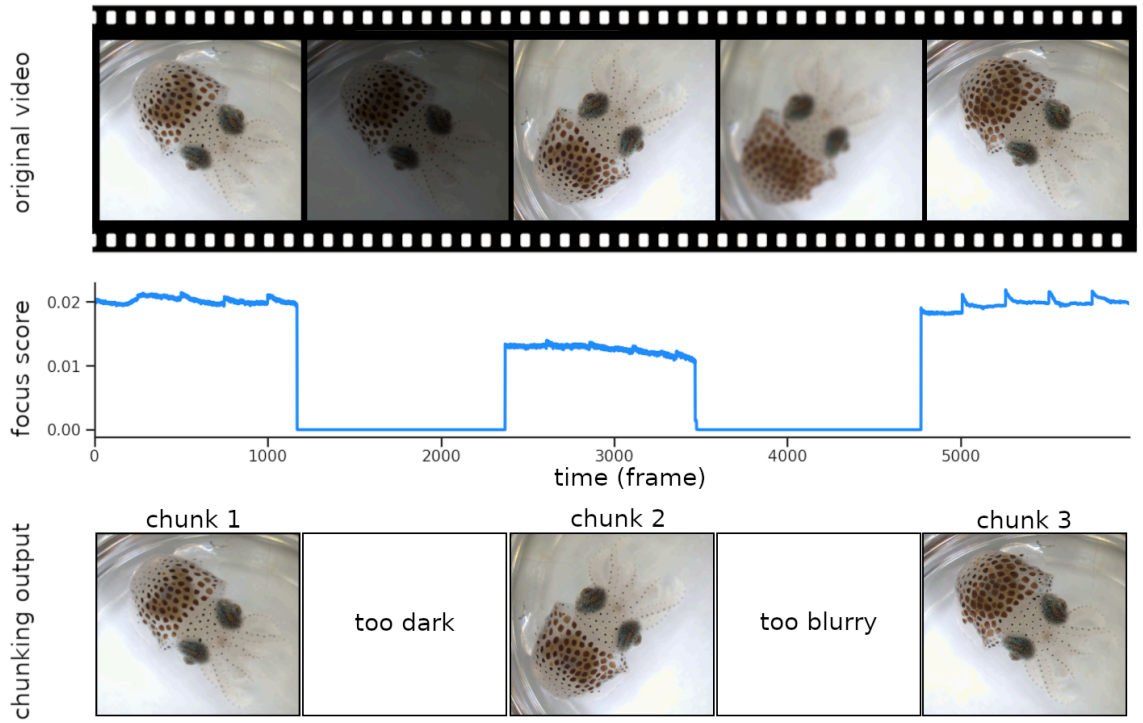

Figure 2: Chunking process. The video frames are sorted based on their focus score. Images that are too dark or too blurry are removed from the dataset, and chunks made of consecutive sharp frames are created. In the figure above, the score decreases due to a brightness drop at frame 1200 and to the subject going out of focus around frame 3500, resulting in three distinct chunks. Continuity between chunks is reestablished during the “Stitching” process (see Fig. 5).

Chromatophore segmentation and color classification

It is essential to identify and isolate individual chromatophores accurately in video frames before a detailed analysis can begin. "Segmentation" refers to the process of generating a binarized image, where each pixel is classified as either part of a chromatophore or part of the background (Fig. 3). Chromatophore classification by color is also supported. Segmentation relies on deep learning models trained on manually annotated data. We provide two trained models: one for binary (yes, no) and one for chromatophore-color segmentation (dark, orange, yellow), trained on annotated data of *Sepia officinalis* and *Euprymna berryi* under different lighting

conditions and camera setups. Furthermore, these models can be fine-tuned on custom data, and various models can be trained from scratch on custom data to enable research on different species. Available models include Fully Convolutional Networks (Long et al., 2015), DeepLabV3 (Chen, 2017) or U-Net (Ronneberger et al., 2015), with ResNet50, ResNet101 (He et al., 2016) or MobileNetv3-Large (Howard et al., 2019) serving as backbones. Alternatively, segmentation can also be achieved by a Random Forest Classifier or a combination of both, followed by “majority vote”.

```
Command line:
> chromas segment /path/to/sepia_example.dataset
```

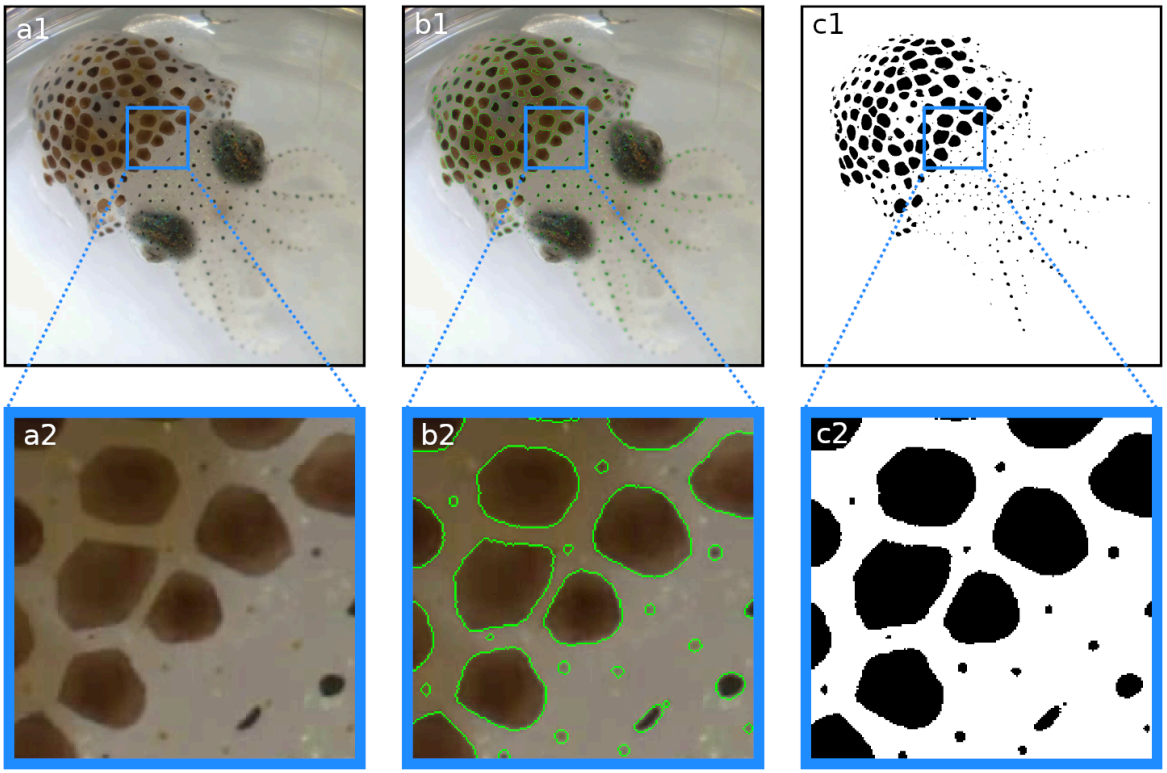

Figure 3: Segmentation results. In each frame, each pixel is classified as either part of a chromatophore or not. a1-2) Original frame before segmentation. b1-2) Original frames with edges of segmented chromatophores in green. c1-2) Binary segmentation with chromatophores in black and background in white. Note that even the smallest chromatophores, though faint and difficult to discern on the original footage, are segmented.

Compensating for movement: Registration

To measure a chromatophore's activity (deformations), its identity must be correctly transferred across frames; this is a difficult task if its coordinates change from frame to frame. Chunk selection ensures that video segments with blur due to rapid motion have been excluded from the data. However, chromatophores often change position from one frame to another due to slower or more subtle movements such as breathing, slow drifting, or skin deformation, while remaining in focus. Therefore, what we call the "Registration" operation compensates for the animal's movements to maintain a consistent location of individual chromatophores over time. For this, the Lucas-Kanade Optical Flow algorithm (Lucas & Kanade, 1981) tracks points that are initially randomly sampled on the first frame, from frame to frame. Full displacement maps are then interpolated from the displacement of these tracking points using a moving-least-squares algorithm. This results in all the frames being registered with the first frame of the video as reference (Fig. 4).

```
Command line:
> chromas register/path/to/sepia_example.dataset
```

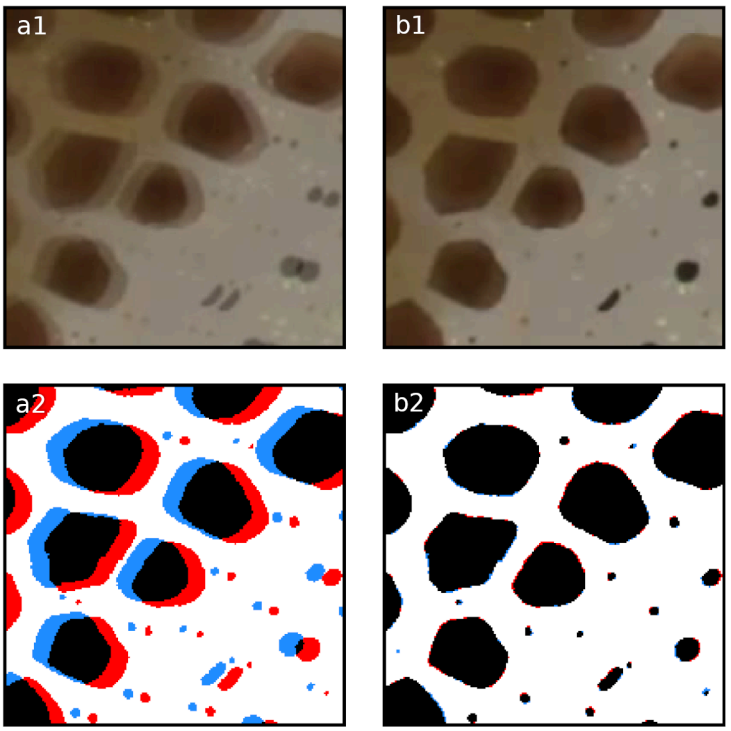

Figure 4: Registration. a) Overlay of two frames 20 frames apart within the same chunk. Panels (a1-a2) illustrate the misalignment of two chromatophores (red and blue) before registration. Panels (b1-b2) show the same frames after automated registration. The registration process compensates for small movements or drifts over time, ensuring correct identification and tracking of the chromatophores over time.

Stitching across video chunks

To create a cohesive dataset, the correct identity of each chromatophore must be tracked across chunks. The absence of visual information between chunks (by definition of the chunks), however, renders standard optical-flow techniques unsuitable for this task. A new technique was developed for this end involving the following steps. All registered frames within a chunk are averaged to produce a descriptive image of the position and color of each chromatophore. This average is referred to as the “masterframe” of that chunk (Fig. 5a, without the ellipse). If the animal’s body is fully visible, an initial alignment of the masterframes of successive chunks is achieved through automatic ellipse-fitting around the animal’s silhouette (Fig. 5a-c). Otherwise, alignment is performed after the user manually selects matching points between masterframes. Subsequently, displacement vectors are calculated for a regular grid of points sampled over the animal’s body by maximizing phase correlation between image patches surrounding these points (Fig. 5b-d). Full displacement maps are then interpolated using a moving-least-squares algorithm. The accuracy of these mappings is quantified using the reprojection error metric, as detailed in the methods section, and chunks with insufficient reprojection error are discarded from the analysis dataset.

```
Command line:
> chromas stitch /path/to/sepia_example.dataset
```

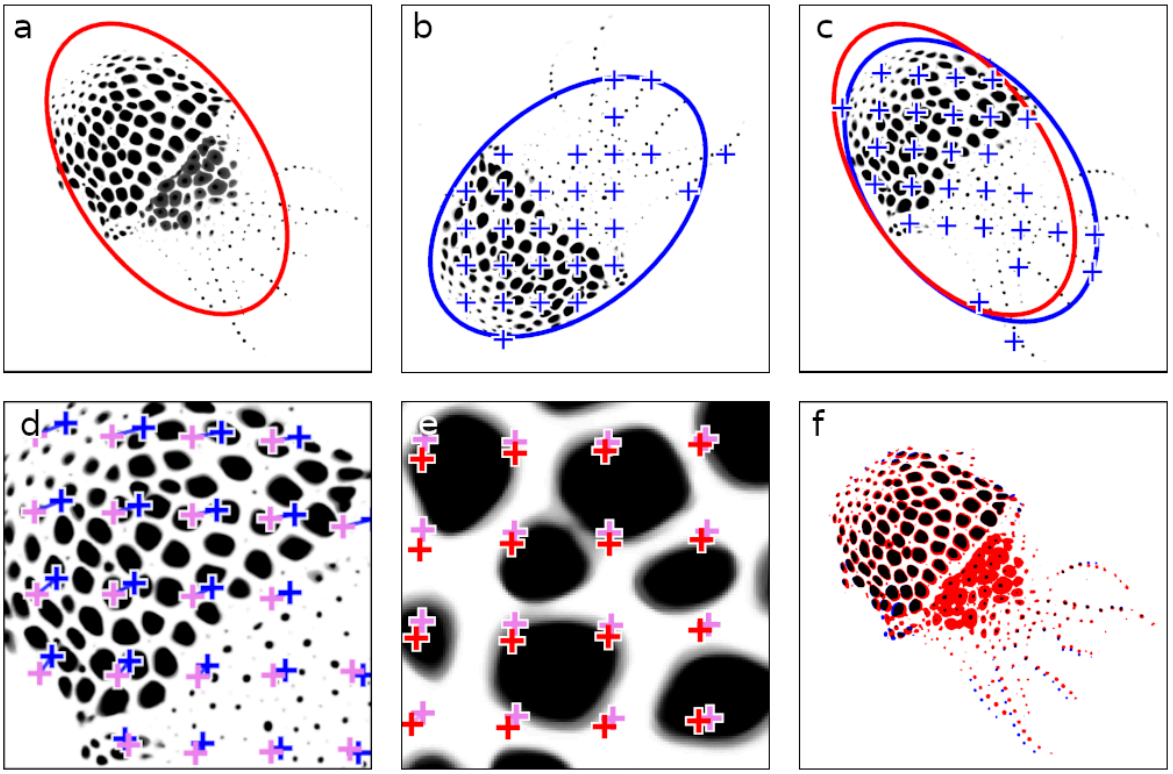

Figure 5: Stitching video chunks. This figure illustrates the stepwise refinement process for stitching adjacent video chunks, ensuring precise alignment critical for downstream analyses. a) Masterframe of chunk 1, with an initial ellipse fit shown in red. b) Masterframe of chunk 2, featuring the ellipse fit and a grid of points for fine alignment in blue. c) Chunk 2 aligned on chunk 1 using the ellipse-fitting method. d) First round of fine alignment, maximizing phase correlation between patches surrounding the points. Points prior to fine alignment are shown in blue, and their adjusted positions post-alignment are in pink. e) Second round of fine alignment, from pink to red. f) Final overlay between chunk 1 and the fully aligned chunk 2. In black are the chromatophore areas common to both chunks; in red and blue are chromatophore areas that are unique to chunk 1 and chunk 2, respectively.

Tracking chromatophore size

Now that the chromatophores have been aligned across frames and chunks, their identity can be tracked and their expansion quantified (Fig. 6a). First, a 2D array, termed “Cleanqueen”, is generated, that delineates individual chromatophore territories, i.e., the space that each chromatophore can occupy (Fig. 6b). Using inverse registration maps, the cleanqueen is mapped onto the unregistered segmentation frames and the identity of each chromatophore can be tracked. The area of each chromatophore is then calculated as the number of pixels forming the

largest connected component inside the chromatophore's corresponding territory (Fig. 6c).

```
Command line:
> chromas area /path/to/sepia_example.dataset
```

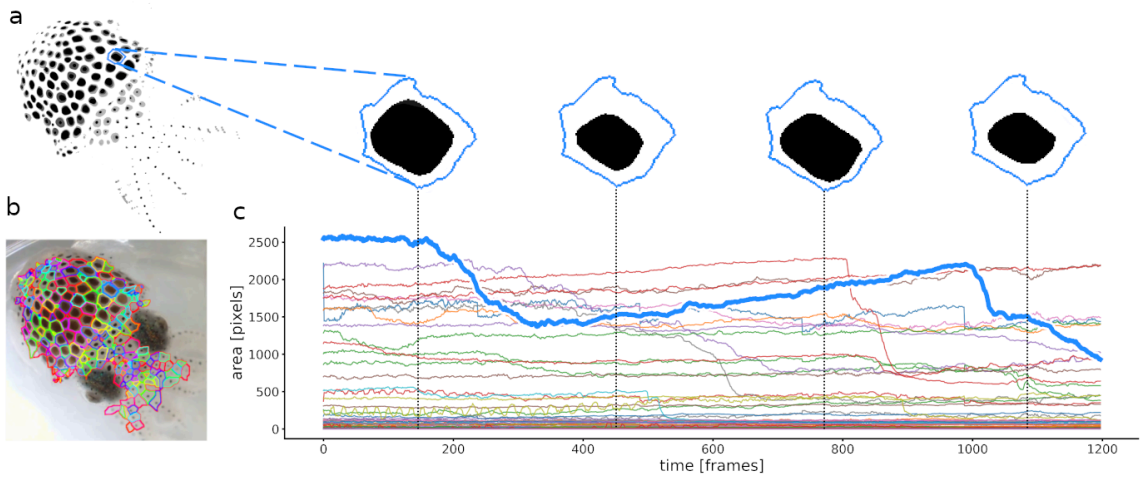

Figure 6: Calculation of chromatophores areas. a) Masterframe and zoom on the segmentation of one chromatophore's contractions and expansions. b) Cleanqueen. Each color represents a different chromatophore territory. c) Surface area of each segmented chromatophore over time (colors as in b). The chromatophore in a is highlighted in the thick blue line. This approach measures the total area occupied by a chromatophore within its defined territory, and offers no information about its shape.

Tracking anisotropic chromatophore activity

For most uses, analysis of the global chromatophore expansion is sufficient. There exist conditions, however, where a high-resolution analysis of chromatophore deformations is useful. We will show that it can reveal the multiple innervation of individual chromatophores and the fine and distributed nature of motor units.

Quantifying the kinematics of an ever-changing shape on a surface which is also subject to deformations is not a trivial task. Whereas chromatophores can be viewed as monochromatic objects that change shape over time, they reside on a surface (the skin) that itself undergoes both local and global deformations, such as mantle muscle contractions or movements caused by breathing, locomotion or external forces such as water currents. Therefore, an accurate description of the kinematics of a chromatophore can be done only once it has been disentangled from the distortions

of the skin. We thus addressed the need to accurately stabilize (i.e., compensate for the movements and deformations of) the background. Registration maps were not suitable for this task, because an optic flow approach would try to correct for chromatophore expansion. We thus developed a different type of registration. This step takes advantage of the fact that the 2D configuration of chromatophores in their most contracted state tends to be fixed in space. The center of mass of chromatophores that remain small and constant in size throughout the video (typically, chromatophores at early or intermediate stages of development, Reiter et al., 2018) can thus be viewed as inert points on the animal's skin, and their displacement over frames can then be used to derive skin deformation. We call these points "motion markers". Once this operation has been accomplished, the details of a chromatophore's deformations can be obtained.

To quantify these deformations, each chromatophore is divided into radial slices, and the area within each slice is calculated independently (Fig. 7a). To slice a chromatophore radially, its "epicenter" (the starting point of its expansion, blue dots in Fig. 7a-c), must be identified and tracked over frames. This point cannot be the center of mass of the chromatophore, because anisotropic deformations would wrongly shift this point between frames. Instead, a chromatophore's epicenter is computed following the logic of the previous paragraph: it is calculated as the center of mass at the frame where the chromatophore is in its most contracted state. Because the spatial relationships between epicenters is constant in time, their coordinates can be stored relative to the coordinates of their respective motion markers (Fig. 7c, green dots), i.e. the three nearest motion markers spanning a triangle around the chromatophore in a Delaunay triangulation (Fig. 7c, green triangles represent the Delaunay triangulation). Relative coordinates are stored as barycentric coordinates $(\lambda_1, \lambda_2, \lambda_3) \in R^3$, such that the absolute coordinates c of the epicenters can be calculated as $c = \lambda_1 \cdot a_1 + \lambda_2 \cdot a_2 + \lambda_3 \cdot a_3$ from the absolute coordinates a_1, a_2, a_3 of their motion markers. The combination of these local coordinate systems, along with a sufficiently dense distribution of motion markers, enables us to eliminate the global and local deformations of the skin and to successfully track the positions of all epicenters. In addition, the motion markers are

used to keep the orientation of the slices constant over frames, which is necessary to track the identity of each slice.

By default, chromatophores are sliced into 36 sectors. This number was selected based on the Nyquist-Shannon sampling theorem (Shannon, 1948), and on histological analysis showing that chromatophores in *Euprymna berryi* and *Sepia officinalis* are each controlled by approximately 10-15 radial muscles. This number of slices struck an acceptable balance, ensuring a high enough resolution to capture realistic shape changes, while keeping computational demands manageable. To reduce measurement errors induced by pixel discretization, slice areas were computed by averaging the distance from the border pixels of the slice to the epicenter *r* and then computing the respective area as $area = \pi \cdot r^2 / n$, see Fig 7b.

```
Command line:
> chromas slice /path/to/sepia_example.dataset
```

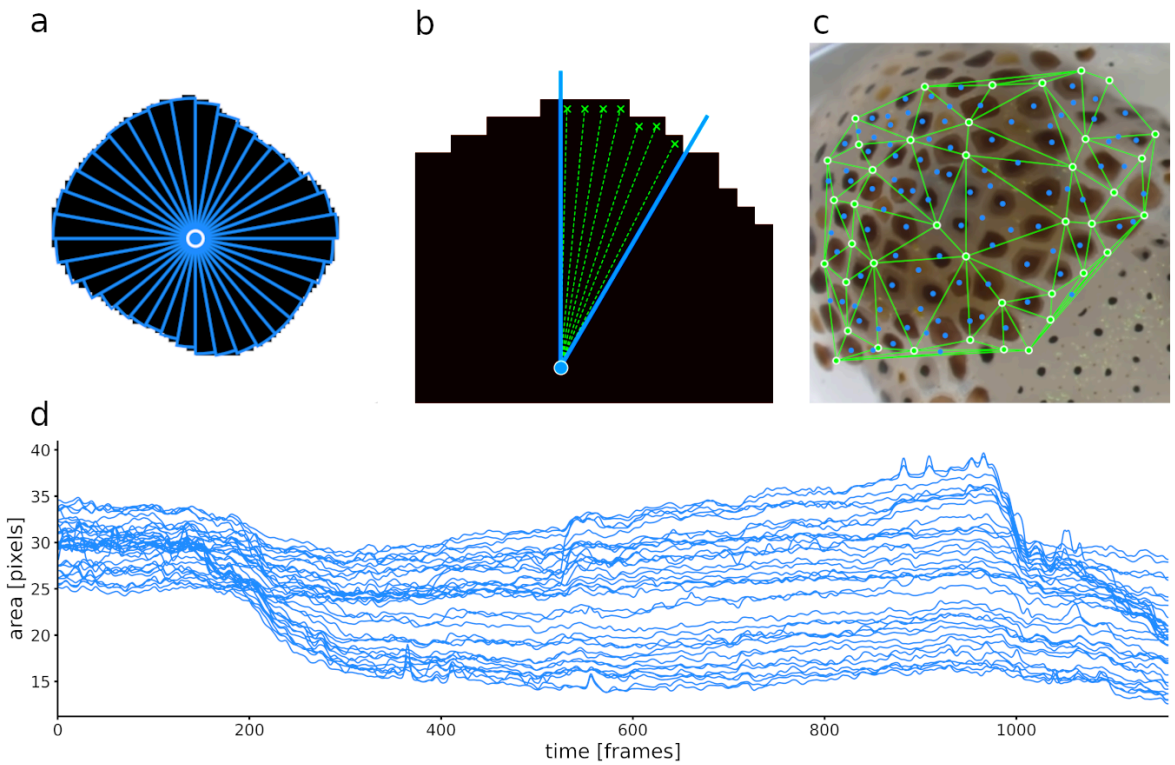

Figure 7: a) A single chromatophore (the same one as in fig. 6), is divided radially into 36 slices. b) Close-up on one of the 36 slices, indicating how its surface area is estimated. c) Chromatophore "epicenters" (see text) in blue and motion markers (see text) in green, with edges for triangulation. d)

Plot representing the surface areas of the 36 slices of this chromatophore over time. Note that the dynamics of a chromatophore are now described by 36 values per frame, rather than one, as in Fig 6c. Note also that the top half of the traces describing this chromatophore deviate clearly from the others around frames 550 and 960, indicating their differential control. This fine-grain description of chromatophore deformations over time is used to reveal the fine details of their individual and collective motor control.

Long-term chromatophore tracking

Tracking individual chromatophores over extended periods (days or weeks) is necessary to examine, among others, functional and developmental issues of control. While standard stitching techniques (see 'Stitching') suffice for datasets captured in close temporal proximity (hours), they fail for longer time periods, due to significant changes in animal size, chromatophores color, chromatophore numbers, and occasionally the disappearance of existing ones.

To address this, CHROMAS includes an option for a manual pre-alignment step prior to automated stitching. This process is facilitated by an intuitive graphical interface (based on *matplotlib* and *tkinter*) allowing users to select corresponding points between two frames from videos captured at different stages. The interface has been developed with usability in mind, providing options such as zoom, rotation and an "undo" function. These selected points are then utilized to generate warping maps using either the moving-least-squares algorithm or thin plate splines (*NumPy* and *SciPy*), providing a precise initial alignment that significantly improves the continuity and accuracy of subsequent stitching and analysis (Fig. 8).

```
Command line:
> chromas superstitch /path/to/sepia_example.dataset
/path/to/later_sepia_example.dataset
```

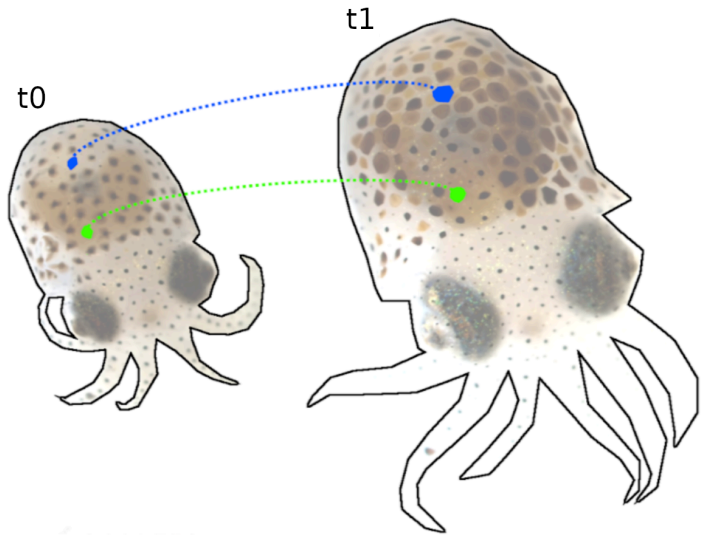

Figure 8: Tracking of chromatophores over 12 days of development (t_0 = 7 days post hatching (dph) and t_1 = 21 dph). During this time, the number of chromatophores nearly doubled, with many changing chromatic properties. Despite these changes and intermittent filming, our pipeline reliably tracks individual chromatophores, as illustrated with two arbitrarily selected chromatophores, in blue and green.

Built-in analysis tools and additional features

CHROMAS also offers a series of built-in analysis tools. Clustering algorithms, for example, are particularly useful to study how motor neurons control small groups of chromatophores, or sets of sectors of chromatophores, potentially revealing patterns of co-innervation and synchronized activity across specific regions of the skin. Our software enables users to perform Principal Component Analysis (PCA), Independent Component Analysis (ICA), and clustering analysis, including, but not limited to, Affinity Propagation (AP) and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). PCA and ICA are instrumental in disentangling the influence of motor neurons on individual chromatophores, while clustering algorithms uncover the broader distribution of motor neuron influence across larger scales (Fig. 9). Such tools can be found under “Methods - Clustering motion correlation”.

```
Command line:
> chromas analyze XXX /path/to/sepia_example.dataset
```

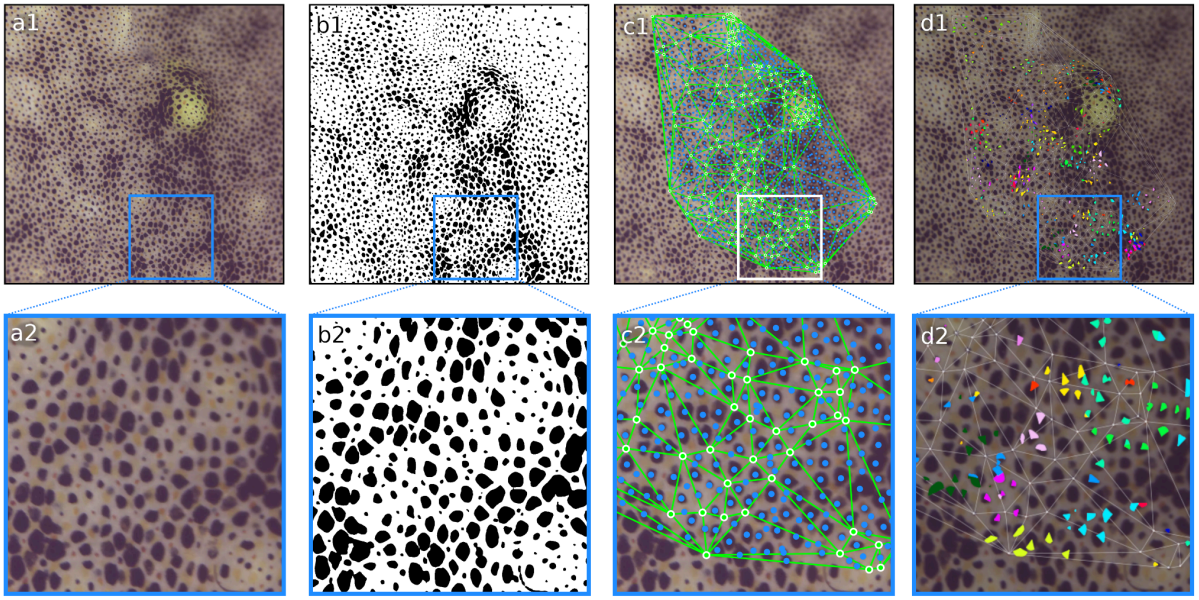

Figure 9: Example of 4K footage of *Sepia officinalis* processed through the CHROMAS pipeline. a1-2) Original video frame. b1-2) Binary segmentation with chromatophores in black and background in white. c1-2) Epicenters of chromatophores in blue and motion markers with triangulation in green. d1-2) Putative motor units calculated using the Affinity Propagation algorithm. In this example, PCA-ICA analysis revealed independent components influencing the activity of each chromatophore. We found 4.68 ± 1.41 (mean ± std) ICs per chromatophore (n=1616). Those independent components were then clustered together based on their covariation and plotted with the same color. For visualisation purposes, only clusters of size 8 and 9 are plotted. In this segment, we found 10.37 ± 14.23 (mean ± std) chromatophores per cluster, with 90% of clusters consisting of less than 20 chromatophores. This example highlights CHROMAS's ability to process complex datasets, such as footage of *Sepia officinalis*, which exhibits a high density of chromatophores and very active dynamics correlated with its camouflage and other behaviors.

Furthermore, CHROMAS integrates advanced tools for analyzing full-body patterns, enabling studies on camouflage effectiveness and pattern dynamics. The software employs the Structural Similarity Index (SSIM) to measure perceptual similarity between biological and background patterns, implemented using the *scikit-image* Python package. Patterns are embedded into a lower-dimensional "pattern space" using dimensionality reduction techniques like PCA and t-Distributed Stochastic Neighbor Embedding (t-SNE), facilitated by the *scikit-learn* and *openTSNE* libraries. Temporal dynamics of patterns are analyzed by plotting trajectories in this space using the *pandas* and *matplotlib* packages, revealing the speed, directionality, and stability of pattern transitions, as described in Woo, Liang et al. (2023).

Performance

Performance was benchmarked using a 5-minute-long 4K video (shot at 20 frames per second, totaling 6000 frames and 494.5 MB) featuring a close-up view of live skin of *S. officinalis*. The benchmark also included two modified versions of the same video, cropped to Full-HD (2000 × 2000) and SD (1000 × 1000) resolutions. Cropped videos preserve the original pixel density per chromatophore but include fewer chromatophores for analysis.

Table 1 and Table 2 document the processing time per frame (in milliseconds) for individual tasks, based on different image sizes and hardware configurations. The hardware categories include a workstation (with a 10 core Intel Core i9-10900X with 64 GB RAM and a NVIDIA RTX A4000 GPU), as well as a laptop (with a 4 core Intel Core i5-1135G7 with 16 GB RAM and no NVIDIA GPU).

While the runtime performance of all tasks (except for stitching) is influenced by hardware capabilities, video length, and resolution (in this particular order), the performance of areas and slice-areas calculation and analysis is also influenced by the number of visible chromatophores in the footage. Furthermore, the performance of stitching depends only on the hardware, the resolution and the number of chunks.

Resolution	Number of chromatophores	Segmentation (neural net)	Registration	Tracking chromatophore size	Tracking anisotropic activity
SD	146	5.38±0.1	18.61±0.1	13.01±1.4	4.98±0.9
HD	566	22.31±0.7	62.87±2.3	140.44±1.0	135.24±0.9
4k	2842	986.21±10.4	611.74±5.4	744.69±2.5	495.17±2.8

Table 1: Performance on a workstation with a 10 core Intel Core i9-10900X with 64 GB RAM and a NVIDIA RTX A4000 GPU. Runtime in milliseconds per frame (mean ± s.d.).

Resolution	Number of chromatophores	Segmentation (lookup)	Registration	Tracking chromatophore size	Tracking anisotropic activity
SD	146	11.55±1.81	4.67±0.63	14.56±5.1	9.31±0.6
4k	2842	1679.62±251.8	1490.94±249.9	3072.94±121.6	1488.25±84.0

Table 2: Performance on a laptop with a 4 core Intel Core i5-1135G7 with 16 GB RAM and no NVIDIA GPU. Runtime in milliseconds per frame (mean ± s.d.).

Discussion

We developed CHROMAS to be an easy-to-use, flexible and expandable tool for the objective and statistical quantification of the dynamics of chromatophore activity. Most of the steps in CHROMAS are built upon methodologies established in previous work (e.g., Reiter et al, 2018 and Woo, Liang et al, 2023), in particular chunking, segmentation, registration, stitching, and area calculation (Fig. 1). These processes were entirely restructured and rewritten in this study to prioritize efficiency, modularity and ease of use. Beyond these changes and improvements, CHROMAS introduces novel features which further refine the scope of achievable chromatophore analysis using computer vision methods.

First, our results achieve high levels of spatial resolution. Indeed, our anisotropy analysis captures the dynamics of the smallest cellular components of the chromatophore organs—their radial muscles. The numbers of motor neurons innervating single chromatophores and the numbers of chromatophores innervated by a single motor neuron, both estimated via our image analysis methods, match those measured earlier by direct and minimal electrical stimulation of terminal chromatophore nerves (Reiter et al., 2018), providing confidence that our new results are accurate. This image-based approach should thus enable one to describe the precise innervation patterns of very large fields of chromatophores, and their evolution in time, at chromatophore-motor-neuron resolution. This level of description for a complex motor output and behavior, especially with tools that require no external indicator is, to our knowledge, unique.

Second, our semi-automatic methods for tracking chromatophore identity overcome key challenges in long-term monitoring, the biggest one of which is the continuous addition of new chromatophores as the animal develops. This development enables the tracking of chromatophores over very long times (weeks to months) thus spanning the entire lifespan of an animal.

Scalability

Tasks within CHROMAS are designed for efficient parallelization. This enables scaling to the full capacity of single machines as well as execution on high-performance supercomputers. In addition, “larger-than-memory” execution is supported, allowing for the processing of datasets whose size exceeds the available RAM (e.g., with 8K-resolution images). CHROMAS uses an approach to parallelization that does not depend on the hardware, ensuring code maintainability and versatility. This design allows the same script to run efficiently on various hardware configurations, from single-core machines to multi-core systems and clusters. Tasks are parallelized along multiple axes (chunk-based, temporal and spatial). While temporal parallelization is mainly used, some tasks (segmentation, registration, areas, slice areas) are parallelized along the spatial dimensions as well. For cluster-level parallelization, initial setup on the cluster is required. CHROMAS supports most high-performance-computing job schedulers found in scientific research environments, such as SLURM, SGE, TORQUE, LSF, DRMAA, and PBS. Hence the spatial scale and temporal resolution of our analyses are bound less by the capabilities of CHROMAS than by the quality of the image data and the available computational power.

Usability

CHROMAS can operate in four distinct modes, adapted to user needs. The software can be controlled by a graphical user interface (GUI), enabling quick and easy operation without programming knowledge, and widening the pool of potential users. It can also be controlled by a command line interface (CLI), offering additional flexibility as well as automation and scripting capabilities. We provide an extensive low-level application programming interface (API), enabling custom workflows, integration with external tools and the development of extensions. Last, we provide jupyter notebooks facilitating interactive data analysis for common use cases.

The software can be installed either via the Pypi package manager, through the executable installer (.exe) or manually, using the Git repository. Extensive documentation is supplied and hosted.

Both high-level parameters, like algorithmic choices or storage configurations, and lower-level parameters, like concrete parameters of algorithms, are configured for ease of use within a single YAML file, which can also be modified through the GUI.

Known limitation

CHROMAS analyzes the activity of chromatophores by dividing the mantle into small mosaics (in the “cleanqueen”), where each chromatophore is assigned its own territory. Because these territories are discrete — they belong to only one chromatophore at a time— there can be species, ages and conditions in which the chromatophores expand so much that they encroach on each other's space and overlap partly. Currently, our software is not able to resolve these overlaps.

Methods

Implementation details

CHROMAS is written for GNU/Linux, Windows and MacOS operating systems in the Python programming language, requiring Python 3.9 or higher. The software relies on several key libraries for various functionalities. *NumPy* (Harris et al., 2020), *SciPy* (Virtanen et al., 2020), *pandas* (McKinney, 2010) and *xarray* (Hoyer & Hamman, 2017) are used for most of the numerical computations, utilizing labeled multi-dimensional arrays. *Dask* (Rocklin, 2015; Dask Development Team, 2016) is used for parallel computing and distributed task scheduling. Image processing and computer vision tasks are performed using *OpenCV* (Bradski, 2000), *scikit-image* (Van der Walt et al., 2014), and *decord* (Wang, 2019) for efficient video decoding. *scikit-learn* (Pedregosa et al, 2011) is used for data preprocessing, classification models and clustering, *PyTorch* (Paszke et al., 2019) and *torchvision* (Marcel & Rodriguez, 2010) for deep-learning-based segmentation models, and *albumentations* (Buslaev et al., 2020) for image augmentation used in the training of these models. Visualization is accomplished using *Matplotlib* (Hunter, 2007) for creating static, animated, and interactive visualizations, and *Bokeh* (Bokeh Development Team, 2014) for interactive dashboards visualizing computations. Data storage is performed using *Zarr* (Miles et al., 2020) for chunked, compressed, N-dimensional arrays. This

allows for efficient storage and access of large, chunked, and compressed N-dimensional arrays. The command-line interface is created using *Click* (Ronacher, 2014), while *tqdm* (TQDM Developers, 2016) provides progress bar functionality and the TUI is built using *Trogon* (Freeman, 2022). Documentation is generated using *Sphinx* (Sphinx Team, 2007). *nbsphinx* (Grünwald, 2017) is used to include Jupyter notebooks in the documentation, with *jupyter-client* and *ipykernel* providing Jupyter notebook support (Jupyter Development Team, 2015). *Pandoc* (MacFarlane, 2006) is used for document conversion and *rpy2* (Gautier, 2008) used for exporting data in R formats.

Selected sample data accompany the tutorials.

Training machine learning models to segment chromatophores

Data Augmentation

To reduce the amount of manually annotated images needed for training segmentation models, data augmentation techniques are employed to expand the size of the training dataset. This approach was also used in the training of the enclosed models. The *albumentations* library (Buslaev et al., 2020) is used for this purpose, offering a variety of transformations including resizing, shifting, scaling, rotation, horizontal and vertical flips, RGB-value shifting, brightness and contrast adjustment, downscaling, random shadows, FancyPCA and perspective transforms.

Models

CHROMAS offers three distinct types of segmentation classifiers, each tailored to accommodate varying levels of visual variability in the dataset, processing speed and available amounts of training data.

The first and simplest approach is a lookup table for color values in a specified color space. The training of this classifier involves selecting a color space (e.g. RGB, HSV, or Lab) and mapping a small set of representative color values to specific classes (e.g., background, chromatophore). Once the data points are mapped, all possible color values are classified in advance based on these inputs, generating a lookup

table that assigns pixel color values to the respective classes. This method is particularly well-suited for videos where the chromatophores have distinctly different color values from the surrounding skin.

Some of our datasets exhibited significant visual variability, stemming from changes in illumination (LED intensity, blue light, or lighting angle), optical adjustments such as aperture, and the skin's varying light reflectance and transmittance. For example, in *Euprymna berryi*, internal structures visible through the transparent skin contribute to making certain skin regions darker. To deal with these issues, we developed two other methods.

The second option uses a random forest classifier, a computationally efficient approach when lacking a Graphical Processing Unit (GPU) that requires relatively little training data and sometimes outperforms other methods when segmenting very small chromatophores (less than 10 pixels in size). However, its performance remains strongly tied to the lighting conditions of the training data.

The third approach uses deep learning models, specifically Fully Convolutional Networks (Long et al., 2015), DeepLabV3 (Chen et al., 2017), and U-Net (Ronneberger et al., 2015), with either ResNet50, ResNet101 (He et al., 2016), or MobileNetV3-Large (Howard et al., 2019) as backbones. This neural network-based method requires more training data, especially at the beginning. Our pre-trained models (trained on manually annotated images of *E. berryi* and *S. officinalis* using Cross Entropy loss), however, are designed to function right away, or to serve as robust baselines for fine-tuning, thus reducing the need for large datasets. While this approach requires access to an NVIDIA GPU, it generalizes well across conditions, making it the most flexible and powerful option.

Registration details

The Lucas-Kanade optical flow algorithm, used for video registration, tracks a sparse set of points across the video. These points are initially defined in the first frame as centers of mass of small chromatophores. The process starts by identifying

chromatophore regions through segmentation of the first frame, with an offset calculated to exclude points near the edges based on a percentage of the image dimensions. Candidates are filtered based on shape properties such as eccentricity, solidity and area, and are subsampled evenly across a grid to ensure that tracking points are well-distributed.

After selecting and tracking these points throughout the video, displacement maps of a finer equidistant grid are interpolated using the moving-least-squares algorithm. Full displacement maps are then calculated by linearly interpolating between these points, providing comprehensive registration of the video frames.

Stitching reprojection error

The stitching algorithm is used to align each masterframe in a dataset to every other masterframe. The accuracy of these non-symmetric mappings is quantified by calculating the reprojection error, as developed in (Reiter et al., 2018). Specifically, all points within a mask—either the animal's body if it is fully visible or a selected region of interest—of masterframe A are mapped to the reference frame of masterframe B using the A-to-B map, and then mapped back to the reference frame of A using the B-to-A map. The reprojection error is defined as the Euclidean distance between the original points and their remapped counterparts.

Clustering motion correlation data

PCA is a statistical technique used to reduce data dimensionality while retaining as much variance as possible (Jolliffe & Cadima, 2016). It identifies principal components—orthogonal directions that capture the most variance in the data. These components represent independent patterns of variation within the dataset. In the context of chromatophore anisotropy, motor neurons influence subsets of radial muscles, creating distinct patterns of contraction and expansion. By applying PCA to the activity data of the slices (and indirectly, the radial muscles), we could uncover the primary directions of variance, corresponding to the independent influence of individual motor neurons (to the extent that they were recruited independently of one another at least some of the time). We used the **elbow method** to set the number of components worth retaining: in a cumulative-explained-variance versus number of

components, the value of *x* at the inflection point beyond which additional components captured less and less additional variance— was chosen. Essentially, each retained principal component reflects a distinct source of coordinated muscle activity, likely driven by a separate motor neuron.

After identifying the number of principal components using PCA, we apply **ICA** to further identify the underlying sources of activity within our dataset. ICA attempts to separate the data into statistically independent sources (Bell & Sejnowski, 1995; Hyvärinen & Oja, 2000). In the context of our study, each motor neuron can be thought of as an independent source of muscle activity, which ICA is designed to isolate, even if the signals overlap spatially or temporally. This two-step approach—using PCA to find the number of components and then ICA to extract the independent sources—allowed us to capture both the number and activity of motor neurons acting on a single chromatophore's radial muscles, ultimately providing a clearer picture of the likely neuromuscular control behind their dynamic shape modulation.

To identify motor units (a motor neuron and the chromatophores it innervates), we first perform ICA on each individual chromatophore to extract the independent components (ICs) which correspond to motor neuron signals influencing that chromatophore's activity. To group the chromatophores innervated by the same motor neuron, we use clustering algorithms such as **Affinity Propagation** (Frey & Dueck, 2007) or **HDBSCAN** (Campello et al., 2013) to the ICs across all chromatophores, clustering by covariation. AP or HDBSCAN are particularly well-suited to this task because they determine automatically the optimal number of clusters based on the structure of the data, are capable of handling high-dimensional data, accommodate noise, and detect subtle relationships. This method allows us to identify putative common motor neurons and map their influence across multiple chromatophores, giving us an understanding of the overall motor units controlling the dynamic patterns of chromatophore activity.

Data availability

CHROMAS is distributed via the pypi package index (https://pypi.org/project/chromas/) and is publicly released on GitLab (https://doi.org/10.17617/1.pa38-mh49) under the 3-Clause BSD License. The documentation is hosted on GitLab. The data used to train the segmentation models, the trained models and example videos for the tutorial can be found at: https://public.brain.mpg.de/Laurent/Chromas2025/.

Acknowledgments

We thank F. Bayer for assistance in building the experimental setup; F. Kretschmer for optimizing the camera control, recording software and Git workflows; F. Vollrath and S. Junek for help with image acquisition; P. Musset for help with high performance computing; L. Jürgens, S. Schwind, L. E. Reyes de Frey, D. Burgard, M. Landler, M. Minde, S. Kranz, P. Dominiczak, N. K. Vogt, G. Wexel, S. Dizdarevic and E. Northrup for animal care; and T. Woo, X. Liang, D. Evans, M. Elmaleh and other members of the Laurent laboratory for constructive exchanges. We thank Alice Perenzin and Antje Berken for grant management and scientific coordination. This research was funded by the Max Planck Society (GL), by the LOEWE Schwerpunkt CMMS (State of Hesse) (GL) and by the European Union (ERC grant CAMOUFLAGE, 10114150) (GL).

References

Barbosa, A., Mäthger, L. M., Buresch, K. C., Kelly, J., Chubb, C., Chiao, C.-C., & Hanlon, R. T. (2008). Cuttlefish camouflage: The effects of substrate contrast and size in evoking uniform, mottle or disruptive body patterns. *Vision Research*, *48*(10), 1242–1253. https://doi.org/10.1016/j.visres.2008.02.011

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. Neural Computation, 7(6), 1129–1159. https://doi.org/10.1162/neco.1995.7.6.1129

Bokeh Development Team. (2014). Bokeh: Python library for interactive visualization. GitHub Repository. https://github.com/bokeh/bokeh

Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and Flexible Image Augmentations. Information, 11(2), Article 2. https://doi.org/10.3390/info11020125

Campello, R. J. G. B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, & G. Xu (Eds.), Advances in Knowledge Discovery and Data Mining (pp. 160–172). Springer. https://doi.org/10.1007/978-3-642-37456-2_14

Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation (No. arXiv:1706.05587). arXiv. https://doi.org/10.48550/arXiv.1706.05587

Cloney, R. A., & Florey, E. (1968). Ultrastructure of cephalopod chromatophore organs. Zeitschrift Für Zellforschung Und Mikroskopische Anatomie, 89(2), 250–280. https://doi.org/10.1007/BF00347297

Dask Development Team. (2016). Dask: Parallel computation with blocked algorithms and task scheduling. GitHub Repository. https://github.com/dask/dask

Florey, E. (1969). Ultrastructure and Function of Cephalopod Chromatophores. American Zoologist, 9(2), 429–442. https://doi.org/10.1093/icb/9.2.429

Florey, E., & Kriebel, M. E. (1969). Electrical and mechanical responses of chromatophore muscle fibers of the squid, Loligo opalescens, to nerve stimulation and drugs. Zeitschrift Für Vergleichende Physiologie, 65(1), 98–130. https://doi.org/10.1007/BF00297991

Freeman, S., & Contributors. (2022). Trogon: GUI generator for Click CLI commands. GitHub Repository. https://github.com/Textualize/trogon

Frey, B. J., & Dueck, D. (2007). Clustering by Passing Messages Between Data Points. Science. https://doi.org/10.1126/science.1136800

Gautier, L. (2008). rpy2: A Python-R bridge. GitHub Repository. https://github.com/rpy2/rpy2

Grünwald, M., & Contributors. (2017). nbsphinx: Jupyter Notebook Tools for Sphinx. GitHub Repository. https://github.com/spatialaudio/nbsphinx

Hanlon, R. T., Chiao, C.-C., Mäthger, L. M., Barbosa, A., Buresch, K. C., & Chubb, C. (2008). Cephalopod dynamic camouflage: Bridging the continuum between background matching and disruptive coloration. Philosophical Transactions of the Royal Society B: Biological Sciences. https://doi.org/10.1098/rstb.2008.0270

Hanlon R. T., Messenger J. B., & Young John Zachary. (1988). Adaptive coloration in young cuttlefish (Sepia officinalis L.): The morphology and development of body patterns and their relation to behaviour. Philosophical Transactions of the Royal Society of London. B, Biological Sciences, 320(1200), 437–487. https://doi.org/10.1098/rstb.1988.0087

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 770–778. https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. 1314–1324.

https://openaccess.thecvf.com/content_ICCV_2019/html/Howard_Searching_for_MobileNetV3_ICCV_2019_paper.html

Hoyer, S., & Hamman, J. (2017). xarray: N-D labeled Arrays and Datasets in Python. Journal of Open Research Software, 5(1), Article 1. https://doi.org/10.5334/jors.148

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90–95. Computing in Science & Engineering. https://doi.org/10.1109/MCSE.2007.55

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. Neural Networks, 13(4), 411–430. https://doi.org/10.1016/S0893-6080(00)00026-5

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Jupyter Development Team. (2015). Jupyter-client: Client libraries for Jupyter. GitHub Repository. https://github.com/jupyter/jupyter_client

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. 3431–3440. https://openaccess.thecvf.com/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html

Lucas, B. D., & Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. IJCAI'81: 7th International Joint Conference on Artificial Intelligence, 2, 674–679. https://hal.science/hal-03697340

MacFarlane, J., Krewinkel, A., & Rosenthal, J. (2024). Pandoc [Haskell]. https://github.com/jgm/pandoc (Original work published 2010)

Marcel, S., & Rodriguez, Y. (2010). Torchvision the machine-vision package of torch. Proceedings of the 18th ACM International Conference on Multimedia, 1485–1488. https://doi.org/10.1145/1873951.1874254

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Scipy. https://doi.org/10.25080/Majora-92bf1922-00a

Messenger, J. B. (2001). Cephalopod chromatophores: Neurobiology and natural history. Biological Reviews of the Cambridge Philosophical Society, 76(4), 473–528.

Miles, A., Kirkham, J., Durant, M., Bourbeau, J., Onalan, T., Hamman, J., & Patel, Z. (2020). zarr-developers/zarr-python: v2. 4.0. Zenodo https://doi. org/10.5281/zenodo.4069231

Packard, A. (1982). Morphogenesis of chromatophore patterns in cephalopods: Are morphological and physiological “units” the same? Malacologia, 23, 193–201.

Ronacher, A. (2014). Click: A Python package for creating beautiful command-line interfaces. GitHub Repository. https://github.com/pallets/click

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems, 32. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Reiter, S., Hülsdunk, P., Woo, T., Lauterbach, M. A., Eberle, J. S., Akay, L. A., Longo, A., Meier-Credo, J., Kretschmer, F., Langer, J. D., Kaschube, M., & Laurent, G. (2018). Elucidating the control and development of skin patterning in cuttlefish. Nature, 562(7727), 361–366. https://doi.org/10.1038/s41586-018-0591-3

Rocklin, M. (2015). Dask: Parallel Computation with Blocked algorithms and Task Scheduling. 126–132. https://doi.org/10.25080/Majora-7b98e3ed-013

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015 (pp. 234–241). Springer International Publishing. https://doi.org/10.1007/978-3-319-24574-4_28

Shannon, C. E. (1948). A mathematical theory of communication. The Bell System Technical Journal, 27(3), 379–423. The Bell System Technical Journal. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shohet, A., Baddeley, R., Anderson, J., & Osorio, D. (2007). Cuttlefish camouflage: A quantitative study of patterning. Biological Journal of the Linnean Society, 92(2), 335–345. https://doi.org/10.1111/j.1095-8312.2007.00842.x

Sphinx Team. (2007). Sphinx: Python documentation generator. Documentation. https://www.sphinx-doc.org/en/master/

tqdm Developers. (2016). tqdm: A fast, extensible progress bar for Python. GitHub Repository. https://github.com/tqdm/tqdm

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Wang, Y., & Contributors. (2019). Decord: Video loader and processor for machine learning. GitHub Repository. https://github.com/dmlc/decord

Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). scikit-image: Image processing in Python. PeerJ, 2, e453. https://doi.org/10.7717/peerj.453

Woo, T., Liang, X., Evans, D. A., Fernandez, O., Kretschmer, F., Reiter, S., & Laurent, G. (2023). The dynamics of pattern matching in camouflaging cuttlefish. Nature, 619(7968), 122–128. https://doi.org/10.1038/s41586-023-06259-2