Statistical Consulting

Segmentation of Myotubes and Myoblasts

Department of Statistics
Ludwig-Maximilians-Universität München

Giorgi Nozadze, Moustafa Amin, David Gorbunov

Munich, February 28th, 2024

By: Giorgi Nozadze, Moustafa Amin, David Gorbunov
Supervisors: Prof. Dr. David Rügamer, Tobias Weber
Project Partner: Dr. Maximilian Saller

Abstract

Myogenesis, the process of muscle tissue formation and regeneration, involves the proliferation, migration, differentiation, and fusion of so called myoblasts, which ultimately form multinucleated myotubes and mature into myofibers. To gain a better understanding of myogenesis, automated instance segmentation models are proposed to accurately quantify individual myoblasts and myotubes in microscopy images with the intention to overcome challenges such as overlapping instances and time consuming preprocessing steps, thereby enhancing analysis speed, reproducibility, and eliminating human bias. The pretrained `Stardist` model was used to segment myoblasts and Meta's `SAM` was trained to learn myotubes. In order to obtain segmentations for the training, a tool was created which utilized `SAM`'s remarkable zero-shot generalization. This way, fewer myotube masks had to be created manually. While `Stardist` displayed superb performance out of the box, the trained myotube segmentation model, dubbed `MyoSAM`, while displaying promising, did not generalize as well as `Stardist` due to a lack of training data.

Zusammenfassung

german text

List of Figures

Contents

1 Background and Project Goal

Myogenesis is the process that describes the formation, growth, development as well as regeneration of muscle tissues in the body. In detail, mononucleated, partially differentiated precursor cells, also called myoblasts, proliferate, migrate, differentiate, align, fuse to ultimately form multinucleated myotubes. In their quiescent state, myoblasts are called myosatellite cells and require mechanical stress for activation. In vivo, myotubes mature into myofibers that stretch along the entirety of skeletal muscles [1–3].
Several diseases such as sarcopenia, muscle dystrophy, obesity, or diabetes severely affect the physiological homeostasis and lead to a loss of skeletal muscle mass and function (muscle atrophy) [4]. In contrast, physical activity does not just lead to an increase of muscle mass and function (muscle hypertrophy), but also reduces glucose levels and lipid accumulation. Treatment of diseases that lead to muscle atrophy often requires a multifactorial approach. Therefore, candidate compounds, genes, proteins or metabolites are initially analyzed in large-scale in vitro experiments utilizing myoblast cell lines such as murine C2C12 cells [5].
Given the spatiotemporal nature of myogenesis, it is crucial to identify and quantify individual myoblasts and fused myotubes in microscopy images to evaluate the effect of each compound or biological molecule on myoblast fusion or atrophy/hypertrophy. Other than simply counting the number of each cell type, the creation and morphological quantification of individual myotube masks enables a thorough analysis of several other biologically relevant parameters such as the differentiation state/extend or myotube area and diameter in addition to interaction of different myoblasts. Unfortunately, the manual creation of individual cell masks in large-scale experiments is tremendously time consuming and might take months to finish. Therefore, this project intends to obtain such quantitative statistics without resorting to heuristics by using two different instance segmentation models for cell nuclei and myotubes.
These models need to overcome two hurdles. First and foremost, it must be able to differentiate between overlapping instances. Besides coincidence, these overlaps might be caused by myogenesis itself or due to the threedimensional nature of the cells caused by acutal overlapping myotubes. Secondly, it should be robust in its predictions. Many times, microscopy images require preprocessing that can create artifacts by amplifying noise or blurring small scale structures. Therefore, the instance segmentation should aim to be as independent of the preprocessing as possible.
Taken together, the goal is not only to speed up otherwise time-consuming analyses, but also to improve reproducibility and eliminate all human bias within the process of instance segmentation. Human bias can occur at various points. Many times, it is difficult to distinguish whether a continuously bright region is one object or several ones. Furthermore, the quality of preprocessing can influence the number of counted cells because some instances may be too dim too spot with the naked eye.

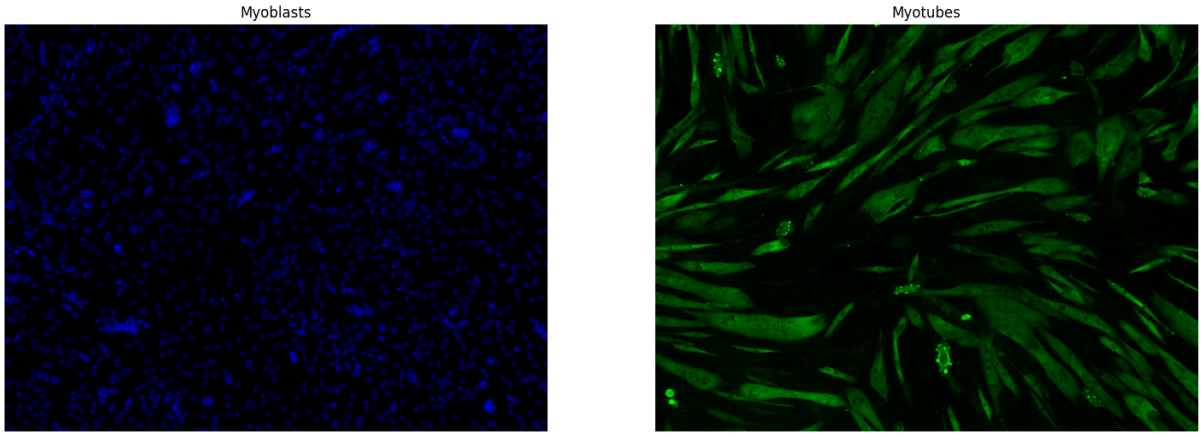


Figure 2.1: Microscopy image of myotubes and myoblasts. In this juxtaposition it is clearly visible that some myoblasts form the nuclei of the myotubes due to the dark circles within them.

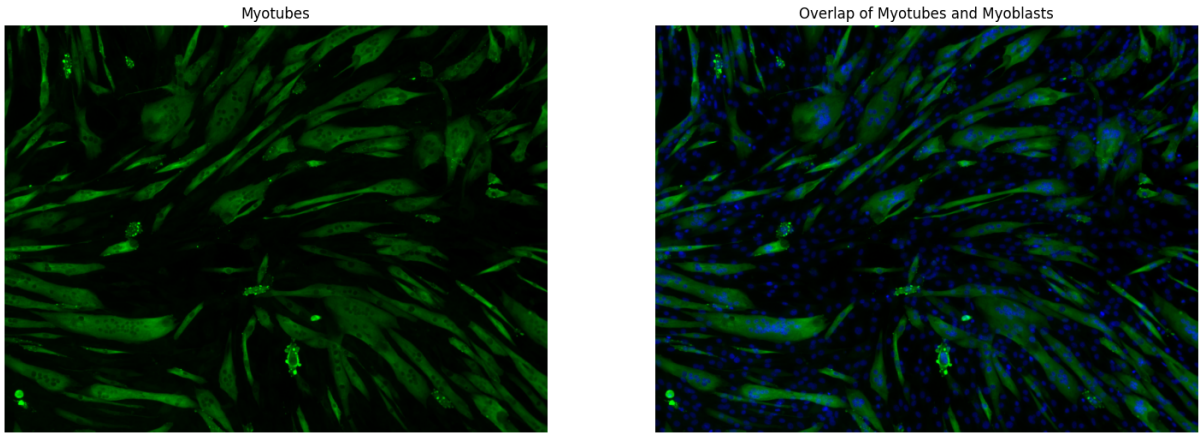


Figure 2.2: Overlap of myoblasts and myotubes. It clearly shows what myoblasts have become nuclei to myotubes.

2 Segmentation using Unsupervised Methods

In the beginning, established, unsupervised methods were used to segment the images in order to gain a better understanding of the data. Unsupervised learning includes algorithms which intend to find regularities, structures, or patterns within unlabelled datasets. As they require no ground truth labelled data and are not too computationally expensive, they are a great fit for exploratory analysis. The results of two classical methods (watershed and ellipse fitting) applied to the given images are discussed in the following. For this section, one myotube-myoblast image pair in Fig. 2.1 and Fig. 2.2 will be used to exemplify the unsupervised methods.

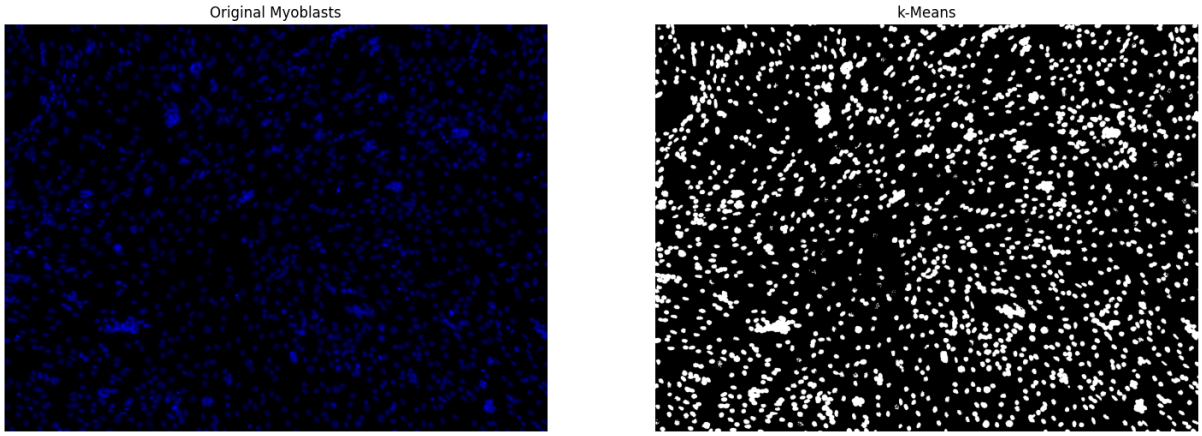


Figure 2.3: Foreground detection of myoblasts using k-Means

2.1 k-Means Clustering and Otsu Thresholding

As a starter, foreground detection algorithms were tried out in order to discriminate between cells, be it myotubes or myoblasts, and background. There are two common methods to do so in computer vision.
The first one is the k-Means algorithm, whose goal it is to minimize the within-class-variance of a (predetermined) number of k clusters [6]. To this end, k points are initialized and function as the first centers of the clusters. All observations are then grouped by proximity to the centers (according to some distance measure, commonly the $L2$–norm). This procedure is reiterated until convergence. In its simplest form, all initial points are drawn from a uniform distribution. A more common implementation is the k-Means++ algorithm [7] which samples only one point from a uniform distribution. All other points are sampled from a distribution proportional to the (squared) distance to the first point so as to minimize the chance of drawing cluster centers close to one another.
In order to use k-Means for image segmentation, the image first needs to be transformed to grayscale. Upon calculating the histogram of pixel intensities, the counts for every bin are the observations. Using $k = 2$ results in one threshold value seperating two intervals. All pixels with values above this threshold are considered foreground, all the other ones are background.
In a similar vein, Otsu's method [8] was utilized. In many ways, Otsu thresholding is not too different from k-Means in the sense that it also minimizes within-class-variance of the grayscale histogram. They can even be shown to be equivalent but only the Otsu method guarantees convergence to the global optimum [9]. Implementation wise, variance estimates are calculated explicitly by iterating the threshold over all possible grayscale values to find the optimal one with respect to the within-class variance as the objective function.
Both foreground detection algorithms applied to Fig. 2.1 can be found in Fig. 2.3-2.6. While this is not instance segmentation per se and therefore one cannot distinguish between overlapping cell instances, it is very helpful to gain intuition on the available images. One major drawback of thresholding methods is the fact that a hard cutoff implies that instances that are too dim are taken to be background.

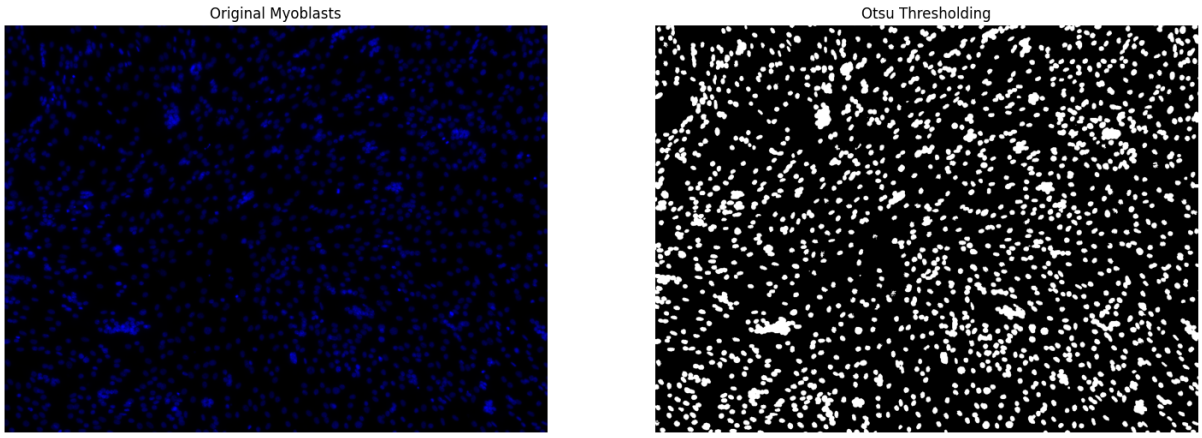


Figure 2.4: Foreground detection of myoblasts using Otsu thresholding

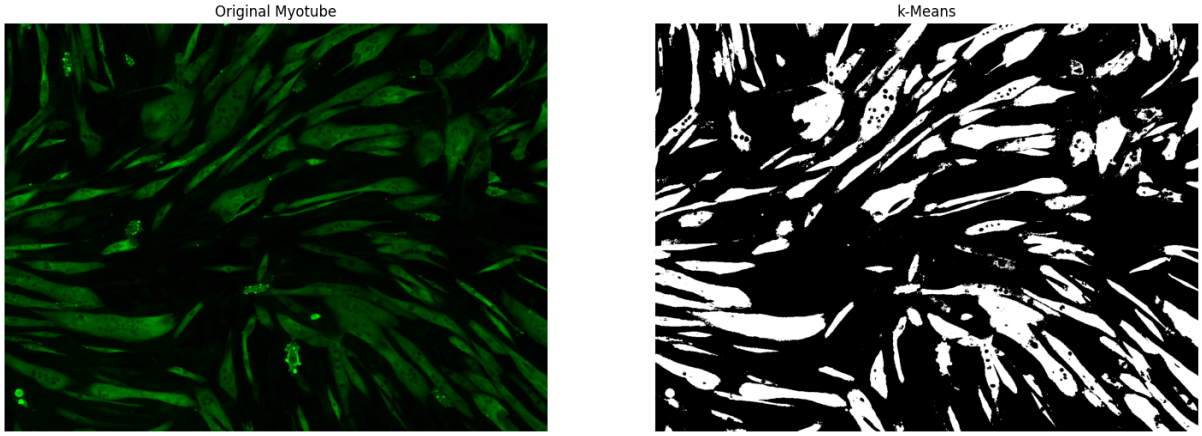


Figure 2.5: Foreground detection of myotubes using k-Means

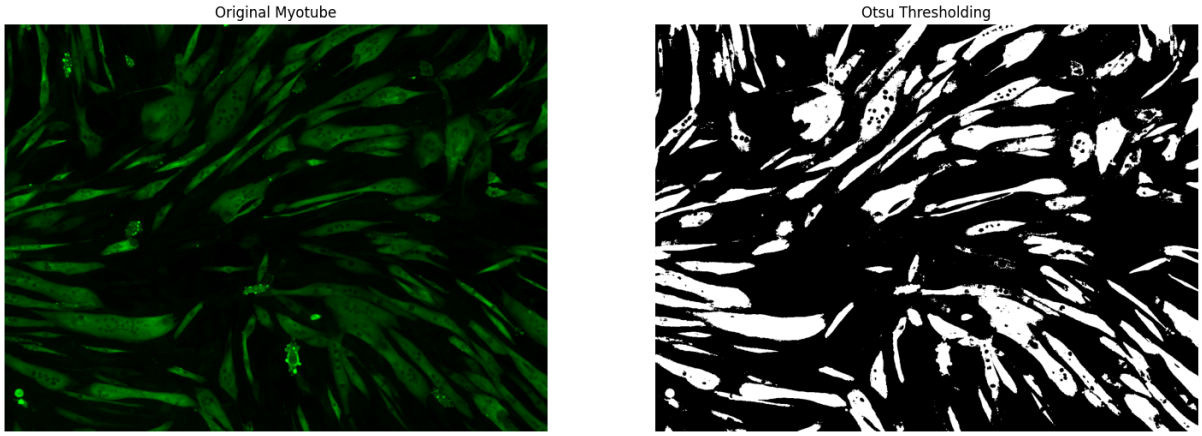


Figure 2.6: Foreground detection of myotubes using Otsu thresholding

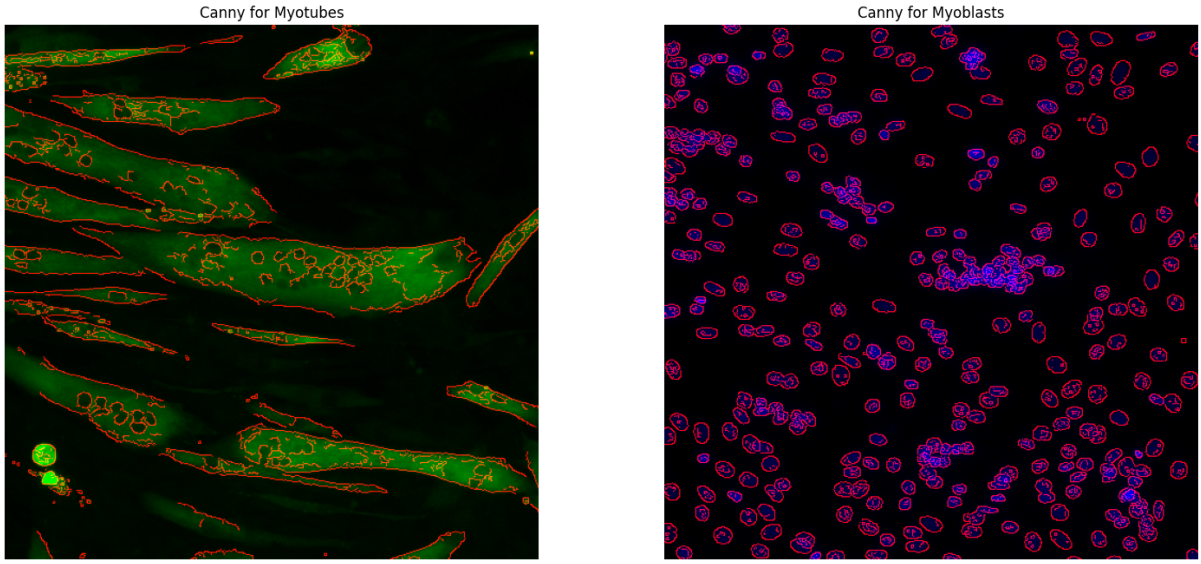


Figure 2.7: Canny edge detection applied to both microscopy images.

2.2 Edge Detection

As a further attempt to obtain instances of the cells, Canny edge detection [10] was employed in the hopes that overlapping instances have clear enough borders to be individually identified. In the Canny edge detection algorithm, a Gaussian filter is applied to the image to smooth out any noise present in the image. Next, the intensity gradients of the image are computed highlighting regions of significant change in intensity. The gradient thereby serves as a proxy for the edges. In this case, the Sobel filter [11] was used to estimate the image gradient. To further refine the edge detection, lower bound cut-off suppression is employed to thin out edges and retain only strong, relevant ones. This is accomplished by iterating over all non-zero pixels in the gradient image and checking whether its a local minimum among its eight neighbors. Following this, a double thresholding technique is utilized to classify pixels as potential edge pixels or non-edge pixels based on their gradient magnitudes. The lower value defines a hard cut-off such that every gradient pixel below that value is set to zero. Any pixel value greater than the larger value defines a strong edge. If a pixel is between the chosen values, the pixel is assumed to part of a weak edge. Weak edges are only kept if they are connected to strong ones. This is also known as hysteresis thresholding. This multi-step process ensures robust edge detection, particularly in scenarios where noise levels vary or edges are fragmented.
But yet again, this approach is not particularly desirable because obtaining decent segmentations from it requires a lot of tuning of hyperparameters as well as preprocessing and postprocessing steps tailored to every indivdual image (in addition to running the Canny algorithm in and of itself). Furthermore, the Canny algorithm in and of itself has many tunable parameters and also possesses the downsides that come with thresholding.

2.3 Watershed

Little math is necessary to understand how segmentation using watersheds functions. First, the image needs to be transformed to grayscale because the resulting single channel needs to be thought of as the a third dimension defining a height profile or topography. In case of a uint8 encoding, the height may take values between 0 and 255. Each pixel can be either of the three following types: a (regional) minimum, a catchment basin or watershed of that minimum, or watershed lines. The first type of pixel is self-explainatory. Continuing with the metaphor, a pixel of second or third type can be thought of in the following manner: picture the position and intensity of the pixel as defining the starting point on the 3d topography defined by the grayscale image. Placing a drop of water on this location can either have it run down (second type) or stay put (third type). All the points where water would run downhill are known as watersheds. All the other points that are not minima define crests, which are the divide (or watershed) lines, beyond which water would not move at all. Iterating over possible intensities starting from the lowest one in the image, or, by analogy, flooding the 3d landscape by poking a hole in the minimum, defines connected areas, or collections of water within a basin, around every regional minimum. Continued flooding will have the water level rise until the first two connected areas merge into one. To prevent that, a dam, whose locations define the pixels of the watershed lines, would need to be built. How to properly construct such dams by means of morphological operations [12] is thoroughly explained in [11, 13]. The resulting watershed lines are then interpreted as the boundaries of an instance. In this case, a marker-based approach to watershed [14] was opted for. Therein, before applying the watershed algorithm, the identification of markers, which are typically connected patches of pixels that represent objects of interest. These markers determine where the flooding begins.

Based on this intuition, two observations can be made. Firstly, on first glance a catchment basin can have the shape of a myotube or cell nuclei making watershed a sensible segmentation method. Secondly, this method requires the instances to have low grayscale values. This implies that images need to be processed before applying the watershed algorithm since cells are accumulations of high intensity areas. The most naive approach would be a simple inversion of the image. But this can lead to oversegmentation due to noisy sections. More sophisticated approaches either are based on image gradients or a distance transform applied to a binary representation of the original image. The former approach is used in this report and will be its result we be shown before long.

Just from these theoretical discussions alone, it becomes evident that the algorithm will have a hard time differentiating between merging nuclei because they presumably will be interpreted as one single catchment basin due to their intesities being similar. Therefore it still is very difficult to resolve sizable overlaps, unless of course the markers are set such that the algorithm knows that they are different a priori. So again, the results will, again, heavily rely on the quality of preprocessing which is unique work for every new image and something that is ideally avoided. All of these problems are glaringly obvious when applied to the the typical images in Fig. 2.8.

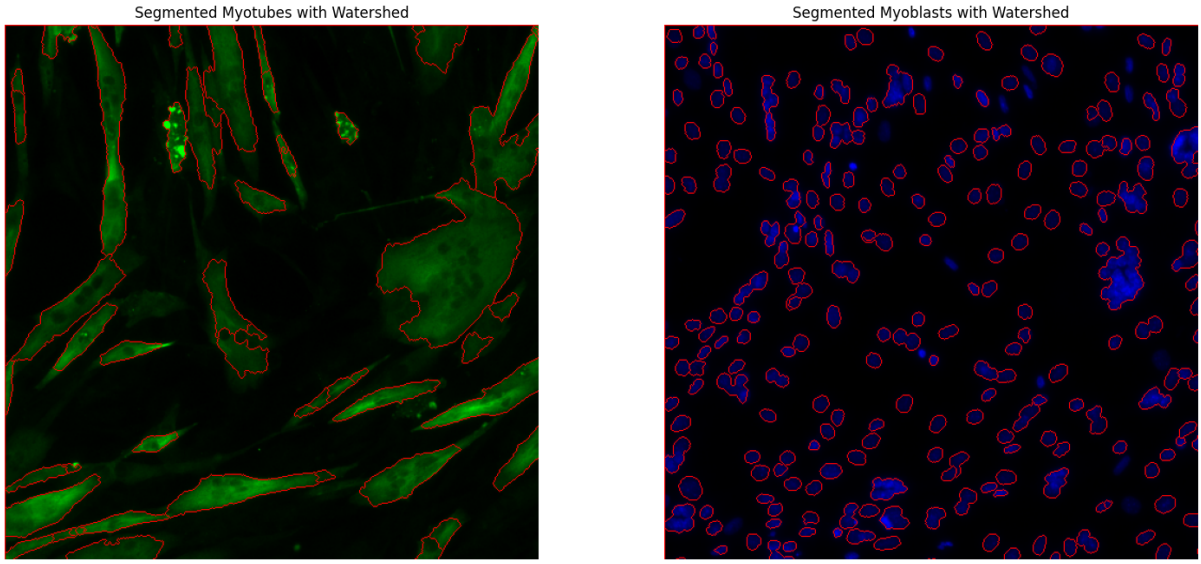


Figure 2.8: Marker-based watershed applied to both microscopy images.

2.4 Ellipse Fitting

As a last resort for unsupervised methods, it was attempted to fit ellipses to the myoblast images. It is evident that this approach would not work for myotubes due to their oftentimes crooked shapes. But for a typical myoblast image, it does seem to be a sensible approximation. There are several methods that have attempted similar in the literature [15–17]. All of them have a common structure. The image is preprocessed, which usually consists of methods previously delineated in this section like thresholding, Gaussian filtering, edge detection but also other things like hole-filling. At this point, the typical foreground detection has been performed. At this point, methods start diverging. In some cases ellipses are fit based on the curvature of the foreground. In other cases, many ellipses are fit and the fit with the highest goodness measure like AIC is kept. In addition to that, there are some computed quantities determining whether a segment is large, small, circular, or bright enough to be a cell. All in all, the final part distinguishes touching and overlapping cells one way or another. But even with all these things considered, the segmentation quality is easily influenced by the chosen preprocessing, as can be seen in Fig. 2.9.

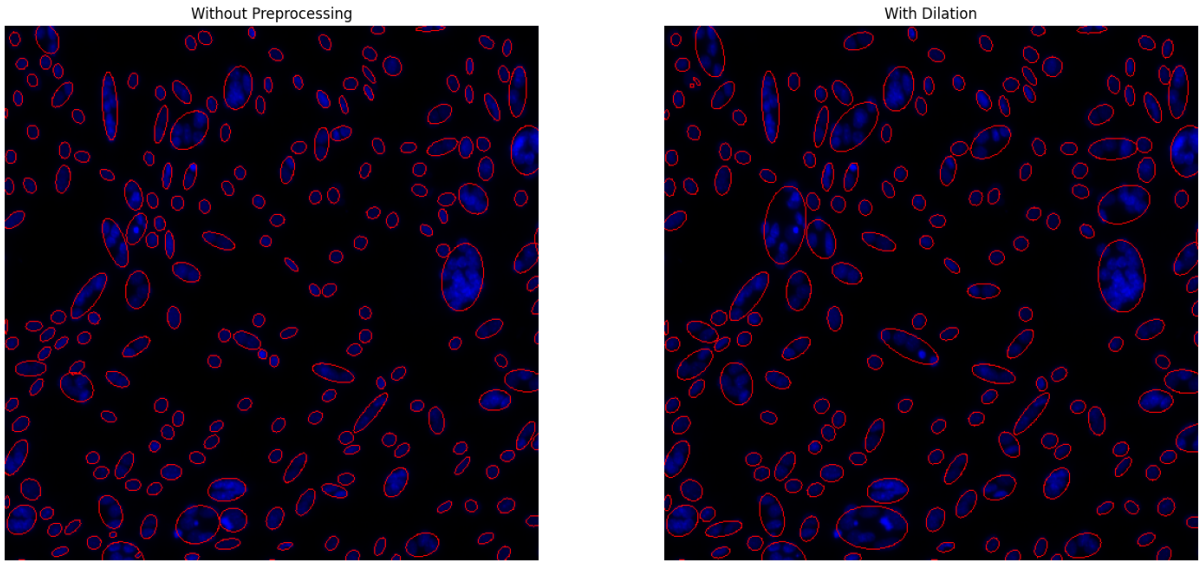


Figure 2.9: Ellipse fitting for myoblasts. One additional application of a dilation operation significantly alters the outcome of the segmentation.

3 Myoblast Segmentation using `StarDist`

As discussed in Sec. 2, typical instance segmentation methods suffer from the suppresion of valid objects or the merging of instances given overlaps. In order to better segment the myoblasts, a cell detection method called `StarDist`[18, 19] is used. It was developed with the intent to segment microscopy data like the nuclei found in this project. A deep learning model would have the advantage of being flexible enough to not require much preprocessing and can ideally be used out-of-box if there exists a pretrained model.

3.1 `StarDist` Background

In brief, a convolutional neural network is trained to predict polygons imitating typical cell shapes for every pixel of the image. More concretely, it predicts a star-convex polygon for every (non-background) pixel. Intuitively speaking, a star-convex set $\mathcal{S}$ is one where there exists one point s_0 such that for every point $s \in \mathcal{S}$ the line segment connecting s_0 to s is element of $\mathcal{S}$. Such a shape can be approximated by following n predefined radial directions for a distance of $\{r^k\}_{k=1}^n$ starting from a point s_0. This better approximates the myoblast shape than ellipses. The training data contains pairs of both raw and fully annoted label images in the sense that every pixel either has a unique object identifier or is marked as background. Given the annoted images, it is possible to compute the radial distances $\{r_{ij}^k\}_{k=1}^n$ to the assigned object's boundaries starting at every single foreground pixel parametrized by i, j serving as the point s_0. Furthermore, for every single foreground pixel the Euclidean distance to the closest background pixel is computed. Normalizing this distance gives a value d_{ij} between 0 and 1 interpreted as a probability to be a foreground point. The calculations are visualized in Fig. 3.1. At the end of the day, both object probabilities and the n distances are predicted for each image and denoted with $\hat{d}$ and $\hat{r}$ respectively. The object probabilities are penalized according to the binary cross entropy loss

$$L_{\text{obj}}(d, \hat{d}) = -d \log \hat{d} - (1 - d) \log(1 - \hat{d}), \tag{3.1}$$

and the object probability weighted mean absolute error is used as a loss

$$L_{\text{dist}}(d, \hat{d}, r_k, \hat{r}_k) = d \cdot \frac{1}{n} \sum_k |r_k - \hat{r}_k|, \tag{3.2}$$

for the radial distances. This implies that the further a point is away from an object boundary, the more it contributes to the loss and background pixels do not contribute at all.

In `StarDist`, the `U-Net` architecture [20] is used as the backbone and is slightly modified by adding another 128-channel 3x3-convolutional layer with ReLu activation to the `U-Net` output in order to add a lot of flexibility to the network. This output, in turn, is fed into two other convolutional layers. The first one it is input into is a single channel convolutional layer with sigmoid activation meant to learn object probabilities. The second one has a linear activation and as many channels as there are radial directions. In total, $n + 1$ values are predicted for each pixel. Presumably, two points close by will learn two similar representations for the same instance, a way too remove redundant predictions

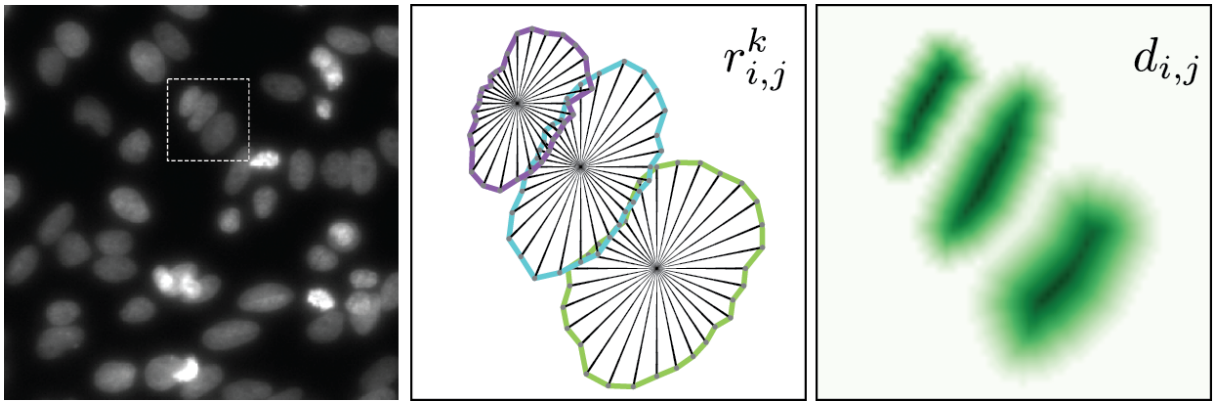


Figure 3.1: Example of an area that is complicated to segment due to overlaps. `StarDist` creates the segementation by forming star-convex polygons and computing the probability to belong to said instance. Source: [18].

is needed. This is accomplished through non-maximum suppression (NMS) [21, 22] where only polygons associated with high probability pixels are taken into account.

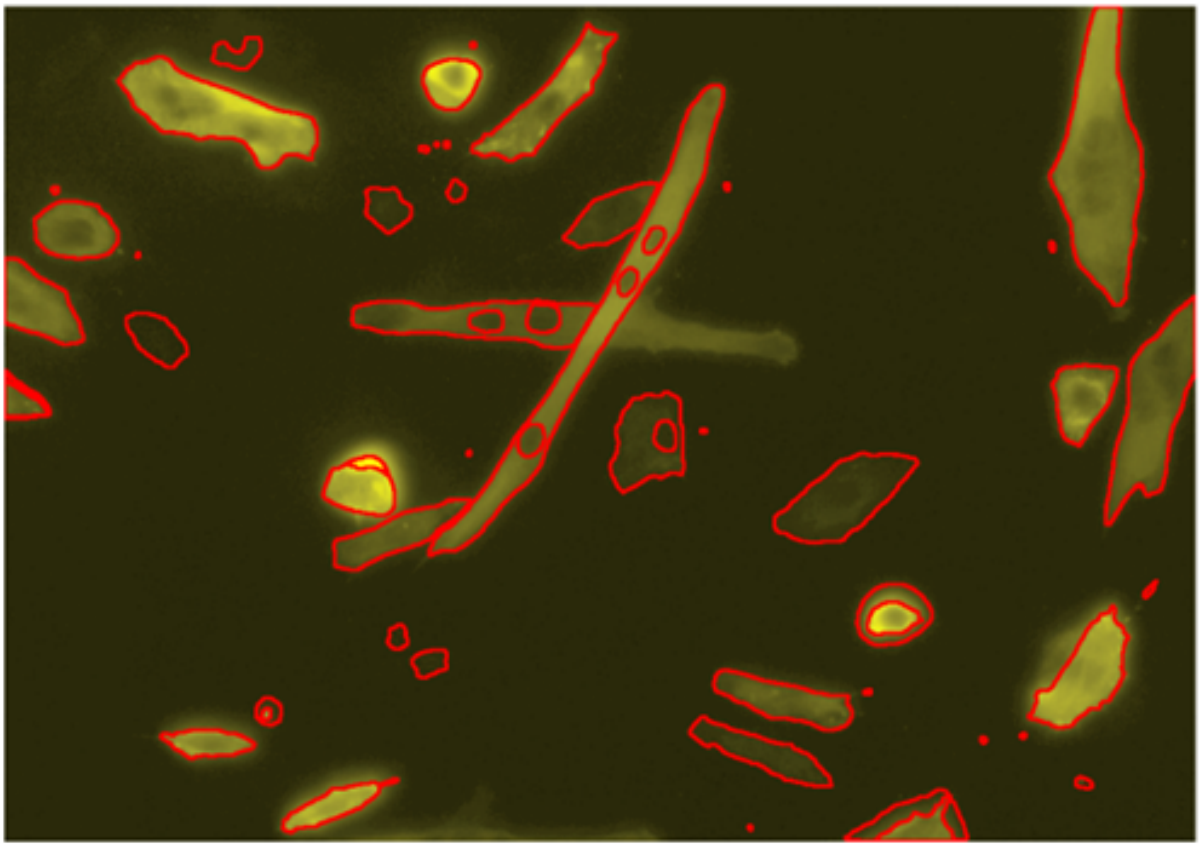


Figure 4.1: Out-of-box application of the pretrained version of SAM to one patch of a myotube image.

4 Segmentation of Myotubes with MyoSAM

Alas, typical myotubes cannot be represented as star-convex polygons since oftentimes they have slight constrictions (cf. Fig. 2.1) or even bifurcations. In order to be able to segment those, it was necessary to turn to a foundation model. The Segment Anything Model [23] (SAM) lays claim to this title. It is a promptable image segmentaion model based on an encoder-decoder network that can be used for downstream segmentation problems. Much of SAM was inspired by advances made in natural language processing (NLP). By visual inspection of its zeroshot performance, it becomes evident that it needs to be trained on suitable data. While it does a good job segmentating a lot of the structures, it also tends to oversegment noisy details or the cell nuclei. In addition to that, it runs into problems with the overlapping myotubes and does not find the second part of the tube in its entirety.

4.1 SAM Background

A high-level overview of SAM can be gained thorugh Fig. 4.2. The broad idea behind the model is simple: there needs to be a way to independently embed both prompt and image in order to jointly predict masks. The model therefore mainly consists of image encoder,

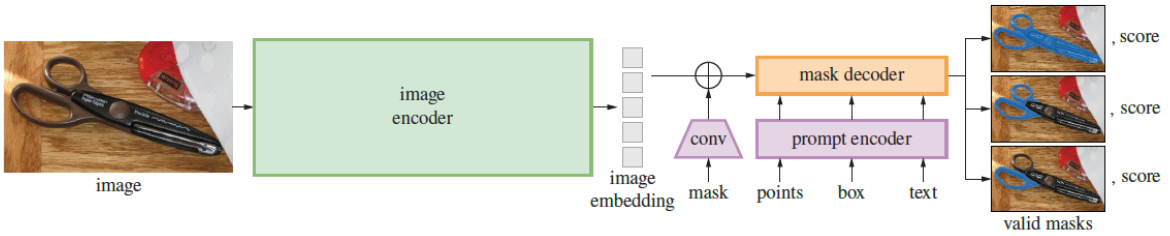


Figure 4.2: High-level overview of the two encoder networks interact to predict segmentation masks. Source: [23].

prompt encoder, and a decoder network. A modified vision transformer [24] (ViT) is used as the image encoder. When applying ViTs the input image is divided into fixed-size patches. Each patch is embedded and treated as a token analogous to words in text tying in with the similarities to NLP. One dimensional positional encodings are added in order to retain spatial information. The image embeddings, along with positional encodings, are fed into a transformer encoder [25] consisting of attention blocks and feedforward neural networks, enabling the model to capture global dependencies between patches. Following the transformer encoder, the output is pooled to generate a fixed-size representation of the entire image, which is subsequently processed by fully connected layers for the final task. This approach offers benefits such as global context understanding, scalability to larger image sizes, and fewer inductive biases compared to traditional convolutional neural networks. Images are rescaled and padded to have a resoultuion of 1024 x 1024. The image embedding is of size 64x64 and convolved twice to get to 256 x 64 x 64.
While the image encoder is allowed to be computationally expensive since it only needs to be applied one time per image, the prompt encoder should be fast to guarantee the ability to set many prompt at once. `SAM` allows for two types of prompts: sparse (text, boxes, points) and dense (masks) ones. As only point prompts were used in this work, only those will be discussed in the following and will simply be referred to as prompts. In order to encode the prompts, two pieces of information are required: foreground/background and position knowledge. The positional encoding is either summed with an embedding for the foreground or one for background to form a 256 dimensional vectorial embedding.
Both types of embedding are then included into the decoder network. The decoder is depicted in Fig. 4.4 and its primary part consists of four steps. These four steps are performed twice to update both image and prompt embeddings. In addition to the prompt embeddings, an output token is introduced and the combination their combinatation is then referred to as tokens. This is the only stage where there is any sort of information exchange between prompt and image. In the first step, self-attention is applied on the tokens. Afterwards the tokens are used as queries to compute the cross-attention to the image embeddings[1]. As a thrid step, a multi-layer-perceptron (MLP) updates the tokens. Eventually, the complement of the second step is performed: the cross attention of image to tokens. The learned image embeddings, which are now aware of the prompt information, are upsampled through transposed convolutions and, at the same time, are attented to once more by the tokens. A 3-layer MLP is used to transform these tokens to

[1]Note that image embeddings go hand in hand with their positional embedings, too.

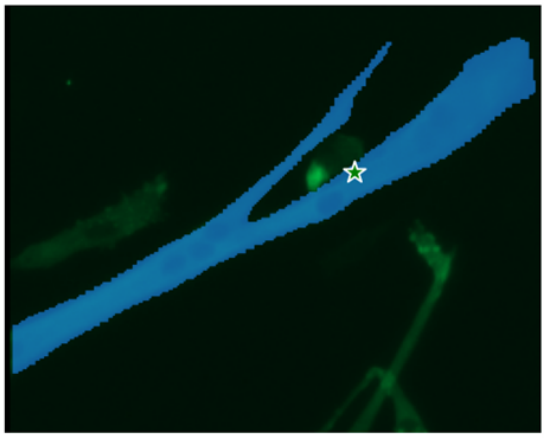
a dimension so as to match the upscaled image embeddings. A pointwise product between this MLP's output and the upscaled image embeddings results in the prediction of the mask. The aforementioned output token is used within a second head to predict values for the intersection over union (IoU) to the ground truth. In order to remove ambiguity due to a given prompt, several (three by default) masks are predicted and ranked by their IoU scores. During training only masks with the highest IoU are used for the calculation of the loss.
It remains to discuss the trainings loop. Interactive segmentation is simulated during training by randomly selecting foreground point sampled uniformly from the ground truth mask $G = (g_i)$ and selecting subsequent points from the error region (both false positive and false negative) between the previous (binary) mask prediction M and the ground truth, with each new point classified as foreground or background based on what type of error region E it was sampled from. A mask prediction is provided from the previous iteration as an additional prompt, using unthresholded mask logits $P = (p_i)$ to maximize information transfer. When multiple masks are returned, the one with the highest predicted IoU is used for the next iteration. After eight iteratively sampled points diminished returns are observed. Two additional iterations are included without new external information to allow the model to refine its predictions. One such iteration is performed at the very end and one is introduced at a random spot. This results in a total of 11 iterations, balancing the use of external guidance with the model's own learning. This lightweight mask decoder allows for a relatively large number of iterations with minimal computational overhead compared to previous methods. The loss that is used is made up of three components: dice loss, focal loss [9], and IoU loss. The latter is just the penalization of the IoU prediction and is nothing but the mean squared error. The dice loss penalizes predictions by the percentage of misclassified pixels. The focal loss is a modification of the cross entropy loss with a modulation factor γ to scale down the loss assigned to well-classified examples. All in all, the loss takes the form

$$L_{\text{train}} = L_{\text{dice}} + 20L_{\text{focal}} + L_{\text{IoU}} = \left(1 - \frac{2|M \cap G|}{|M| + |G|}\right) - 20 \sum_i (1 - p_i^t)^\gamma \log p_i^t + \|\text{IoU}_{\text{pred}} - \text{IoU}_{\text{gt}}\|_{L^2}^2, \tag{4.1}$$

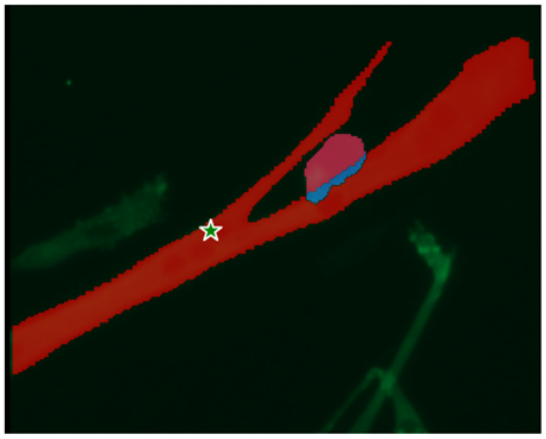
where $|\cdot|$ denotes the number of pixels of an area and the logits are defined as

$$p_i^{\text{t}} = \begin{cases} p & \text{if} \quad g_i = 1 \\ 1 - p & \text{if} \quad g_i = 0, \end{cases},$$

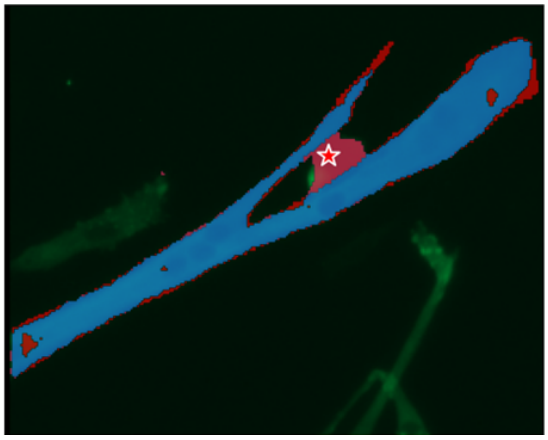
and the summation runs over all pixels. $\gamma = 2$ was used during training.
In order to not set all of the point prompts manually during segmentation, the SAMAutomaticMaskGenerator class was created. This allows to automatically set the number of points per side, the number of crops performed before setting these points, NMS thresholds, and many other scores in order to get many masks at once.



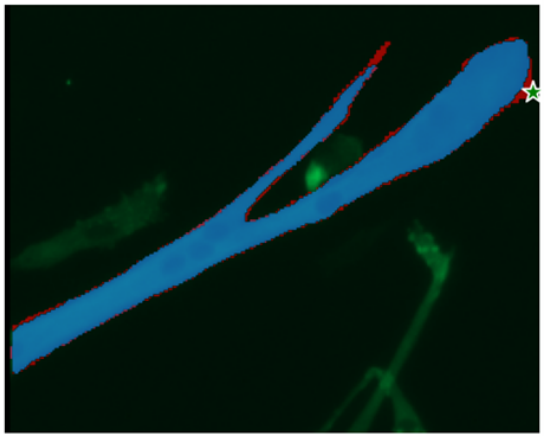
(a) First sampled point from the ground truth mask



(b) Error region after the first prediction including new sampled point



(c) New error region with another new point prompt



(d) Iteration beyond which resampling displays marginal return

Figure 4.3: Exemplifying the main idea behind the `SAM` training loop

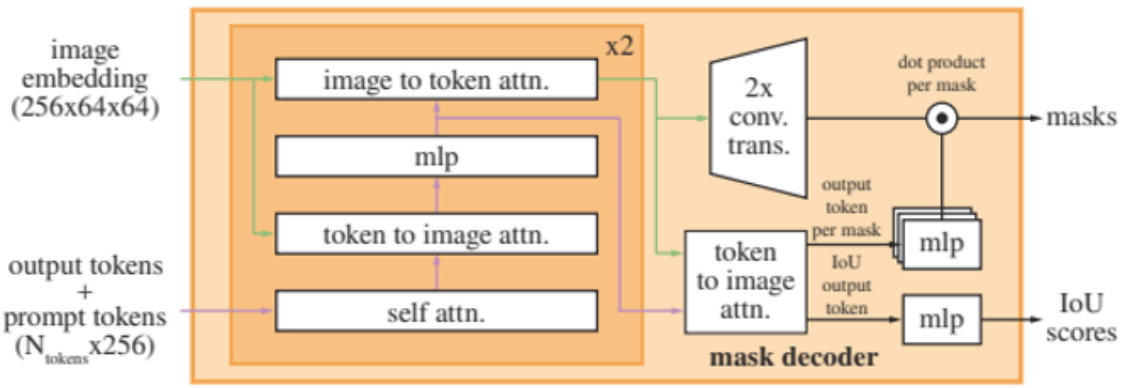


Figure 4.4: Four attention steps in the mask decoder. Source: [23].

Algorithm 1 SAM Training Algorithm

Initialize ground truth mask G
Initialize total number of iterations n
Initialize logit mask P = None
Initialize threshold t
Sample random integer $u \in \{k \in \mathbb{N} | k \leq n\}$
Uniformly sample a foreground point p from G
for $i = 1$ to n **do**
    Predict new logit mask P using point p and logit mask P
    if i = u or i = n **then**
        continue
    end if
    Update binarized mask $M_{\text{pred}} = P > t$
    Update error region $E = |G - M_{\text{pred}}|$
    Update point p by sampling uniformly from error region E
    if $E(p)$ is false negative **then**
        p is a foreground point
    else if $E(p)$ is false positive **then**
        p is a background point
    end if
end for

5 Implementation and Dataset

We developed our segmentation tool by integrating both `Stardist` and `SAM`, using `Stardist` for nuclei segmentation and a customized version of `SAM`, which we refer to as `MyoSAM`, for myotube segmentation. The primary modification in our approach involved training `MyoSAM` to enhance its capability for precise myotube segmentation. Below, we detail how each model was implemented to meet our project objectives.

5.1 `Stardist`

`Stardist` exceeded our expectations for nuclei segmentation, eliminating the need for additional training. We employed the "2D_versatile_fluo" model with its default settings. In contrast to `SAM`, Stardist operates efficiently without image patching, capable of processing 4000x4000 pixel images in approximately one minute on a CPU. The model outputs, including mask contours and learned centroids, were made use of in our subsequent analysis.

5.2 `MyoSAM` vs. `SAM`

One distinction between `SAM` and our curated version, `MyoSAM` (short for Myovision Segment Anything Model), stems from our analysis of myotube images and their unique pixel activation patterns. Unlike `SAM`, which was designed for everyday images with different pixel distributions, `MyoSAM` required specific normalisation for the RGB channels, using mean values of red 13.21, green 21.91, blue 15.04, and standard deviations of 7.26, 16.40, and 12.12. Additionally, given that myotubes do not possess hierarchical structures of interest, we simplified `MyoSAM` to use a single prediction head instead of three. We also limited the model to two types of prompts: point prompts to later make use of AutoMaskGenerator algorithm introduced in the original paper and discussed in Sec. 4.1 and mask prompts which were used to steamline the focus and improving efficiency of training. A retained feature from `SAM` is the use of bilinear interpolation to resize input images to a standard size of 1024x1024 for compatibility with the vision transformer. For large images, `MyoSAM` employs a patching strategy similar to that described in our data labeling process. Each image is divided into patches if necessary, with `MyoSAM` predicting masks for each patch using the `SAM` Automask Generator. After segmentation, the patches are reassembled and masks are merged at split points. Finally, we calculate all relevant metrics for the segmented images, providing comprehensive data for medical analysis.

5.3 Data Labeling

Our project's goal is to provide medical experts specializing in the musculoskeletal domain with a tool to expedite the segmentation of myotubes, a process traditionally known for its complexity. Initially, we faced a paradox: to train a model for myotube segmentation, we required labeled data, yet obtaining this data necessitated undergoing the very labor-intensive process we aimed to streamline. At the outset, we had no labeled data at our disposal. However, inspired by the promising zero-shot performance of `SAM`, particularly

its ability to recognize myotubes and their shapes, we decided to leverage it to accelerate our data annotation process, despite its tendency to produce extraneous masks. To cautiously generate labeled data suitable for training, our workflow involved several steps: We developed a tool, that employed great zero-shot generalisation performance of `SAM`, that enabled our experts to review all predicted masks and discard any inaccuracies. This approach significantly reduced the time required for data labeling, especially when compared to manual annotation. The myotube images we worked with varied in size, from 4000x4000 pixels to as large as 9000x9000 pixels. Predicting masks on images larger than 2000x2000 pixels demanded substantial computing resources not at our disposal, causing us to adopt an image patching technique for larger images. Given that `SAM` was trained on natural images, which differ significantly from the dark, detail-sparse backgrounds of most microscopy images, we allowed our expert to preprocess the images to improve mask prediction accuracy. Our preprocessing efforts revealed that, contrary to our initial belief, myotube microscopy images do not have a purely black background but instead ones with subtle coloring, from the markers used to visualise myotubes and nuclei. By focusing on eliminating large clusters of pixels with identical lightness based on the HSL space, we aimed to isolate the myotubes against a black background, enhancing their visibility. This preprocessing step, along with the expert validation of zero-shot generated masks, markedly accelerated the annotation process. However, image patches needed to be reassembled into their original dimensions, requiring adjustments to the binary masks to reflect the original image accurately. This reassembly process involved stitching together myotube segments that had been divided across patches. We merged masks at horizontal and vertical divisions, dropping any masks without corresponding segments in adjacent patches and merging those with a significant overlap as measured by the intersection over union at the edges. `SAM`'s design to accommodate prompt ambiguity led to the generation of redundant masks for the same myotube. To address this, we retained only one mask permyotube, discarding any that were completely enclosed by larger ones. Additionally, `SAM` sometimes produced disconnected masks for a single instance, which were also removed. The refined predictions were then presented to our expert for final validation and manual completion of the segmentation process. This iterative approach to data annotation was indispensable for creating a high-quality training dataset. Without it, we would have faced significant delays in acquiring manually annotated images diverse enough to train our model effectively. This diversity is crucial to accurately represent the true data distribution of myotube microscopy images in terms of their size, color, shape, and distribution.

5.4 `MyoSAM` Training Approach

In developing `MyoSAM`, our training methodology was based on the original `SAM` training process, with adjustments made to accommodate the challenges of data scarcity and computational resources. This section outlines the key differences and strategies we implemented. Our dataset comprised 19 annotated myotube images, which we divided into patches of 1500x1500 pixels. This division resulted in a total of 283 patches. To augment our dataset and increase its diversity, we applied 16 different augmentation techniques to these patches. Specifically, we performed colour jittering five times on each patch to make

the training set independent of color. In terms of spatial transformations, we applied five random rotations with rotation angles between ±45, inverted the image, and applied both vertical and horizontal flips. This created a more isotropic dataset. In order to imitate visual distortions, we added Gaussian noise twice, and applied a Gaussian blur. These augmentation processes expanded our dataset to 4528 patches. We divided the augmented dataset into training (80%) and validation (20%) sets. To ensure a balanced distribution of images and unbiased estimation of validation error, we shuffled the patches derived from the original images before splitting to make sure that both training and validation sets contain diversified patches and all images are represented in both sets. During training, we limited the number of instances sampled in each step to 54 in order to maintain stable memory usage and prevent out-of-memory issues. However, for validation purposes, we consistently used the same instances to ensure the comparability of validation error over time. The training was planned for 40 epochs but was terminated at the 21st epoch due to early stopping. To enhance training efficiency and manage memory use more effectively, we utilised Automated Mixed Precision (AMP) training. Additionally, we adopted Distributed Data Parallel (DDP) training to distribute the computation across six 40GB GPUs, allowing us to parallelize the workload effectively. The training batches were configured to include six patches, corresponding to one patch per GPU, with each patch containing up to 54 instances. This setup meant that a single training step could involve processing up to 324 instances. For optimization, we chose the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, incorporated L2 regularization with $\lambda = 0.1$ to prevent overfitting and set an initial learning rate of $10^{-}4$. We also applied a learning rate decay factor of 0.1 after every second epoch to gradually reduce the learning rate and improve convergence. The loss functions and training algorithm used in training `MyoSAM` were consistent with those used in `SAM`'s original training.

6 Performance Evaluation and Metrics

Evaluating the performance of computer vision tasks is far from trivial. They are typically categorized into four areas: image classification, object detection, semantic segmentation, and instance segmentation, each with distinct objectives. As a result, the performance metrics applicable to one category may not be suitable for another. Moreover, each segmentation task has unique requirements and deals with different types of images, which influences the relevance of performance metrics. In our work, we focus on binary instance segmentation for both myotube and cell nuclei images. Given the distinct characteristics of these image types, we selected a set of performance metrics that we believe most accurately reflect our model's performance. Instance segmentation tasks fundamentally involve classification, where metrics gauge a model's ability to differentiate between categories using counts of true positives, false positives, and false negatives. In binary instance segmentation, true negatives, which would correspond to the image background, are not considered. We assess our models' detection quality through metrics like precision (the rate at which predicted instances are correctly classified), recall (the ability to identify all relevant instances), and accuracy (the average of precision and recall). We also employ overlap-based metrics to compare predictions with ground truths in images and to gauge our models' segmentation quality. Metrics such as Intersection over Union (IoU), Intersection over Reference (IoR), and Normalized Surface Distance (NSD) are used, depending on the image type. These metrics are calculated for matched true positive and ground truth instances and reported both before and after normalization by the number of ground truth masks. Additionally, we report Panoptic Quality, a hybrid metric that combines segmentation and detection quality into one, yielding more conservative results due to its basis on measures ranging from 0 to 1.
(METRICS SUMMARY TABLE)

6.1 Performance Evaluation

In the following two subsections, we present these metrics for Stardist, used to segment cell nuclei, and `MyoSAM`, our model for myotube segmentation. We also compare the performance of `MyoSAM` to Meta's `SAM` model for direct comparison.

6.1.1 Performance of Myoblast Segmentation with Stardist

Qualitatively, Stardist effectively segments cell nuclei, closely matching the shapes of manually annotated reference masks as can be seen from Fig. 6.1. Despite its generally precise segmentation, Stardist tends to slightly underestimate the number of nuclei in densely clustered areas. This observation is supported by the difference in the number of reference masks (n=1923) compared to predicted masks (n = 1823). To quantify this performance, we matched predictions with references using the Intersection over Reference metric, allowing less penalization for predictions that segment multiple references and accommodating single predictions to multiple references. Double assignments in matching are considered false positives. The common threshold for mask matching in computer vision is 0.5, but this may be too conservative for small objects like cell nuclei, where

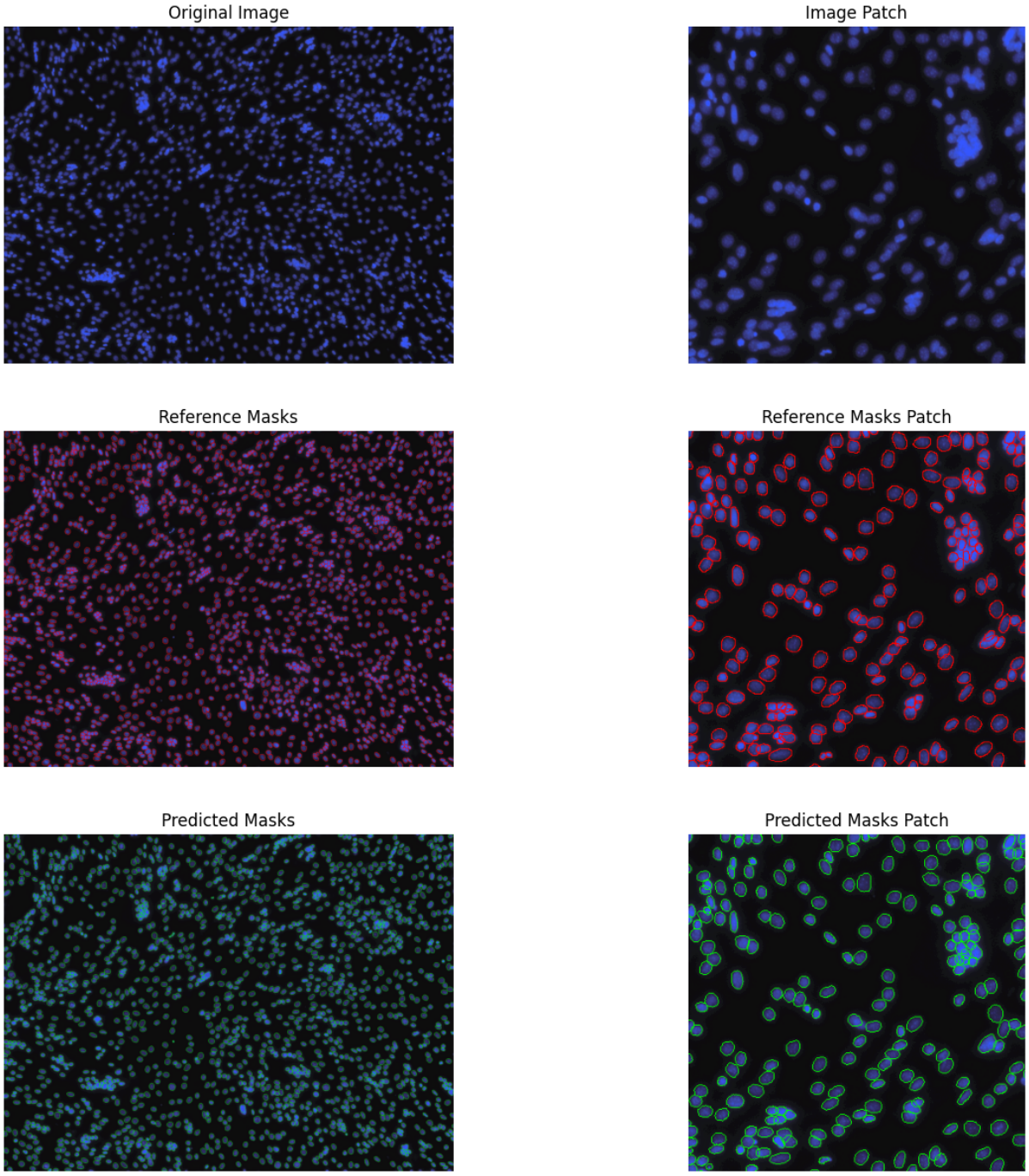


Figure 6.1: Comparison of one sample image and one patch therein in terms of ground truth and `Stardist` prediction.

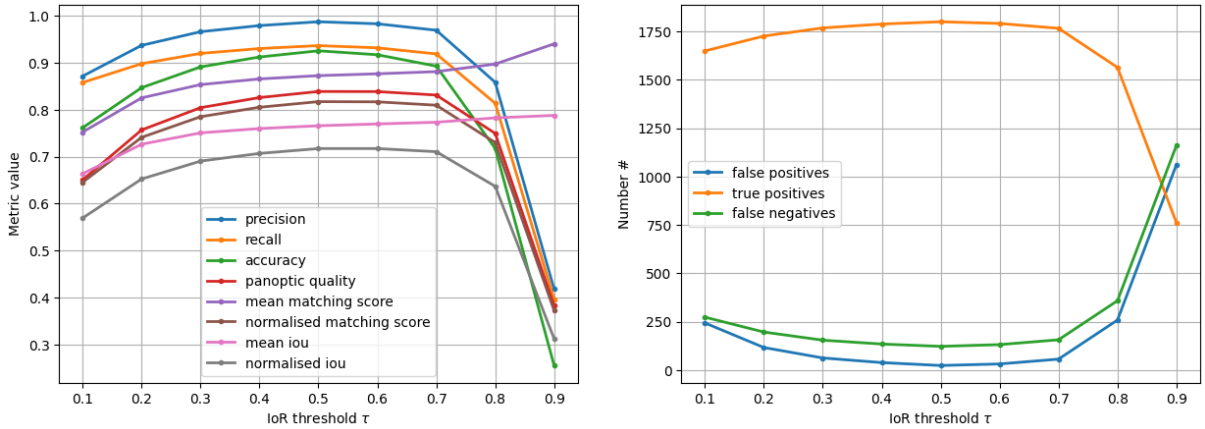


Figure 6.2: Performance of `Stardist` in terms of the described metrics and ROC metrics.

minimal prediction shifts can significantly impact overlap metrics. Despite this, Stardist's performance remains robust even at conservative thresholds (cf. Fig. 6.2).

6.1.2 Performance of Myotube Segmentation with `MyoSAM`

The performance comparison between `MyoSAM` and the standard `SAM` model highlights several key differences. Our focus with `MyoSAM` was on accurately delineating myotube boundaries, crucial for medical evaluations of microscopy images. To this end, we report the Normalized Surface Distance as an overlap metric for myotubes, which is a measure of overlap of the mask boundaries. NSD has an adjustable parameter tau, with which the boundaries of both masks can be widened to account for inter rater variability. We set the parameter at three, meaning the boundaries were widened by three pixels both inward and outward. Despite `SAM`'s capabilities, it faces challenges in accurately segmenting overlapping myotubes, often misidentifying them as separate entities or as disconnected parts of the same segment. Its design to segment a wide range of features results in the unnecessary segmentation of insignificant details, such as minor grains or shades within myotubes. Additionally, SAM sometimes generates redundant masks for a single myotube, a byproduct of its training to handle ambiguous prompts, despite there being no such ambiguity in myotube identification. This highlights the need for tailored approaches in myotube segmentation to avoid these specific issues. Our qualitative comparison in Fig. 6.3 shows `MyoSAM` significantly improves on these issues, but how do the metrics reflect this? We matched myotube references and predictions using IoU, and we recommend a 0.5 matching threshold due to the larger size of myotubes compared to nuclei. `MyoSAM` outperforms `SAM` in almost all aspects and at all except one matching threshold, except for a slight increase in the number of predicted masks, demonstrating a quality over quantity advantage. Their individual performances are seen in Fig. 6.4 and Fig. 6.5 while the improvement due to the training measured in terms of differences in performance metrics is seen in Fig. 6.6.
(INSERT TABLE WITH PERFORMANCE AT 0.5)
cite Metrics Reloaded, ref figs

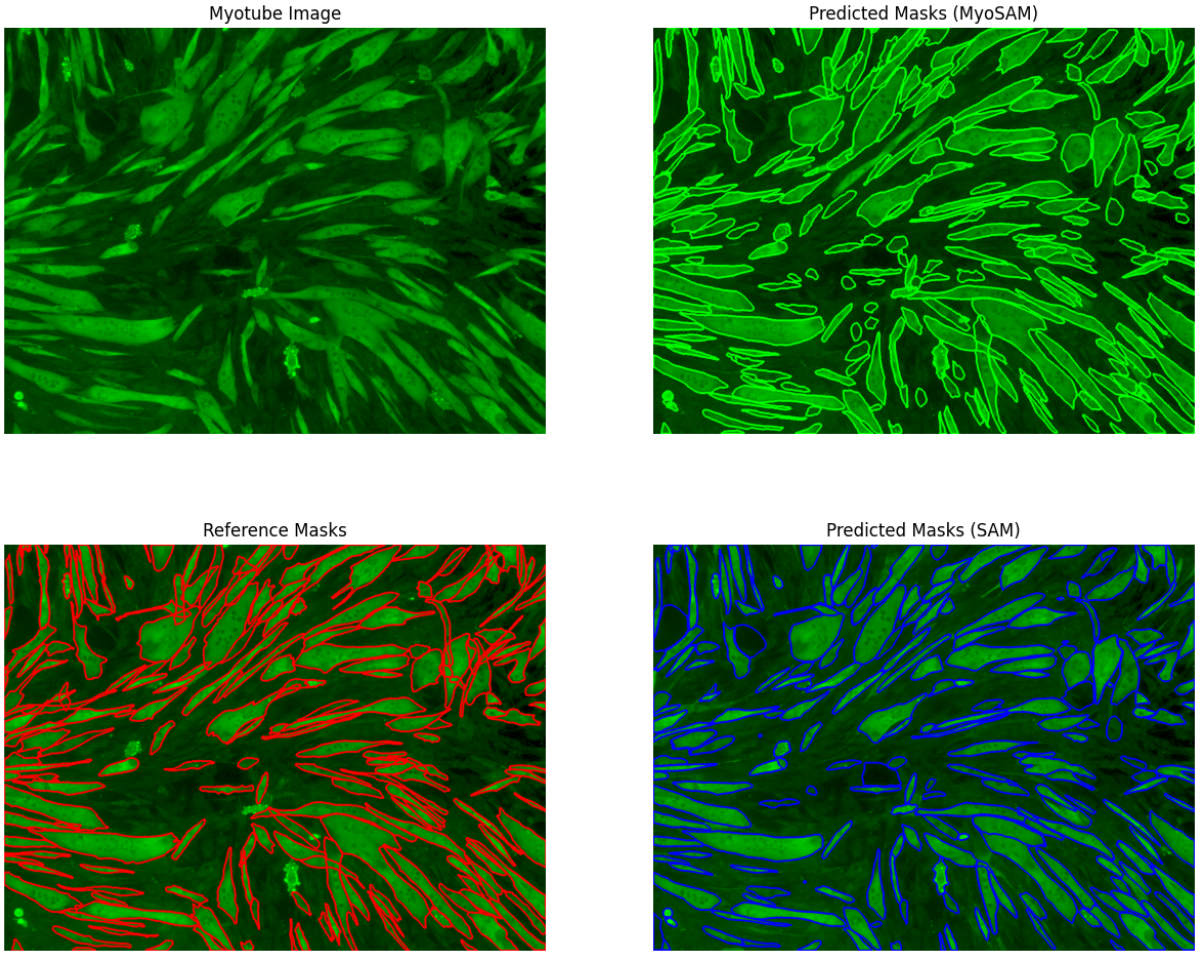


Figure 6.3: Qualitative comparison of a myotube image with `SAM`, `MyoSAM`, and ground truth.

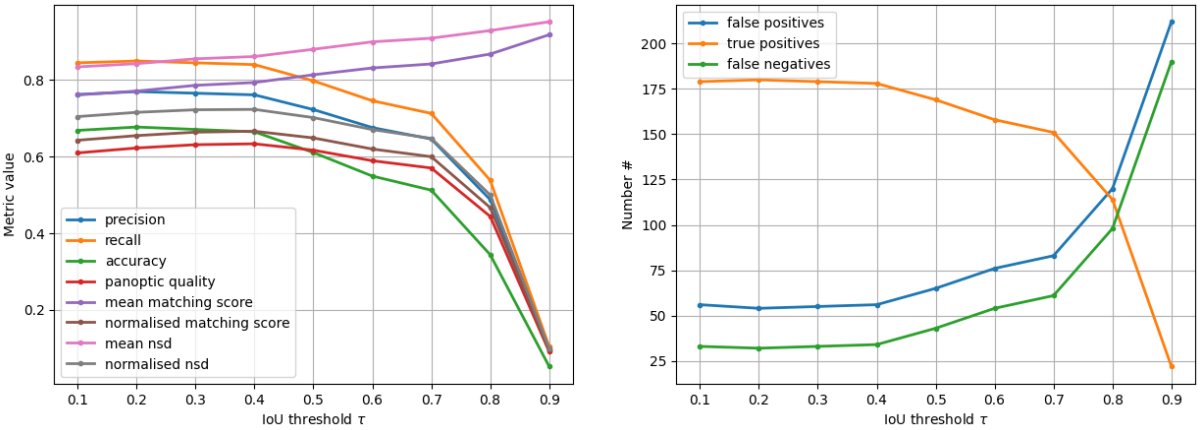


Figure 6.4: Performance of `MyoSAM` in terms of the described metrics and ROC metrics.

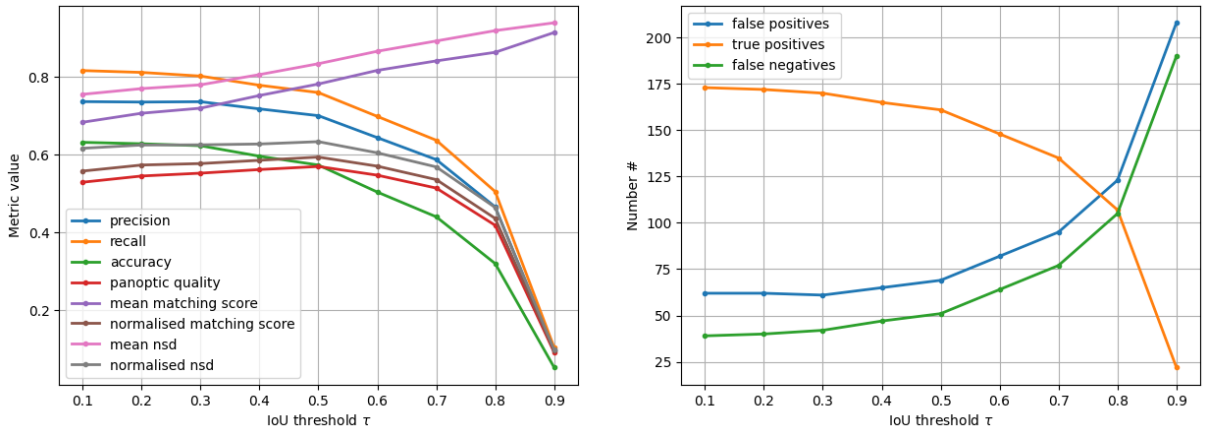


Figure 6.5: Performance of `SAM` in terms of the described metrics and ROC metrics.

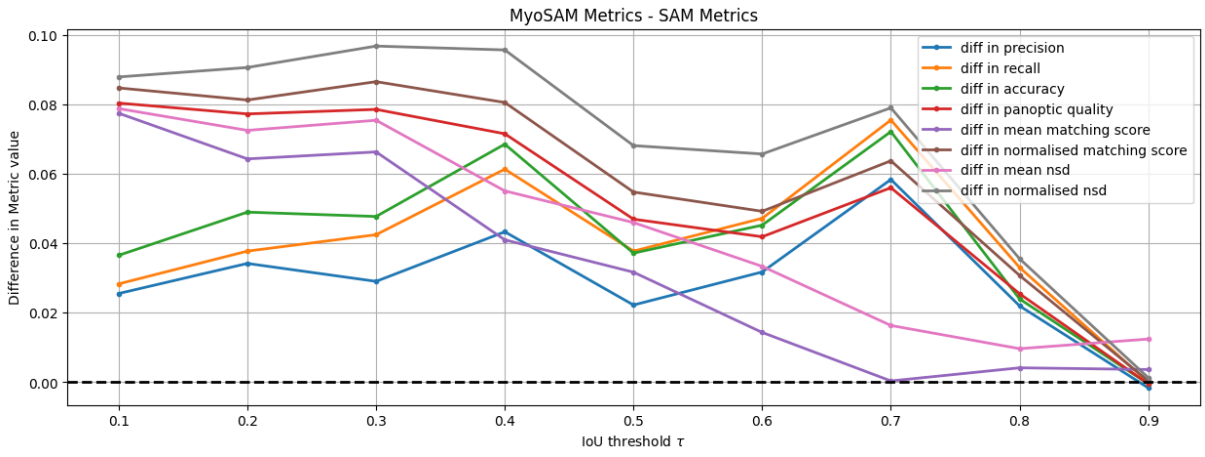


Figure 6.6: Performance differences between `MyoSAM` and `SAM`.

6.2 Myovision Metrics

Our research involves segmenting microscopic myotubes and cell nuclei, a process crucial for musculoskeletal researchers and medical practitioners. This segmentation allows for detailed analysis and understanding of myotubes and cell nuclei, providing valuable insights into muscular health and diseases. To support this analysis, our tool has been developed to offer a wide range of metrics in real-time to facilitate speedy examination with no human error. We offer a total of 31 metrics, divided into image-specific and myotube-specific categories, as detailed in our Myovision metrics table.

Table 1: Image and Myotube Specific Metrics

Image Specific Metrics	**Myotube Specific Metrics**
• Total myotubes	• Predicted IoU
• Total nuclei	• Stability
• Total myoblasts	• Is on edge
• Total nuclei inside myotubes	• RGB min
• Total fusion index (formula)	• RGB max
• Number of nuclei clusters	• RGB mean
• Total myotube area	• RGB median
• Total nuclei area	• RGB mode
• Total myoblasts area	• RGB standard deviation
• Total nuclei inside myotubes area	• Integrated density RGB
	• Area
	• Convex area
	• Solidity (formula)
	• Aspect ratio (formula)
	• Roundness (formula)
	• Perimeter
	• Feret's Diameter (min & max)
	• Circularity (formula)
	• Instance fusion index
	• Centroid
	• Cluster information

6.2.1 Image Specific Metrics

These metrics primarily count the instances of various musculoskeletal components within an image. Myoblasts are defined as nuclei not contained within a myotube. A nucleus is considered part of a myotube if 95% or more of its area is inside the myotube. This determination involves a multi-step process, starting with identifying if nuclei centroids are within a myotube's bounding box, then verifying if their centroids are within the myotube's perimeter. We further refine this by converting the relevant myotubes and nuclei into a binary mask to check the percentage of a nucleus's area inside a myotube. However, the masks we create correspond to the size of the myotube's bounding box instead of the whole image thus optimizing memory usage. Clusters of nuclei within the

same myotube are identified by detecting overlaps and creating a graph where nuclei are nodes connected by their overlaps. The number of nodes in this graph represents the number of nuclei in a cluster. Area measurements for myotubes, nuclei, and myoblasts are obtained by counting the pixels within each entity, allowing for conversion to metric scale measurements if the pixel size in the microscopic image is known.

6.2.2 Myotube Specific Metrics

These metrics include the predicted Intersection over Union (IoU) and Stability, both built in `SAM` metrics. Predicted IoU measures the predicted overlap between predicted and reference masks, while Stability assesses the sensitivity of predicted masks to changes in prompts. Pixel activation metrics (such as RGB min, max, mean, median, mode, and standard deviation) provide colour intensity information across three channels. The integrated RGB density metric sums the pixel intensities for each colour channel. Shape descriptors like Area, Convex Area, Solidity, Aspect Ratio, Roundness, Perimeter, Feret's Diameter, and Circularity offer detailed insights into each myotube's physical characteristics. For example, Convex Area is calculated from the area enclosed by the convex hull of a myotube, while Solidity and Roundness measure convexity and circularity, respectively. Feret's Diameter, both minimum and maximum, are derived from linear projections of myotubes using their eigenvectors, measuring their diameter in the directions of minimum and maximum variance. The Instance Fusion Index quantifies the number of nuclei within a specific myotube. Lastly, each myotube is accompanied by cluster information, detailing the number of clusters within a myotube and the number of nuclei in each cluster, along with their indicators. This comprehensive suite of metrics provided by our `MyoSAM` tool is designed to facilitate a deeper understanding and analysis of myotubes and their cellular components, offering a robust framework for musculoskeletal research and medical diagnostics. todo: add formulas in table, change font of SAM and Stardist

7 Conclusion

References

[1] S Tajbakhsh. "Skeletal muscle stem cells in developmental versus regenerative myogenesis". In: *Journal of internal medicine* 266.4 (2009), pp. 372–389.

[2] C Florian Bentzinger, Yu Xin Wang, and Michael A Rudnicki. "Building muscle: molecular regulation of myogenesis". In: *Cold Spring Harbor perspectives in biology* 4.2 (2012), a008342.

[3] Shelby E Bollen and Philip J Atherton. "Myogenic, genomic and non-genomic influences of the vitamin D axis in skeletal muscle". In: *Cell biochemistry and function* 39.1 (2021), pp. 48–59.

[4] Josep M Argilés et al. "Skeletal muscle regulates metabolism via interorgan crosstalk: roles in health and disease". In: *Journal of the American Medical Directors Association* 17.9 (2016), pp. 789–796.

[5] Piyush Bajaj et al. "Patterning the differentiation of C2C12 skeletal myoblasts". In: *Integrative Biology* 3.9 (2011), pp. 897–909.

[6] Christopher Bishop. "Pattern recognition and machine learning". In: *Springer google schola* 2 (2006), pp. 531–537.

[7] David Arthur, Sergei Vassilvitskii, et al. "k-means++: The advantages of careful seeding". In: *Soda*. Vol. 7. 2007, pp. 1027–1035.

[8] Nobuyuki Otsu. "A threshold selection method from gray-level histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[9] Dongju Liu and Jian Yu. "Otsu method and K-means". In: *2009 Ninth International conference on hybrid intelligent systems*. Vol. 1. IEEE. 2009, pp. 344–349.

[10] John Canny. "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698.

[11] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing* -. Amsterdam: Addison-Wesley, 1992. ISBN: 978-0-201-50803-1.

[12] J.P. Serra and J. Serra. *Image Analysis and Mathematical Morphology*. Image Analysis and Mathematical Morphology. Academic Press, 1982. ISBN: 9780126372410.

[13] Michel Couprie, Laurent Najman, and Gilles Bertrand. "Algorithms for the topological watershed". In: *Discrete Geometry for Computer Imagery: 12th International Conference, DGCI 2005, Poitiers, France, April 13-15, 2005. Proceedings 12*. Springer. 2005, pp. 172–182.

[14] Serge Beucher and Fernand Meyer. "The morphological approach to segmentation: the watershed transformation". In: *Mathematical morphology in image processing*. CRC Press, 2018, pp. 433–481.

[15] Costas Panagiotakis and Antonis Argyros. "Region-based Fitting of Overlapping Ellipses and its application to cells segmentation". In: *Image and Vision Computing* 93 (2020), p. 103810.

[16] Costas Panagiotakis and Antonis A Argyros. "Cell segmentation via region-based ellipse fitting". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 2426–2430.

[17] Sonal Kothari, Qaiser Chaudry, and May D Wang. "Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques". In: *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE. 2009, pp. 795–798.

[18] Uwe Schmidt et al. "Cell Detection with Star-Convex Polygons". In: *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*. 2018, pp. 265–273. DOI: 10.1007/978-3-030-00934-2_30.

[19] Martin Weigert et al. "Star-convex Polyhedra for 3D Object Detection and Segmentation in Microscopy". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. Mar. 2020. DOI: 10.1109/WACV45572.2020.9093435.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *CoRR* abs/1505.04597 (2015).

[21] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. *Learning non-maximum suppression*. 2017. arXiv: 1705.02950 [cs.CV].

[22] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[23] Alexander Kirillov et al. "Segment anything". In: *arXiv preprint arXiv:2304.02643* (2023).

[24] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[25] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

A Appendix