Volume 18
Number 12
26 March 2026
Pages 2371–2566

Analytical Methods

rsc.li/methods

ISSN 1759-9679

ROYAL SOCIETY OF CHEMISTRY

PAPER

Curtis J. Larimer *et al.*

M-Count: an application that uses machine learning object detection and color thresholding to count settled mussel larvae

Indexed in Medline!

Analytical Methods

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Cite this: *Anal. Methods*, 2026, **18**, 2395

M-Count: an application that uses machine learning object detection and color thresholding to count settled mussel larvae

Lance W. Miller,† Navaj Nune,† Thomas B. LeFevre, Clare N. Hermanson, Joseph D. Daddona, Anita Valdez, Ashwin Kannan, Wilaiwan Chouyyok, George T. Bonheyo, Raymond S. Addleman and Curtis J. Larimer *

Quantification of biofouling is a complex task that often involves counting both isolated and tightly packed groups of organisms on a test surface. Mussel larvae, which settle as both individuals and as clumps, are an important fouling organism because adult mussel colonies form *via* the settlement of larvae, making the settlement or repellence of mussel larvae a good indicator of a surface's antifouling performance. Manual quantification methods are time-consuming, and existing automatic machine learning-based methods are poorly suited for use by coding non-experts and often lack the ability to detect both isolated and grouped organisms in one workflow. The objective of this work was to develop a machine learning-based application that is user-friendly and well-suited to the quantification of biofouling. We developed M-Count, an application that combines a neural network object detection model for individual organism detection and a color thresholding algorithm for grouped organism detection. M-Count was demonstrated on the quantification of mussel larvae on sample surface images obtained from a previously developed mussel larvae fouling assay. This study revealed three important characteristics of M-Count: speed, consistency, and accuracy. The primary benefit of M-Count is its speed, being 60× faster than manual counting. The secondary benefit of M-Count is consistency, as it performs the same task repeatedly without bias. Finally, these benefits are obtained while maintaining good accuracy, with the normalized average maximum residual being 0.220 for M-Count and 0.209 for manual counting.

Received 21st October 2025
Accepted 15th January 2026

DOI: 10.1039/d5ay01759a

rsc.li/methods

Introduction

In aquatic environments, human-made surfaces become colonized by a variety of organisms. This unwanted biological accumulation, known as biofouling, is a problem in many sectors including aquaculture,[1] marine energy,[2] environmental monitoring,[3,4] and transportation.[5] The detection and quantification of biofouling is therefore important for a variety of stakeholders such as: industries seeking to optimize cleaning and maintenance schedules,[6–8] marine scientists performing image-based ecological surveys,[9] and materials scientists seeking to develop improved antifouling surfaces.[10,11] However, the visual (image-based) quantification of biofouling is made difficult by the complex composition of biofouling communities.[12,13] Visually, biofouling is often comprised of an intricate mixture of two forms of biological settlement: (1) well-defined individual organisms and (2) amorphous groupings or films of organisms. The former is best suited to object detection, and the latter is best suited to image segmentation.[14,15] However, object detection and segmentation are each complex tasks often performed in separate workflows.

Traditional (non-machine learning) algorithms and manual counting can be valid options for quantifying biofouling. For smaller sample sets, individual organisms can be quantified by manual counting, which is simple, accurate, and effective. But manual counting is subject to human bias and error and is laborious for large data sets. Manually operated digital methods, such as histogram-based thresholding, can be effective for groups or films of organisms when the background and lighting are amenable to such analysis. However, thresholding alone is not effective when there is insufficient contrast between the fouling layer and the background surface. Traditional algorithms that do not use machine learning also provide varying ranges of functionality for thresholding, blob detection, and particle tracking,[16–19] but lack the ability to quantify individual objects of a specific appearance.

Automatic deep learning methods have become popular for many tasks and are valuable tools for consistent and accurate analysis of large data sets.[20,21] In particular, machine learning object detection algorithms can be trained to detect specific

Pacific Northwest National Laboratory, 902 Battelle Blvd, Richland, WA, USA. E-mail: curtis.larimer@pnnl.gov

† Co-first authors

objects and patterns in images[22] and thus can rapidly quantify biofouling,[23] but their implementation requires extensive coding and configuration that are a hurdle for non-experts. Standalone applications for custom training and object detection have been developed, but they often lack the two-component detection stream necessary to detect both individual organisms and amorphous groups of organisms; are not open source; or have not been tailored to provide quantitative results.[24,25] Lacking these functionalities is an obstacle for use by coding non-experts. For more widespread adoption of deep learning tools to occur, there is a need for machine learning applications that can be easily installed and operated using a graphical user interface (GUI) to perform model training and object detection. A user-friendly machine learning application should be able to process large sets of images in one workflow, present the results in an organized fashion, and be readily updated with new training models as the machine learning field rapidly progresses.

In a previous work,[26] a settlement of larvae assay using mussels (SLAM) assay for the evaluation of antifouling surfaces was developed. Mussels are a common fouling species in both saltwater and freshwater, and on a wide variety of infrastructure and equipment, such as dams, nets, and boats. Furthermore, mussel larvae are convenient model organisms because small (2.5 cm–7.5 cm on each side) experimental antifouling samples can be exposed to a high density of mussel larvae – hundreds per cm^2 – inside a benchtop aquarium. This test can assess antifouling capability relatively quickly, as larval settling occurs within seven days. In a natural environment, the formation of adult mussel colonies begins with the settlement of mussels at the larval stage; thus, the prevention of larval settling on a given surface indicates that the surface may be effective at preventing adult mussel colonies. Field testing remains the gold standard for determining the efficacy of antifouling surfaces, but field tests typically require months to allow potential biofouling to occur. Field testing is also subject to a high degree of environmental variability and therefore lacks the controlled, repeatable conditions necessary to make precise comparisons between multiple experiments.

Mussel larvae are just large enough (250 μm–300 μm) that they can be imaged using a standard digital camera with an appropriate macro lens. Like many other forms of biofouling, mussel larvae settle as well-defined individuals and as amorphous groups. For the purposes of this work, these two categories of larvae are described as non-clumped and clumped larvae, respectively. Quantifying mussel larvae settlement in these experiments is challenging because manual counting is time consuming, while automatic methods are effective but difficult to use due to multiple workflows, complex and error-prone machine learning configuration, and script-based execution. Without an easy-to-execute analysis workflow, the previously developed SLAM antifouling assay is difficult to quantify and less likely to be adopted.

Here we present M-Count, an accessible and easy-to-use application that incorporates a machine learning object detection workflow to identify non-clumped individuals and a color thresholding process to identify amorphous clumps of individuals. M-Count was motivated by the need to efficiently quantify mussel larvae – both clumped and non-clumped – on experimental antifouling surfaces. M-Count uses two parallel processes to quantify biofouling in an image: a You Only Look Once (YOLO) machine learning model[22] that can be trained by the user to detect individual objects of any kind; and (2) an automatic color thresholding process that isolates the

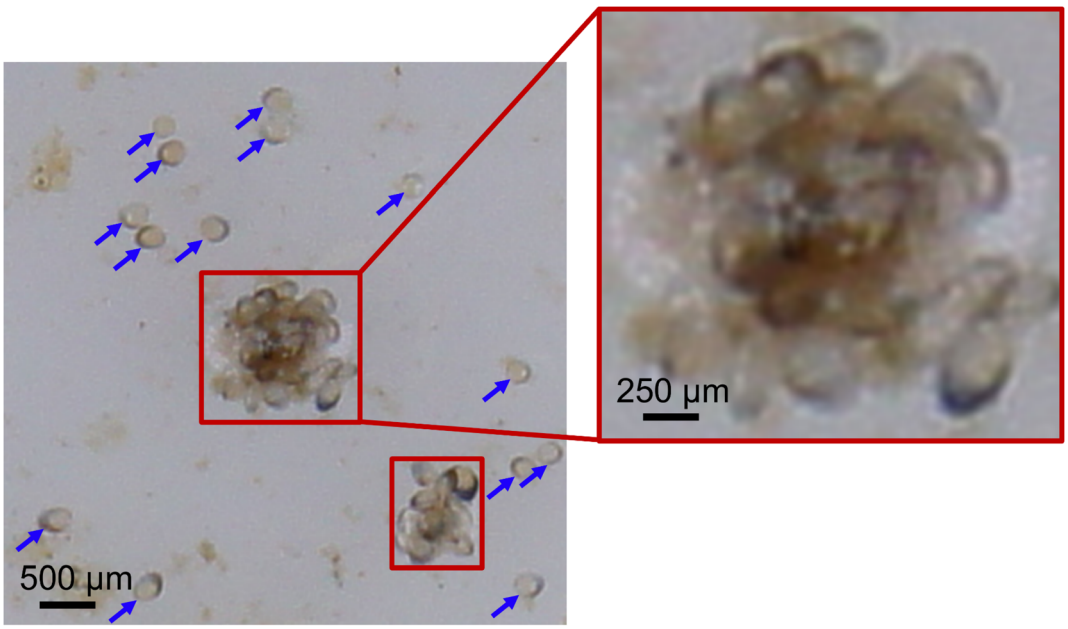

Fig. 1 Cropped subsection of a photograph of a sample coupon after 10 days in an aquarium with mussel larvae. Clumped larvae (manually outlined in red) and non-clumped larvae (blue arrows, manually placed) are clearly visible and can be quantified to characterize the antifouling efficacy of a surface.

amorphous clumps based on their color and quantifies their area. M-Count can be installed like any application, requires no coding knowledge, and features an easy-to-use GUI that allows the user to train their own machine learning model. We demonstrated the efficacy of M-Count on images that were acquired from the larvae settlement assay and compared the speed and efficacy of M-Count with manual counting. M-Count is an open-source, user-friendly application that has been designed specifically for quantification of marine fouling and rapidly quantifies mussel larvae with high accuracy.

Materials and methods

Mussel larvae assay execution and image acquisition

Fouled coupon images were acquired by executing a previously developed assay. Briefly, 5.1 cm × 5.1 cm test coupons of the following materials were prepared: white acrylic (TAP Plastics, USA), travertine stone tile (Home Depot, USA), 316L stainless steel with 2B mill finish (Stainless Supply, USA), Intersleek 1100 SR (AkzoNobel, Netherlands) painted acrylic, and siloxane composite painted acrylic (laboratory prepared). The commercial paint was applied according to manufacturer's instructions. The siloxane composite was prepared by mixing polydimethylsiloxane (PDMS) with crosslinker, suspending silica microparticles in hexane to form a slurry, mixing the particle slurry and a catalyst into the PDMS mixture, and applying the mixture to acrylic coupons using a 10 mL pipette.[27]

The prepared test coupons were placed in a 76 L aquarium with 58 L of artificial seawater. Larvae from a commonly grown commercial shellfish species, *Mytilus galloprovincialis*, were mixed into the seawater at a concentration of 259 larvae per cm^2 (relative to the horizontal area of the tank bottom). The tank was kept at 18 °C ± 0.1 °C, exposed to broad spectrum artificial light on a 12 hour on/12 hour off cycle, and fed with phytoplankton every other day. After 10 days, the coupons were removed and immediately photographed using a Sony NEX-C3 camera with included 18–55 mm lens using the manual focus option. The mussel larvae were approximately 250 μm–300 μm across and were clearly visible in the photographs (Fig. 1). The image analysis process from photographing the images to obtaining an organized spreadsheet containing the biofouling quantification data has been outlined in a block diagram in Fig. 2.

M-Count user interface

M-Count's graphical user interface (GUI) was coded in Python using the PyQt6 package. PyQt6 enables the creation of

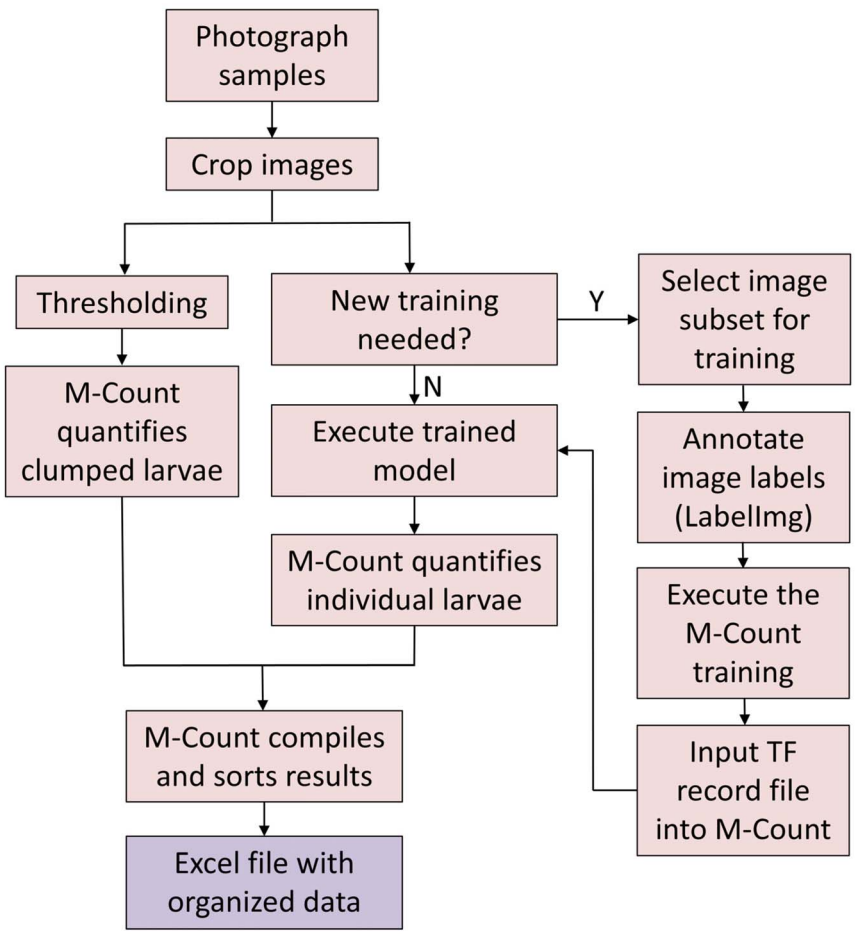

Fig. 2 Block diagram summarizing the entire M-Count workflow from acquiring sample images to obtaining finished results.

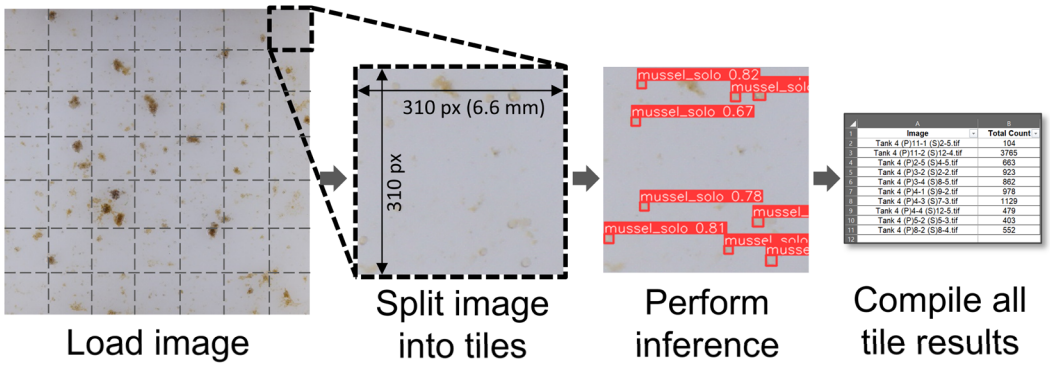

Fig. 3 Automatic process flow for the object detection and quantification of non-clumped mussel larvae using YOLOv8 detection in M-Count.

interactable buttons, windows, and labels, which make M-Count easy to navigate. PyQt6 can also use multi-threading, which allows the backend object detection and thresholding to run concurrently along with the frontend GUI. A configuration script was also created in Python to easily edit various aesthetical elements of the GUI, such as font and button size.

Image preparation

Prior to analysis, the 4912 px × 3264 px (10.9 cm × 7.3 cm) original .jpg images depicting 5.1 cm × 5.1 cm (2370 px × 2370 px) coupons were cropped to 2170 px × 2170 px, and saved as .tif files, to remove background and coupon edge effects from the fouling analysis. 2170 × 2170 is a multiple of 310 × 310 which is a convenient sub-image ('tile') size for analysis due to the standard processing size of many machine learning models. YOLOv8, however, is capable of processing images of any pixel dimensions less than or equal to the dimensions on which it was trained.

Training the object detection model (YOLO v8)

Training data was prepared by acquiring and cropping .jpg photograph images of fouled coupons with the following quantity per substrate color: 50 white (white acrylic and siloxane off-white coating), 5 stone, 5 steel, and 7 blue antifouling (Intersleek) paint. These 67 images were split into 49 tiles each, resulting in a pool of 3283 tiles. From this pool, 882 tiles were randomly selected for annotation and training in the following proportions: 50% white, 17% stone, 17% steel, and 17% blue antifouling paint. The tiles were annotated using the LabelImg package in Python, which creates a text file containing the bounding box coordinates of all the annotations for each image tile. After annotation, these 882 image tiles contained over 4500 annotated non-clumped mussel larvae. A smaller validation (94 tiles) data set was also created using a similar materials ratio.

Detection of non-clumped mussel larvae using an object detection model (YOLO v8)

YOLOv8 was integrated with the application by receiving input image files from the user through a GUI file dialog pop-up, then passing those images through backend Python scripts that run concurrently with the PyQt runtime. The M-Count process for detecting the number of non-clumped mussels, $m_{\text{nonclumped}}$, is outlined in Fig. 3. First, the input image files are passed into the tiling script, which breaks each 2170 px × 2170 px image into 49 different 310 px × 310 px tiles. Next, the detection script iterates through each tile generated, running YOLO predictions each time with the predict() function from the Ultralytics Python package. Once detections are complete, copies of the input tiles that are overlaid with bounding boxes are generated in the M-Count detections directory. The coordinates of each bounding box are temporarily stored in a text file, which is parsed to determine the number of detections per tile, which are then summed to determine the total number of detections per image, $m_{\text{nonclumped}}$. Once parsed, the detection count for each image is written into an Excel spreadsheet that is also saved in the M-Count directory, and the temporary file is deleted.

Detection of clumped mussel larvae using thresholding

The thresholding process is performed without any prior algorithmic training. Using a Python script that uses the OpenCV library, the thresholding process is applied to each full image uploaded by the app; no tiling is performed. An example image at each stage of the thresholding process is shown in Fig. 4. First, the image is converted from a red-green-blue (RGB) image into a hue-saturation-value (HSV) image. Then, the image is segmented by retaining pixels within a range of HSV values ([0, 30, 0] to [100, 255, 255], and [180, 180, 190] to [170, 155, 110]) that were empirically determined to capture a majority of clumps while minimising the capture of non-mussel surface details like texture and shadow. The HSV color model helps to prevent glare and shadow from influencing the result. There are two HSV ranges because the reddish-brown color of the mussels extends across the HSV spectrum such that it is contained on both sides of the 255 upper limit for the channels. The HSV thresholded image is then converted to grayscale using the cv2.color_bgr2gray method; then, a Gaussian blur filter is applied to smooth the image. Finally, all islands of pixels smaller than 15 px × 15 px (which would be non-clumped mussels or other non-mussel features) are removed; all retained pixels are then counted. This total pixel count, p, is then divided by the average pixel area of a single mussel, p_s, to determine the M-Count threshold clumped mussel count in the

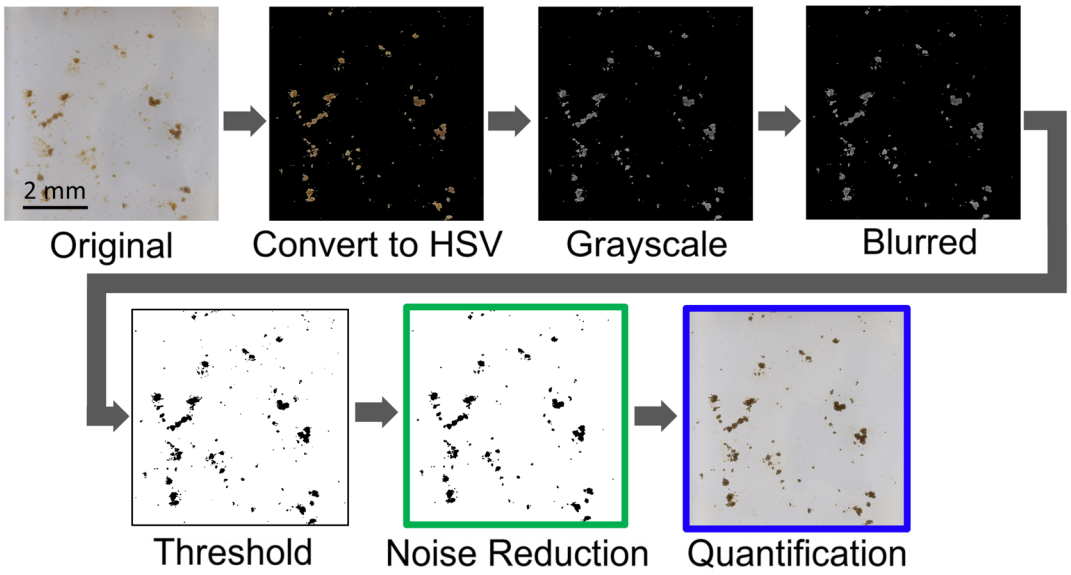

Fig. 4 Process flow diagram shows the steps of the clump detection thresholding process applied to a fouled coupon image. First, red-blue-green image is converted to the hue-saturation-value space and all pixels within a predetermined range of HSV values are retained, leaving all other pixels black. The retained pixels are converted to grayscale and then Gaussian blurred. Finally, the retained pixels are thresholded into a single color (black); islands of pixels smaller than 15 px in both length and width are removed to reduce noise; and the remaining pixels are then counted. Pixel counts are converted to mussel counts by dividing the pixel count by the average number of pixels per mussel, which was determined separately.

image, m_{clumped}, according to $m_{\text{clumped}} = p/p_s$. The average mussel pixel area, p_s, was determined by counting the number of pixels in 50 individual non-clumped mussels across five different coupon images and reducing that number by 20 percent to account for any overlap between mussels in clumps. After accounting for overlap, the average mussel pixel area was found to be 144.5 px. The total number of M-Count-detected mussels, m_{total}, is determined by $m_{\text{total}} = m_{\text{nonclumped}} + m_{\text{clumped}}$.

Detection of mussel larvae using manual counting

To perform manual counting on each image, the user loaded the coupon image into Microsoft Paint, placed red dots on every mussel larva they observed, saved the resulting file as .tiff or .png, and loaded the marked image into ImageJ. Using ImageJ, the images were thresholded to retain only the red dots, which were then counted using the Analyze Particles function to determine the total manual mussel count in the image, h_{total}. Manual counts were separated into clumped and non-clumped counts by using the M-Count generated mask of thresholded clumps to cover the manually tallied clumped mussels and counting only the non-clumped mussels using ImageJ to determine $h_{\text{nonclumped}}$. The manual clumped mussel count, h_{clumped}, was then determined by $h_{\text{clumped}} = h_{\text{total}} - h_{\text{nonclumped}}$.

To determine the repeatability of manual counting, twenty white coupons were manually analyzed by five different members of the research team trained in manual mussel counting. The average inter-day difference, $\bar{d}$ for each user was determined by having the user manually count the same coupons on two different days (days 1 and 2), calculating the inter-day difference for each coupon according to $d_i = |h_{1,i} - h_{2,i}|$ and then taking the arithmetic mean of the differences to find the average inter-day difference for that user across all 20 coupons, according to $\bar{d} = \frac{1}{20}\sum_{i=1}^{20} d_i$.

Calculation of residuals

Residuals for non-clumped mussels, clumped mussels, and total mussels were calculated independently using the following process. Using the 20-coupon data set from the previous section, the average human manual count for the jth coupon was calculated by taking the arithmetic mean of the five day 1 manual count results for that coupon across the five users, $\bar{h}_j = \frac{1}{5}\sum_{i=1}^{5} h_j$. This average manual count was used as the ground truth that each individual user and M-Count were compared against. For each of the 20 coupons, the maximum manual count residual was determined by first finding the largest difference between any manual count and the average manual count, $\max\_r_{h,j} = \max(\bar{h} - h_j)$. These maximum manual residuals were averaged across the 20 coupons to determine the average maximum manual residual, $\overline{\max\_r}_h = \sum_{n=1}^{20} \max\_r_{h,j}$. The M-Count average residual was calculated by the same two-step method: finding the difference between the M-Count result and the average manual count for each coupon by $\max\_r_{m,j} = \max(\bar{h} - m_j)$, and then taking the arithmetic mean of these twenty residual values, $\overline{\max\_r}_m = \sum_{n=1}^{20} \max\_r_{m,j}$. The average maximum residuals were then normalized by the average human count, according to $\text{norm}\_\overline{\max\_r}_h = \overline{\max\_r}_h/\bar{h}$ and $\text{norm}\_\overline{\max\_r}_m = \overline{\max\_r}_m/\bar{h}$.

Measurement of counting speed

Counting speed was measured by having five users manually count mussel larvae on the same single 5.1 cm × 5.1 cm coupon. The M-Count counting process was also executed on the same coupon. The number of larvae counted by each observer, and the time required for each observer to complete the count, were recorded. Count speed for each observer was calculated by dividing the number of larvae counted by the counting time. For manual counting, the counting time included the time required to mark all mussels in the image with red dots, load the image into ImageJ, and perform thresholding and particle analysis to determine the number of red dots in the image. For M-Count, the counting time only included the time required for the program to perform the count after the input image folder had been selected, because the image folder only needs to be selected once for an entire batch of images.

Results

M-Count graphical user interface

M-Count is freely available to the public through Github (https://github.com/pnnl/MCount).[28] Instructions for installation are included in the repository's readme file. To count mussel larvae using a previously trained model, the user clicks on the 'Count' button, which provides the option to 'Begin New Count' or 'View Past Detection Counts' as illustrated in Fig. 5. After clicking on 'Begin New Count,' a new window offers the option to click on 'Select Input Images Folder,' which opens a folder selection dialog box. The user navigates to the folder that contains the images to be analyzed and then clicks 'Select folder'. In the next window, to perform thresholding in addition to object detection, the user selects the 'Run Thresholding' checkbox. Selecting the 'Download Specific Thresholding Images' checkbox provides the user the option to save images from specific stages of the thresholding process (Fig. 4) for further examination; this box is typically left unchecked. Finally, the user clicks 'Run Model'. The app prompts the user to generate a name for the results folder, which it will then create within the 'Detections' subfolder of the 'M-Count (dist)' folder that is downloaded with the app, and the analysis will begin.

After the M-Count analysis is complete, the user can view the analysis results in two ways – either *via* the M-Count GUI, or by directly accessing the saved Excel data files through a file browser. To find saved results using a file browser, the user navigates to the 'Detections' subfolder within the 'M-Count (dist)' folder. Each analysis will have its own folder, named during the previous 'Run Model' step. Upon running the analysis, M-Count creates two subfolders within the named analysis folder: 'spreadsheets' and 'images'. The 'images' folder contains two further subfolders: 'bounding' and 'threshold'. The 'bounding' subfolder possesses copies of the input images overlaid by detection boxes, thus outlining all the detected non-clumped mussels. The 'thresholding' folder has copies of the input images overlaid with masks representing areas detected as clumped mussels by the color thresholding algorithm. The 'spreadsheets' folder contains an Excel spreadsheet named 'overall_counts' with three tabs. The 'Total mussels' tab shows the total mussel count for every image located in the selected input directory. The 'Bounding' tab has the mussel counts for non-clumped mussels detected by the YOLO model only. The 'Thresholding' tab contains the mussel counts for clumped mussels counted using the color thresholding method only.

The user can also navigate to the analysis results through the M-Count app *via* the options in the 'Count Complete' window that appears after the analysis is complete. Clicking on 'Open Detection Pictures' will open the aforementioned 'images' folder, which contains the copies of the original input images overlaid by non-clumped mussel bounding boxes and clumped mussel area masks. Clicking on 'Open Count Spreadsheet' in the same window will open the aforementioned 'overall_counts' spreadsheet with total mussel, non-clumped mussel, and clumped mussel results separated into separate tabs. These same options are available *via* the main menu by clicking 'Count', then 'View Past Detection Counts', and selecting the name of the analysis from a dropdown list of previously done analyses.

The process for training a new object detection model for counting non-clumped mussels is represented in Fig. 6. First, the user must first create a set of annotated images using a software package such as LabelImg, which allows the user to identify ('annotate') the location of each mussel larva in each image. For each annotated image, LabelImg creates a text file that contains the annotation labels (which are the name(s) of the objects being detected; in this case, 'mussel') and coordinates for every identified object. For example, using LabelImg, the user would draw a bounding box around every mussel larva in an image, and then LabelImg creates a text file that lists the x and y coordinates of the four corners of each bounding box in that image. LabelImg provides the option to record these annotations in different formatting schemes; the user should select the 'YOLO' formatting option. Once the annotated image set is ready, the user must create a text file saved in.yaml format that contains the annotation label name for the object class being detected (such as 'mussel') and the file path of the folder where the annotated training image set is located (formatted as 'train: "file path"' exactly as shown in Fig. 6, lower right). Validation of the training is optional: if desired, the user creates a unique separate annotated image set folder for validation and inputs its file path location into the .yaml file, similar to the training image path, under the heading 'val:'. When the training set and.yaml file are ready, the user opens M-Count and selects 'Train Model' in the main menu, followed by 'Select Training Config (.yaml)' in the resulting window. The user will be prompted to name the new model. After the user enters a name and clicks 'ok,' the training is performed, and the new model will be saved as a PyTorch weights file in the 'Training' subfolder within the 'Mcount (dist)' directory. If the user supplied an annotated validation set, the validation results are populated in a subfolder named 'runs' within the 'Mcount (dist)' folder. The newly trained model can be selected by clicking 'Select Model' in the main menu, adding the new model by clicking the 'Add model' button, navigating in the file selection pop-up to the 'Mcount (dist)/Training' folder, and selecting the desired model. The user

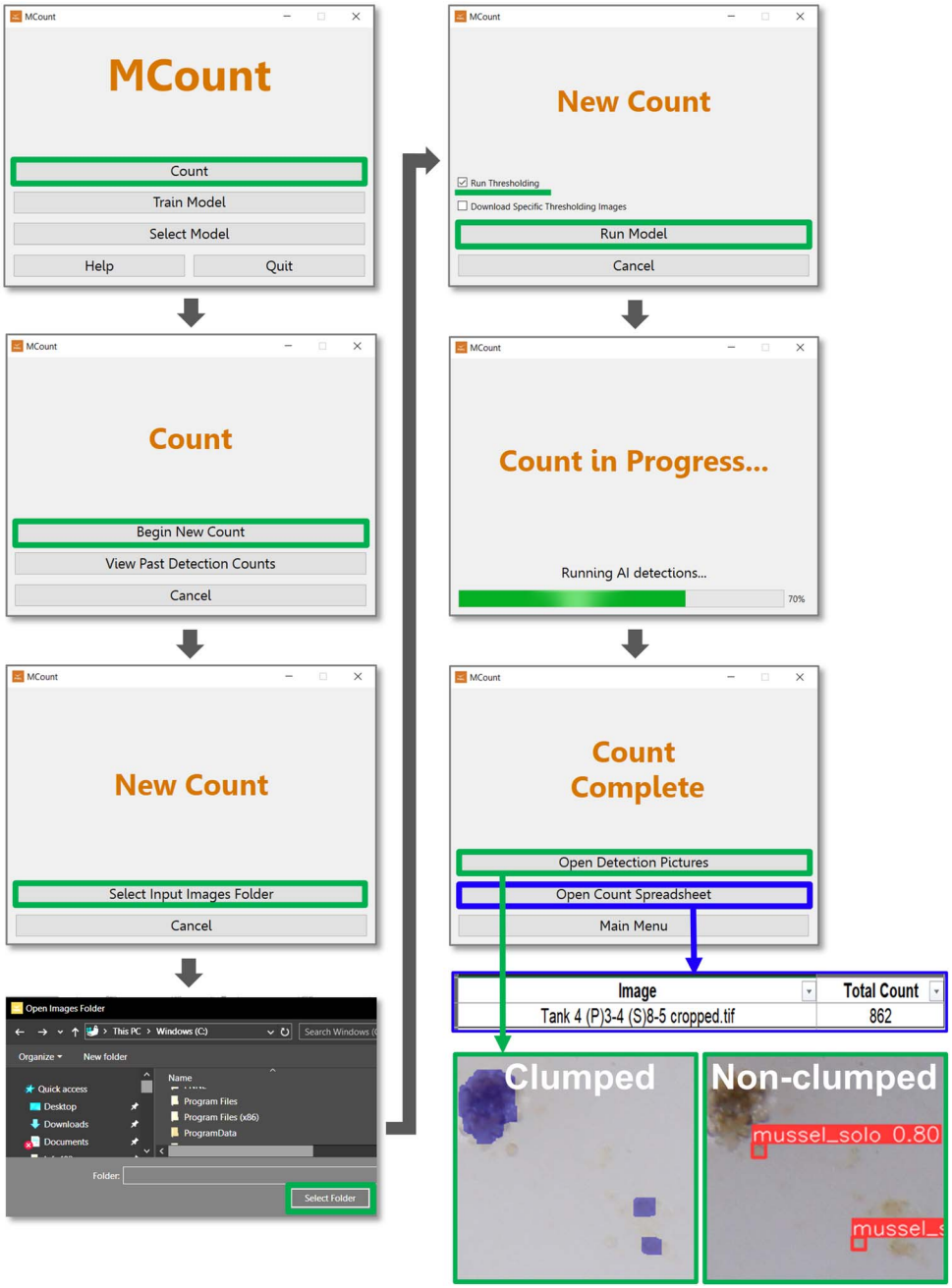

Fig. 5 Screenshots of the M-Count graphical user interface. The GUI is designed to be simple and user-friendly. A folder of images to be analyzed can be selected, and the application generates two analyzed image sets: one image set contains image tiles with object detection boxes overlaid on detected non-clumped mussel larvae, and one full image with an overlay of threshold detected clumped mussels (shown here as a blue overlay in the clumped image, bottom center). For quantitative analysis, M-Count also generates a spreadsheet that contains the total number of mussel larvae (split into non-clumped and clumped mussels) detected in each image in the input folder.

can switch between any models they have added by choosing them in the 'Select Model' dropdown menu.

M-Count performance

When compared with manual counting, M-Count features two primary performance benefits – high counting speed and high precision – while maintaining comparable accuracy. The accuracy of M-Count was quantified by comparing M-Count results with manual counting results. Manual counting is subject to human error; however, it is performed by trained observers and is the chosen basis from which to judge the performance of M-Count.

Training and evaluation of the YOLOv8 machine learning model for detection of individual mussel larvae was performed. A maximum F1 score of 0.76 was calculated at a confidence of 0.32. The full F1-confidence curve, along with other training and evaluation metrics, is shown in SI, Fig. 1. A confusion matrix is shown in SI, Fig. 2. These results show that the model

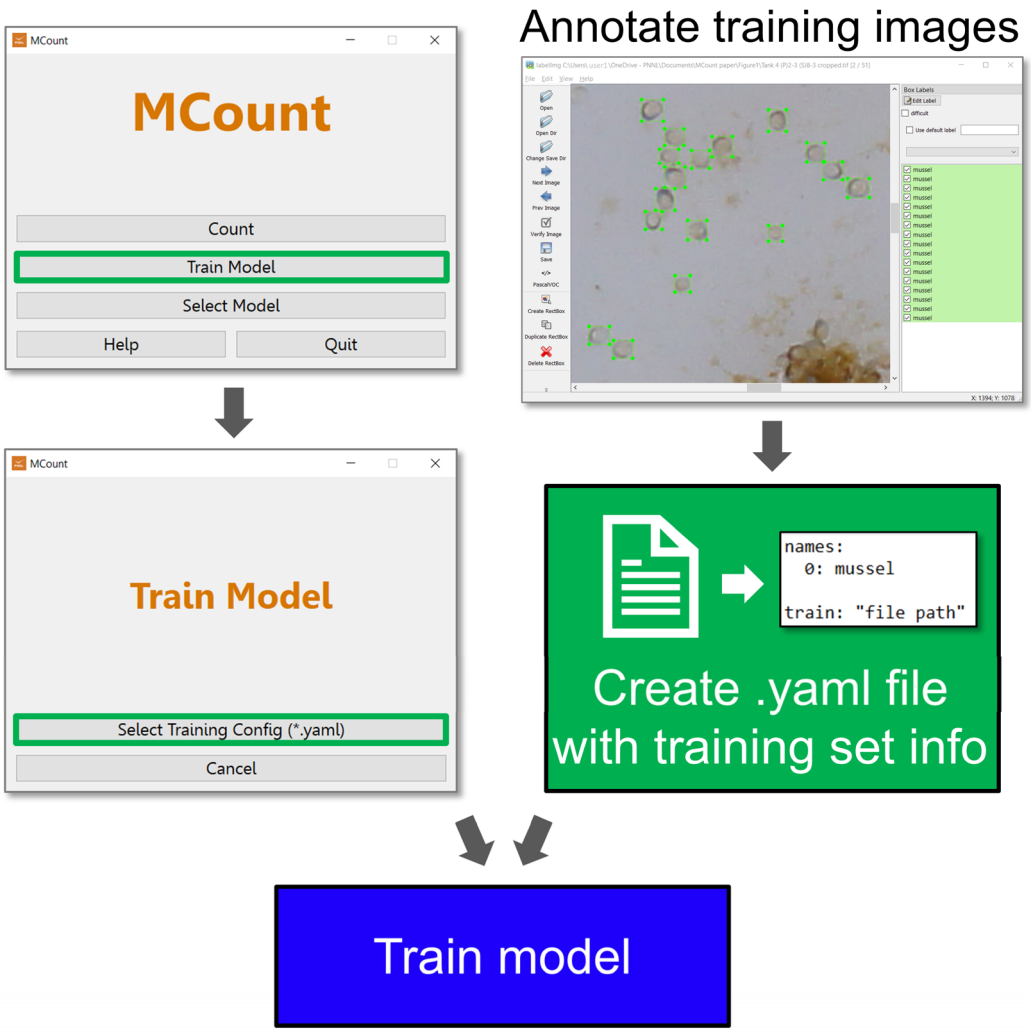

Fig. 6 The M-Count GUI can train a new model using annotated images. The images must be annotated using a separate software package such as LabelImg (upper right). The M-Count application trains the model, outputting the model as a PyTorch weights file that can later be selected as the model to use during the mussel detection process.

was trained sufficiently well to perform well with complementary thresholding analysis. These metrics will vary for other users of M-Count depending on the quality of images, quality of annotation, and features of the objects being detected.

As a more practical demonstration of M-Count, in Fig. 7 the results from the M-Count method and the manual counting method are compared according to the three categories of mussels: non-clumped mussels, clumped mussels, and total (non-clumped and clumped) mussels.

In Fig. 7A–C, the M-Count results (blue-filled circles) are co-plotted against the average manual count (averaged across five users) for each coupon of a set. Co-plotted with the M-Count results are the average manual counts for each coupon (orange-filled squares), the line of fit of which naturally forms a line with a slope of one (dotted black line). Error bars on the average manual counts show the maximum and minimum manual count for each coupon, highlighting the fact that undesirable user-to-user variation is inherent to manual counting. M-Count data points, on the other hand, do not have error bars because M-Count returns the same count value across all runs on a given image. The plots in Fig. 7A–C show that M-Count tends to slightly undercount non-clumped mussels and slightly overcount clumped mussels, but the total count produced by M-Count is centered around the line formed by the average manual count results.

For all three mussel quantities (non-clumped, clumped, and total mussels) in Fig. 7A–C, the M-Count results are fitted to a linear curve using the least squares LINEST function in Excel. The coefficients of determination, R^2, for non-clumped, clumped, and total mussels are 0.9208, 0.6956, and 0.6877 respectively. The coefficients of determination for individual users (whose counts are plotted against the average manual counts in SI, Fig. 3) fall within 0.9230–0.9946, 0.7713–0.9527, and 0.9387–0.9673 for non-clumped, clumped, and total mussels respectively and are listed by user in SI, Table 1.

The normalized M-Count residuals were also compared to the normalized average maximum manual count residuals (Fig. 7D–F). For non-clumped mussels, the manual count

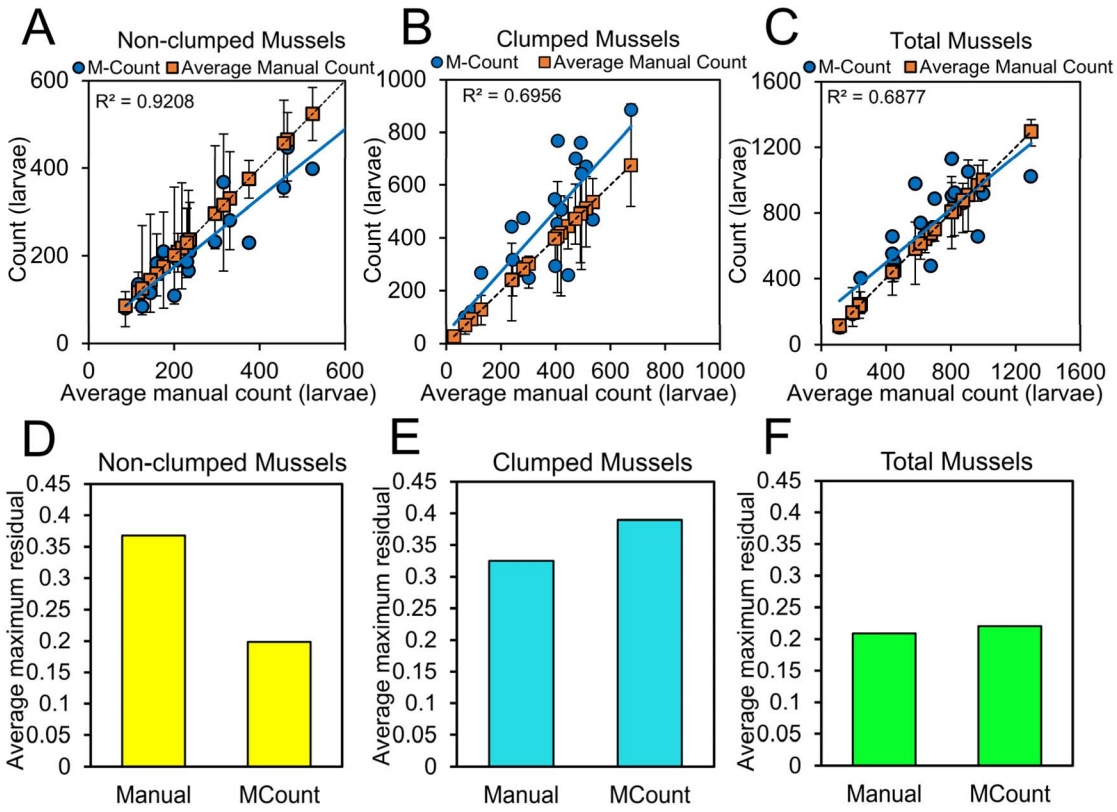

Fig. 7 Accuracy of M-Count compared to manual counting. (A–C) For non-clumped mussels, clumped mussels, and total mussels respectively, M-Count results (blue-filled circles) are plotted against the average manual result. The average manual result is also co-plotted (orange-filled squares), and error bars show the maximum and minimum manual result for each coupon. (D–F) For non-clumped, clumped, and total mussels respectively, the average maximum residuals for manual counters and M-Count are plotted. These averages are normalized by the average manual results for non-clumped, clumped, and total mussels, respectively. These results indicate that the accuracy of M-Count is comparable to manual counting.

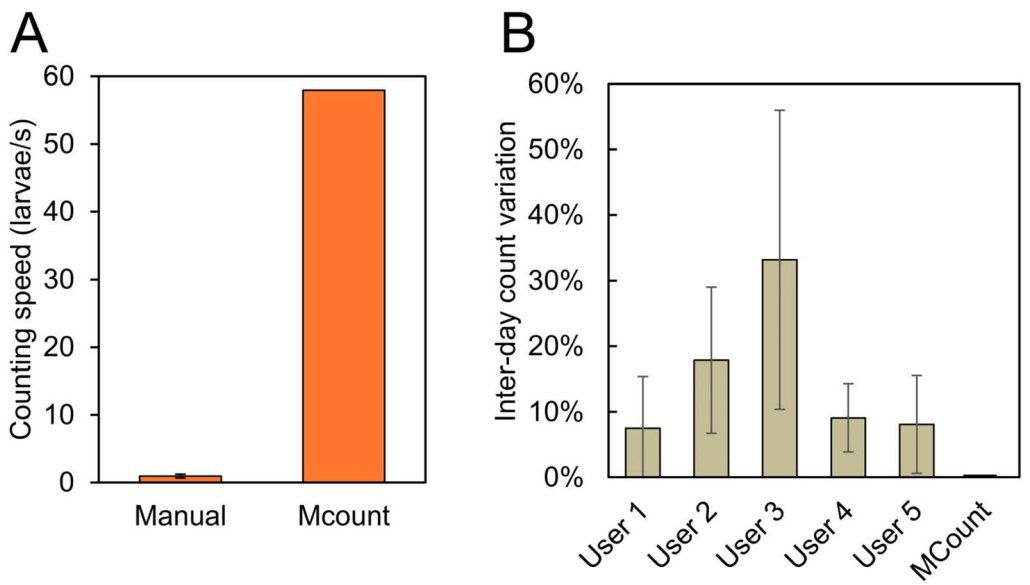

Fig. 8 Counting speed and consistency of the M-Count application compared to manual counting. (A) Graph of the counting speed of manual counting and M-Count automatic counting. M-Count performs the mussel larvae counting process 61× faster than manual counting. Error bars show ± one standard deviation. (B) Plot of the difference in counting results when the same coupons were counted twice by each user. Error bars show ± one standard deviation. Manual counting exhibits substantial differences in counting results on the same images when counted on different days. M-Count, on the other hand, provides the same result for a given set of images every time.

average maximum residual was 0.368 and the M-Count residual was 0.199. For clumped mussels, the manual count average maximum residual was 0.325 and the M-Count residual was 0.390. For total mussels, the manual count average maximum residual was 0.209 and the M-Count residual was 0.220.

A major motivation for the development M-Count was to increase the speed and consistency of counting mussel larvae. As shown in Fig. 8A, manual mussel larvae counting was completed at an average rate of 0.95 larvae per s. M-Count accomplished the same counting task at a rate of 58.0 larvae per s, 61× faster than manual counting.

Consistency of counting is also a major benefit when using M-Count. Human users lack precision in their counting skills, reporting different mussel counts for the same coupon counted on different days (Fig. 8B). Averaged across a set of coupons, individual human average day-to-day variation ranged from 7.50% to 33.2% depending on the user. The maximum day-to-day variation across the set of images ranged from 24.4% to 74.9%. M-Count provides the exact same results every time as long as the same trained model is used.

Discussion

The creation of M-Count was motivated by the need for a user-friendly platform to rapidly detect two categories of biofouling: distinct individuals ('non-clumped' mussel larvae) and amorphous groups of individuals ('clumped' mussel larvae). Specifically, a previously developed larvae settlement assay for determining surface antifouling performance requires that tens of thousands of larvae across a sample set be quantified. Even within such a single species fouling community, both non-clumped and clumped morphologies of fouling are prevalent. Machine learning and other automatic algorithms are well-suited to the task of detecting and quantifying fouling, but their implementation requires in-depth coding experience that many scientists and other stakeholders do not have. Other graphical user interface (GUI) programs exist but contain one or more of the following drawbacks: not designed to be quantitative, overly complex user interfaces, not open source, and, most importantly, do not contain the dual workflow necessary for quantifying both amorphous clumps and non-clumped individuals. In this article, we presented a simple yet versatile counting software, M-Count, which can be downloaded and run as a GUI application. M-Count uses a You Only Look Once (YOLO) v8 machine learning object detection model for detecting non-clumped individual mussel larvae and an automatic thresholding process for detecting clumped groups of larvae. The YOLOv8 model can be updated with new versions of YOLO as they are released, although that would require modification of the source code and re-training of the model.

In this effort, the method of object detection was chosen instead of alternative methods such as semantic segmentation, instance segmentation, or simple image classification because of the many small objects with few defining features present in the images. Early work attempted to break the image into tiny tiles and perform image classification on each tile, but this resulted in many of the mussel larvae being split and poorly quantified. Semantic and instance segmentation were also deemed poorly suited to the task because the exact boundaries of the mussels were not needed and because the mussels were relatively very small – roughly 20 px long by 10 px wide – and lacked defining geometries, making their exact boundaries difficult to determine. While small object segmentation is an area of focused study, both semantic and instance segmentation methods have difficulty in detecting small objects.[29,30] Object detection is better suited to quantifying individual objects without requiring that their exact boundary be detected. Thresholding was then chosen as a parallel method to capture mussel clumps, which varied greatly in size and shape and may have presented a challenge for instance or semantic segmentation. Other works have performed thresholding with object detection, but these studies typically used thresholding as a preliminary augmentation step to improve the accuracy of the model[31] or as a secondary stage[32] after object detection is complete. Other strategies combine multiple machine learning architectures,[33] but such efforts did not produce the type of user-friendly application required for widespread adoption.

M-Count is a user-friendly application that features increased larvae counting speed and precision compared to manual counting. The user menu button options (Fig. 5 and 6) are minimal because M-Count has been created for the specific (yet widely applicable) process of detecting clumped and non-clumped individual organisms that the machine learning model and thresholding algorithm have been designed to detect. An entire set of images can be analyzed in one session, and the application saves an Excel file with the larvae counts tabulated for each image, resulting in well-organized, quantified fouling data. M-Count is set up such that it can be trained to recognize any type of object within an image, which could have widespread applications in a variety of biological detection tasks such as field testing, maintenance scheduling, and ecological monitoring. In addition to effectively quantifying mussel larvae, M-Count could also effectively quantify other marine organisms that tend to exist as both clumped and non-clumped individuals, including adult mussels, barnacles, and colonial hydroids.

The goal of developing M-Count was to obtain a rapid, precise, mostly hands-off method for counting mussel larvae without sacrificing the accuracy of manual counting. The accuracy of M-Count is illustrated in Fig. 7, where the M-Count result is co-plotted with the average manual result, with error bars showing the maximum and minimum manual results for each coupon. M-Count had a moderate R^2 value, 0.688, for the total mussel count. A higher R^2 would be desirable, but a lower R^2 is an acceptable trade-off for the precision and speed increase offered by M-Count.

Another metric for comparing the accuracy of M-Count with manual counts is the maximum residual for each image (Fig. 7D–F). On average, the maximum M-Count residual was comparable to the maximum manual count residual. For non-clumped mussels, the average M-Count residual was lower than the average manual count residual. For clumped and total mussels, the average M-Count residual was only slightly higher

than the average manual count residual. This indicates that M-Count yields results within the distribution of manual counts.

M-Count does not need to be more accurate than manual counting to be useful; rather, its utility lies in being much faster and more precise without losing accuracy. The superior speed of M-Count is clear, as it demonstrated a 61× speed increase over manual counting (Fig. 8A), and most of the time requirement for M-Count is hands-off for the user. The precision of M-Count was demonstrably improved over human users, which varied by as much as 74.9% from day to day, whereas M-Count produced zero inter-day variation (Fig. 8B). Human users were susceptible to changes in their impression of the visual appearance of objects such as mussel larvae, which are small, transparent, and can be easily confused with other debris or surface asperities. M-Count, on the other hand, when provided with the task of analysing the same set of images multiple times, provided the same count results each time.

A variety of other deep learning Image analysis software packages with GUIs for non-experts are available. JustDeepIt[25] is an open-source deep learning software with a GUI and is perhaps the closest alternative to M-Count, but it is better suited for broad image classification, rather than for quantification of many small objects. Ilastik is a powerful machine learning tool developed for non-experts that contains multiple workflows for different image analysis requirements and provides an interactive training environment that reduces the time required for creating training data.[24] FastER is designed for high-throughput segmentation of cell-shaped objects, which are similar in appearance to mussels.[34] However, these and other image analysis packages are relatively complicated to operate and focus on classification and segmentation rather than providing readily accessible quantification of identified objects.[35–38] While object counting may be an available option in these packages, the accompanying workflows did not satisfy our requirements for clarity, simplicity, and ease-of-use. M-Count, on the other hand, prioritizes quantification of objects and simplicity of operation to provide a pragmatic biofouling quantification tool for a variety of users. M-Count was developed to address a specific need – to quantify thousands of mussel larvae that can take the form of isolated individuals and also be grouped together like masses of tissue – while remaining as simple as possible. M-Count may also serve as a useful tool for other image quantification tasks.

M-Count in its current incarnation has some limitations. Despite being trained to detect non-clumped mussels on a variety of substrates, M-Count is currently only capable of analysing clumped mussels on white or off-white substrates because the thresholding method has not been equipped with a way to automatically determine the optimal color thresholding limits. Future versions of M-Count should incorporate automatic threshold optimization, background color detection, and/or perhaps a machine learning instance segmentation algorithm. Instance segmentation was tested as method for quantifying clumped larvae in this effort, but did not deliver acceptable results, likely due to the small pixel dimensions of the mussel larvae, lack of diverse defining characteristics (*e.g.*, shape and color variations) and wide range of clump sizes. The current version of M-Count is also lacking in its coefficient of determination when plotted against average manual counting values; the R^2 for M-Count could likely be improved by further training the model with higher quality training data, as the annotations can be ambiguous due to the uncertain distinction between non-clumped mussels and clumped mussels. M-Count performance could also be improved by further optimizing the larvae imaging process, which currently relies on a coarse manual focus step that produces better focus than automatic focusing but leaves room for error that can cause larvae to be slightly out of focus. Future model training should be conducted by using multiple humans to create annotated training data that averages the biases of multiple humans.

Conclusions

Herein we presented M-Count, an application for the detection of mussel larvae to quantify biofouling. M-Count uses two detection processes in parallel: object detection using YOLOv8 for quantification of individual non-clumped mussel larvae; and color thresholding for the quantification of amorphous clumps of mussel larvae. M-Count is nearly as accurate as manual counting but accomplishes the task over 60× faster and with exact precision. Operation of M-Count requires very little coding knowledge and generates an organized spreadsheet containing detected mussel larvae quantities for each image in a selected folder. M-Count can be trained to detect a variety of objects in images and is well-suited to quantification of marine biological settlement and fouling, which are often comprised of both individuals and amorphous films that require parallel and unique detection processes.

Author contributions

Conceptualization: T. B. L, J. D. D., C. J. L.; formal analysis: L. W. M, N. N., T. B. L.; funding acquisition: R. S. A., G. T. B., C. J. L.; investigation: L. W. M., N. N., T. B. L., C. N. H., J. D. D., A. V., A. K., W. C.; methodology: L. W. M., N. N.; project administration: G. T. B., R. S. A., C. J. L.; software: L. W. M., N. N.; supervision: C. J. L.; visualization: L. W. M., N. N., T. B. L., C. J. L.; writing/original draft preparation: L. W. M, N. N., T. B. L., C. J. L.; writing/review and editing: T. B. L., C. N. H., C. J. L.

Conflicts of interest

There are no conflicts to declare.

Data availability

The raw data is available at https://data.mendeley.com/datasets/9cs8pnhkxj/1.[39] The M-Count application is available at https://github.com/pnnl/MCount.[28]

Supplementary information (SI) is available. See DOI: https://doi.org/10.1039/d5ay01759a.

Acknowledgements

This work was sponsored by the US Department of Energy, Office of Energy Efficiency and Renewable Energy (US DOE-EERE) Water Power Technologies Office through the Pacific Northwest National Laboratory. Pacific Northwest National Laboratory is a multi-program national laboratory operated by Battelle for the United States Department of Energy under Contract DE-AC05-76RL01830.

References

1 J. Bannister, M. Sievers, F. Bush and N. Bloecher, *Biofouling*, 2019, **35**, 631–648.
2 H. Habbouche, H. Rashid, Y. Amirat, A. Banerjee and M. Benbouzid, *Ocean Eng.*, 2024, **312**, 119283.
3 L. Delauney, C. Compère and M. Lehaitre, *Ocean Sci.*, 2010, **6**, 503–511.
4 Q. Kong, S. Zheng, X. Yan, L. Zheng, Y. Yang and Y. Li, *Ocean Eng.*, 2024, **309**, 118546.
5 A. Lindholdt, K. Dam-Johansen, S. M. Olsen, D. M. Yebra and S. Kiil, *J. Coat. Technol. Res.*, 2015, **12**, 415–444.
6 A. Farkas, N. Degiuli and I. Martić, *Int. J. Nav. Archit. Ocean Eng.*, 2021, 102–114, DOI: 10.1016/j.ijnaoe.2020.12.005.
7 A. Coraddu, S. Lim, L. Oneto, K. Pazouki, R. Norman and A. J. Murphy, *Ocean Eng.*, 2019, **176**, 65–73.
8 P. Gupta, A. Rasheed and S. Steen, *Ocean Eng.*, 2022, **254**, 111094.
9 K. Koren and C. M. McGraw, *ACS Sens.*, 2023, **8**, 2432–2439.
10 H. Hong, J. Lv, A. Deng, Y. Tang and Z. Liu, *J. Environ. Manage.*, 2024, **357**, 120766.
11 S. M. Pennell, T. B. LeFevre, J. Bennett, W. Chouyyok, J. D. Daddona, R. S. Addleman, C. J. Larimer and G. T. Bonheyo, *Biofouling*, 2025, **41**, 300–311.
12 S. Dobretsov and D. Rittschof, *Int. J. Mol. Sci.*, 2023, **24**, 6531.
13 J. A. Callow and M. E. Callow, *Nat. Commun.*, 2011, **2**, 244.
14 Z. Q. Zhao, P. Zheng, S. T. Xu and X. Wu, *IEEE Transact. Neural Networks Learn. Syst.*, 2019, **30**, 3212–3232.
15 S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022, **44**, 3523–3542.
16 C. Larimer, E. Winder, R. Jeters, M. Prowant, I. Nettleship, R. S. Addleman and G. T. Bonheyo, *Anal. Bioanal. Chem.*, 2016, **408**, 999–1008.
17 R. Parthasarathy, *Nat. Methods*, 2012, **9**, 724–726.
18 A. Ming and H. Ma, *Presented in part at the Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, Amsterdam, The Netherlands, 2007.
19 T. Lindeberg, *Int. J. Comput. Vis.*, 1998, **30**, 79–116.
20 M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald and E. Muharemagic, *J. Big Data*, 2015, **2**, 1.
21 J. Signor, F. Schoefs, N. Quillien and G. Damblans, *Ocean Eng.*, 2023, **288**, 115928.
22 J. Redmon, S. Divvala, R. Girshick and A. Farhadi, 2016.
23 C. S. Chin, J. Si, A. S. Clare and M. Ma, *Advances in Computer Communication and Computational Sciences*, Singapore, 2019.
24 S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht and A. Kreshuk, *Nat. Methods*, 2019, **16**, 1226–1232.
25 J. Sun, W. Cao and T. Yamanaka, *Front. Plant Sci.*, 2022, **13**, 01–09.
26 T. B. LeFevre, J. D. Daddona, W. Chouyyok, G. King, S. M. Pennell, A. E. Plymale, S. Akins, L. W. Miller, N. Nune, C. N. Hermanson, G. T. Bonheyo, C. Larimer and R. S. Addleman, *Biofouling*, 2025, **41**, 783–797.
27 R. S. Addleman, C. J. Larimer, C. A. Barrett, G. T. Bonheyo, R. T. Jeters and E. M. Winder, *US Pat.*, 14/839471B2, 2019.
28 L. W. Miller and N. Nune, Github, 2024, https://github.com/pnnl/MCount.
29 S. Sang, Y. Zhou, M. T. Islam and L. Xing, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023, **45**, 6289–6306.
30 C. Wang, Y. Ji, Y. Meng, Y. Zhang and Y. Zhu, *arXiv*, 2025, preprint, arXiv:2509.03002, DOI: 10.48550/arXiv.2509.03002.
31 M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec and K. Cho, *arXiv*, 2019, preprint, arXiv:1902.07296, DOI: 10.48550/arXiv.1902.07296.
32 N. Kumar, *Multimed. Tool. Appl.*, 2018, **77**, 19139–19170.
33 K. Drid, M. Allaoui and M. L. Kherfi, *Cham*, 2020, 290–296.
34 O. Hilsenbeck, M. Schwarzfischer, D. Loeffler, S. Dimopoulos, S. Hastreiter, C. Marr, F. J. Theis and T. Schroeder, *Bioinformatics*, 2017, **33**, 2020–2028.
35 S. Dillavou, J. M. Hanlan, A. T. Chieco, H. Xiao, S. Fulco, K. T. Turner and D. J. Durian, *Sci. Rep.*, 2024, **14**, 14281.
36 D. Ershov, M.-S. Phan, J. W. Pylvänäinen, S. U. Rigaud, L. Le Blanc, A. Charles-Orszag, J. R. W. Conway, R. F. Laine, N. H. Roy, D. Bonazzi, G. Duménil, G. Jacquemet and J.-Y. Tinevez, *Nat. Methods*, 2022, **19**, 829–832.
37 T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox and O. Ronneberger, *Nat. Methods*, 2019, **16**, 67–70.
38 A. M. Lucas, P. V. Ryder, B. Li, B. A. Cimini, K. W. Eliceiri and A. E. Carpenter, *Mol. Biol. Cell*, 2021, **32**, 823–829.
39 T. B. LeFevre, 2025, https://data.mendeley.com/datasets/9cs8pnhkxj/1.

