BMC Bioinformatics

RESEARCH Open Access

petiteFinder: an automated computer vision tool to compute Petite colony frequencies in baker's yeast

Christopher J. Nunn[1*], Eugene Klyshko[1,2] and Sidhartha Goyal[1,3]

*Correspondence:
chris.nunn@utoronto.ca

[1] Department of Physics, University of Toronto, Toronto, ON M5S 2W9, Canada
[2] Department of Chemical and Physical Sciences, University of Toronto Mississauga, Mississauga, ON L5L 1C6, Canada
[3] IBBME, University of Toronto, Toronto, ON M5S 3G9, Canada

Abstract

Background: Mitochondrial respiration is central to cellular and organismal health in eukaryotes. In baker's yeast, however, respiration is dispensable under fermentation conditions. Because yeast are tolerant of this mitochondrial dysfunction, yeast are widely used by biologists as a model organism to ask a variety of questions about the integrity of mitochondrial respiration. Fortunately, baker's yeast also display a visually identifiable Petite colony phenotype that indicates when cells are incapable of respiration. Petite colonies are smaller than their Grande (wild-type) counterparts, and their frequency can be used to infer the integrity of mitochondrial respiration in populations of cells. Unfortunately, the computation of Petite colony frequencies currently relies on laborious manual colony counting methods which limit both experimental throughput and reproducibility.

Results: To address these problems, we introduce a deep learning enabled tool, *petiteFinder*, that increases the throughput of the Petite frequency assay. This automated computer vision tool detects Grande and Petite colonies and computes Petite colony frequencies from scanned images of Petri dishes. It achieves accuracy comparable to human annotation but at up to 100 times the speed and outperforms semi-supervised Grande/Petite colony classification approaches. Combined with the detailed experimental protocols we provide, we believe this study can serve as a foundation to standardize this assay. Finally, we comment on how Petite colony detection as a computer vision problem highlights ongoing difficulties with small object detection in existing object detection architectures.

Conclusion: Colony detection with *petiteFinder* results in high accuracy Petite and Grande detection in images in a completely automated fashion. It addresses issues in scalability and reproducibility of the Petite colony assay which currently relies on manual colony counting. By constructing this tool and providing details of experimental conditions, we hope this study will enable larger-scale experiments that rely on Petite colony frequencies to infer mitochondrial function in yeast.

Keywords: Computer vision, Object detection, Mitochondrial respiration, Baker's yeast, Colony morphology

Background

Eukaryotic cells have numerous mitochondria containing multiple copies of their genome, mitochondrial DNA (mtDNA). In most eukaryotes, mtDNA encodes a subset of proteins involved in oxidative phosphorylation, while the remainder is encoded in nuclear DNA (nDNA). *Saccharomyces cerevisiae*, or baker's yeast, is a widely used model organism to study the interdependence of mtDNA and nDNA. This is largely due to the dispensability of respiration in yeast under fermentable conditions, which allows cells to propagate without mitochondrial/nuclear-encoded genes involved in respiration. In particular, tolerance to mutations in mtDNA, which would otherwise be fatal to cells in other model systems, enables the study of mtDNA dynamics in yeast, including interesting questions regarding mtDNA maintenance, selection, and mutation.

Since the discovery of the Petite colony phenotype in yeast, this phenotype has played a central role in studying the interplay of nDNA and mtDNA [1, 2]. As the name suggests, the Petite phenotype is characterized by small, more translucent colonies compared to Grande (wild-type) colonies under fermentable conditions. These colonies are smaller and have lower cell density due to the inability to switch to respiration once fermentable carbon sources have been consumed as the colony spreads on an agar surface. The ease of distinguishing Petite from Grande colonies visually, with nothing but a Petri dish and its agar surface, is part of what has made the Petite phenotype popular as an indicator of mitochondrial function.

Measurements of Petite colony frequencies have found various uses in the literature. By transplanting mtDNA into yeast strains with various genetic backgrounds, changes in Petite frequencies resulting from this transplantation have been used to understand what effect variants in nuclear genes and mtDNA structures have on mtDNA stability [3, 4]. Petite frequencies have also been shown to be useful as a screen to determine what nuclear genes are responsible for mtDNA recombination that affect mtDNA integrity [5–8], and chemicals that induce mitochondrial maintenance, or mitophagy [9]. Petite frequency measurements have been integrated into a powerful approach that discovered numerous genes involved in mitochondrial biogenesis [10]. Furthermore, mating cells with wild-type and mutated mtDNAs and measuring the fraction of progeny that themselves are Petite provides access to the selection and dynamics of competing mtDNAs [11, 12].

While the Petite frequency assay is both accessible and versatile, current methods of Petite colony identification are laborious. These methods often involve manual annotation of Petri dishes under growth conditions where Petite/Grande colony differences are perceptible or replica plating onto non-fermentable media (see for example the methods in [4, 9, 10]). Annotation under different growth conditions across laboratories, which can influence Petite versus Grande colony characteristics, also hinders reproducibility.

Although no automation tool designed specifically for the Petite frequency assay exists to our knowledge, machine learning algorithms have been widely used to detect and count colonies in images to improve the scalability of other colony morphology assays [13–18]. Unsupervised/semi-supervised versions of these methods, such as the popular AutoCellSeg [15], OpenCFU [13], and CellProfiler [14], primarily perform colony segmentation (separating colonies from background). While these tools do perform admirably under ideal conditions (uniform background, lack of imaging artifacts), they can

struggle to segment colonies under some experimental conditions. The consequence is that many of these unsupervised/semi-supervised approaches require user input to help characterize colonies (e.g. max/min size, eccentricity, and plate location in [13–15]). The same user input requirements exist in supervised approaches, where the preprocessing of colonies based on size/eccentricity is common [16, 17]. Unfortunately, as experimental conditions change, user defined parameters often need to be tuned between images which reduces pipeline throughput. Furthermore, in all of the aforementioned approaches except [17], colony classification is left entirely to the users; they must engineer and select features before applying their own supervised classifier to distinguish between different classes of segmented colonies.

In the present study, we develop an automated computer vision tool that addresses the limited scalability and reproducibility of the Petite frequency assay. Our tool, *petiteFinder*, detects Petite/Grande colonies and computes Petite frequencies from scanned images of Petri dishes. Unlike existing colony detection approaches, *petiteFinder* requires no mandatory user input and performs both segmentation and the classification of colonies in one step. *petiteFinder* is built on the open-source object detection toolbox MMDetection [19] and a computer vision library SAHI [20], which improves small object detection via image slicing. It uses a Faster RCNN [21] object detection architecture built on top of a ResNet-50 backbone [22] coupled with a feature pyramid network (FPN) [23]. Besides colony detection, *petiteFinder* also enables users to amend colony classification predictions with a user-friendly interface.

Methods

Acquisition of the Grande/Petite colony dataset

To generate a dataset for this tool, we performed plating experiments where yeast colonies were grown on an agar surface in Petri dishes. We conducted these experiments with fermentable media (both synthetic and complete) that can be constructed to support baker's yeast strains with any auxotrophies (see details in "Media and growth conditions" section). The only constraint on the media was a carbon composition of 3% glycerol and 0.1% glucose that allows for robust visual identification of Grande/Petite colonies after 3–5 days of growth on agar at 30 °C [4]. With this carbon composition, Petite colonies appear smaller and more translucent than their Grande counterparts on an agar surface. On synthetic media (SC-ura-trp), we mated a Grande strain and a variety of Petite strains. We plated this mixture to identify whether or not the progeny of mated parents were Petite or Grande. On complete media (YPADG), we plated cells from a single haploid strain to identify spontaneous transitions from Grande to Petite cells. These experiments yielded Petri dishes with Petite colony frequencies ranging from ~0 to 0.9.

Following these experiments, Petri dishes were scanned with an image scanner, six at a time (see Petri dish imaging, Fig. 1a). Images were then cropped to contain an individual petri dish with the aid of a 3D printed scanner insert that fixed their positions and reduced refraction due to adjacent plates. These images were manually annotated using the LabelMe package [24], where bounding boxes were drawn around Grande and Petite colonies and assigned to their corresponding labels by an expert in Petite/Grande colony identification. The resulting labeled dataset is a collection of 83 petri dish images, 59 in

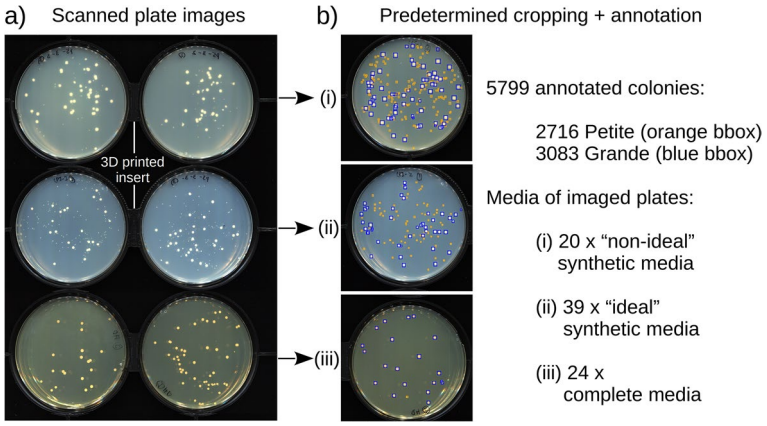

Fig. 1 An overview of the labeled dataset. **a** A composite image showing Petri dishes containing synthetic (top 4 Petri dishes) and complete media (bottom 2 Petri dishes) with yeast colonies on their surface. Six petri dishes at a time were placed in a 3D printed insert (black structure bordering petri dish images) and scanned on a computer scanner bottom-up. The synthetic media was SC-ura-trp and the complete media was YPADG (both 0.1% glucose and 3% glycerol carbon source). **b** Individual plate images were cropped out of the large image, and 83 of these images were annotated using the LabelMe annotation tool [24]. Grande and Petite colonies are indicated by blue and orange bounding boxes, respectively. Experimental variation in media preparation/pouring of plates produces diffuse low-quality scans (non-ideal) and sharper plate images (ideal) as shown by example plate crops (i) and (ii) for synthetic media. Plate crop (iii) is complete media that has uniform agar opacity across all images

synthetic media and 24 in complete media. There are 5799 bounding box annotations encompassing colonies, with 2716 Petite and 3083 Grande labels in total. Variation in agar concentrations within synthetic media poured into plates in this dataset produced 20 images we consider "non-ideal" due to the diffuseness of colonies, and 39 images we consider "ideal" (Fig. 1b). Images of complete media were largely consistent and did not exhibit this variability. All types of images were used as training data. This experimental variation, both at the level of media and the types of experiments that generated these plates, is further augmented during training to produce a robust computer vision model. Images were split into 60% training, 20% validation, and 20% test sets.

Model architecture

The detection and classification of colonies is an object detection problem that is commonly solved in computer vision through convolutional neural networks (CNN) [25, 26]. Object detection frameworks built on region-based convolutional neural networks (RCNNs) remain among top performers in object detection competitions [27]. However, small object detection is challenging for most RCNN approaches, where lower-level features (in early layers of the network) of small objects are lost through coarse-graining via convolutions and pooling. The Faster-RCNN architecture [21] coupled with a feature pyramid network (FPN) has been shown to improve performance in small object detection [23]. Specifically, the FPN achieves high accuracy detection of small objects by propagating high-level features down to lower layers in the network and sharing feature maps of multiple scales with the region proposal component of the detector.

Despite adopting the architectural advantages of FPN for small object detection, a Faster-RCNN ResNet-50 FPN pipeline performed poorly in Petite colony detection on whole images of Petri dishes. Petite colonies were largely undetected in test images,

although Grande colonies were appropriately localized. We attributed this to the initial downsampling of images (1024 × 1024) we performed to fit models on consumer GPUs. This downsampling can eliminate Petite colonies in images due to their small size. To address this poor performance, we employed SAHI [20], a computer vision package that performs model inference through image slicing. Instead of whole images, small slices of images are passed through the Faster-RCNN ResNet-50 FPN architecture, which increases the relative size of colonies to the background in the network. Following object detection on all image slices from sliding windows over the entire image, slice predictions are merged in a greedy non-maximal merging algorithm (NMM). This algorithm operates like non-maximal suppression (NMS), but instead of eliminating bounding boxes, it simply merges them above a particular intersection over smaller area (IOS) threshold. The final output of the model operating on an image is a collection of predicted colony bounding boxes with Grande/Petite classification probability (Fig. 2).

Model training

Network weights were initialized with a network pre-trained on the Microsoft Common Objects in Context (COCO) dataset [28] with 80 classification categories. Network weights were frozen from stage 1 to the beginning of the ResNet-50 backbone (indicated by solid blue blocks in Fig. 2). The rest of the network was trained for 17 epochs with a batch size of 50 through stochastic gradient descent with a learning rate of 0.001, momentum 0.9, and weight decay of 0.0001. We used a learning rate scheduler

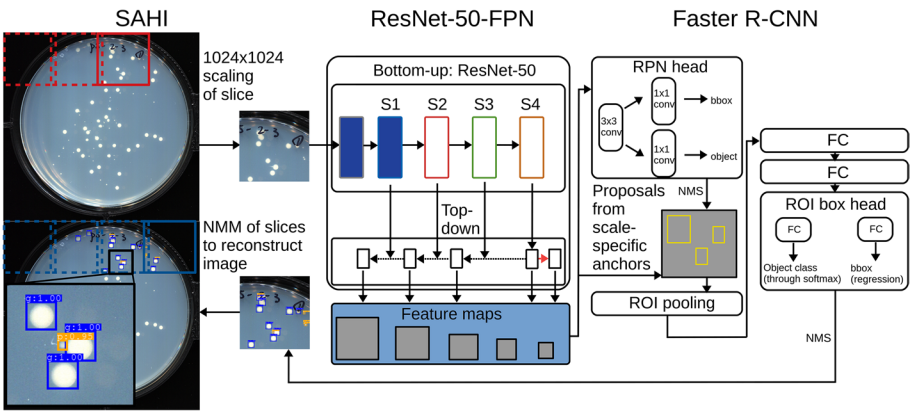

Fig. 2 The architecture of the object detection pipeline. The entire plate image is first sliced into cropped images (red boxes) using a sliding window with the SAHI package. The image slice is scaled to (1024 × 1024) regardless of its size and is passed into a ResNet-50-FPN convolutional neural network backbone. Here, Sx represents the convolutional stage of the ResNet-50 network. Solid blue indicates stages that were frozen during training. Convolution block outputs are rescaled (dotted arrow), undergo element-wise addition (intersection of solid and dotted arrows), and in one case, undergo max pooling (red arrow). This is the feature pyramid underlying the FPN (feature pyramid network) nomenclature. The output of this step is a collection of feature maps of various scales (resolutions) and semantic values. All feature maps are passed into the Faster R-CNN object detection architecture. In this architecture, the region proposal network (RPN), which is fully convolutional, generates region proposals that are likely to contain objects. These proposals are then passed to a final regressor and classifier which outputs a predicted bounding box and probability score for the class of the predicted object. Bounding boxes per image slice are filtered through non-maximal suppression (NMS). Finally, bounding box predictions on all image slices are merged into the final image using a non-maximum merging algorithm (NMM). The final output is a set of predicted bounding boxes on the entire image with associated Grande/Petite class probabilities (lower left image and zoomed portion)

with step-based decay, a linear warmup of 500 iterations, a warmup ratio of 0.001, and decay steps of factor 10 at 8 and 11 epochs. Images fed to the network during training were (512 × 512) image crops of Petri dishes upsampled to (1024 × 1024) through bilinear interpolation. The crop size was selected so that objects in cropped images were of comparable size to objects from the COCO dataset. This step was important as frozen network weights were trained on objects of much larger relative size and correspond to these scale-specific features. Extensive data augmentation was also applied to training images, including random flips, 0.8 to 1.0 scale crops, and photometric distortions. These distortions included: random changes of brightness, contrast, saturation and hue, colour conversions between BGR and HSV, and randomly swapping colour channels (see details in MMDetection documentation: [19]).

The training parameters (including learning policies and optimizers) were selected following an extensive hyperparameter grid search on the validation image set. Once optimal hyperparameters had been selected for the network, we performed a hyperparameter grid search for SAHI. Bounding boxes were merged through NMM if they had a confidence score above 0.6 and had matching class predictions with an IOS of greater than 0.5.

Model evaluation

During model finetuning, model performance was evaluated through the mean average precision metric with a bounding box intersection over union threshold (IOU) of 0.5 (mAP@0.5) on the validation set. Mean average precision is the area under the precision-recall curve across all bounding box confidence scores. Precision $= \frac{TP}{TP+FP}$ and recall $= \frac{TP}{TP+FN}$ with TP, FP, and FN, being true positives, false positives, and false negatives, respectively. For SAHI fine-tuning, we optimized parameters on the absolute error between predicted and ground truth Petite frequencies on images. However, this yielded similar parameters and performance to optimizing SAHI on mAP@0.5. For model performance on the test set, we report the mean average precision over IOUs ranging from 0.5 to 0.95 in 0.05 increments (mAP@0.5:0.95), alongside mAP@0.5, and mAP@0.75, which are commonly used in object detection competitions. We also show the Grande and Petite precision and recall. Finally, we compared predicted and ground truth Petite frequencies in the test set to evaluate real-world performance.

Pipeline implementation

petiteFinder is implemented in Python 3.7.11 and uses the MMDetection package [19] to build, train, and test the model. Training/testing was performed on an Nvidia GTX 1070 and GTX 1080 GPU with CUDA toolkit 11.3.1. It also uses SAHI [20] which relies on MMDetection to perform sliced predictions. Before images are passed through the pipeline, slice heights and widths are determined by the image resolution equation, $S_W = S_H = \sqrt{I_w I_H S_A}$, where S_W, S_H, I_w, I_H are slice and image widths and heights, respectively. S_A is the relative area of the image slices to the area of the training images, which for our training set was $S_A = \frac{512 \times 512}{2376 \times 2288}$. Assuming a user provides images of Petri dishes that fill the frame, this equation computes image slices with colony sizes relative to the background that are comparable to the training images. Under circumstances where colonies differ significantly in size from the training data, an alternative equation,

$S_W = S_H = \frac{GD_I}{GD_t} \times 512$ can be optionally used to compute image slices. In this equation, GD_I represents the typical diameter of a Grande colony provided by a user, GD_t the typical Grande diameter from the training data, and 512 is the size in pixels of the training image slices. This equation serves to enforce the same Grande size to slice size ratio as the training data, but is only necessary under extreme circumstances where adherence to our experimental conditions are grossly violated (See Additional File 1: Appendix C).

Users of *petiteFinder* have the option to output a CSV of Petite frequencies, visually annotated images with bounding boxes and scores, and a COCO formatted JSON file of annotations. There is also an option to amend *petiteFinder* annotations with a GUI that enables users to explore predictions and add/remove bounding boxes around colonies (see Fig. 3).

Experimental details and best practices

Petri dish imaging

Six 100 mm Petri dishes were scanned at a time, bottom-up, on an Epson V370 photo scanner at 600 DPI. The scanner was equipped with a custom 3D printed insert that dishes were placed into on the scanner surface and a black felt backing above the imaging surface. The 3D printed insert fixes Petri dish location for ease in cropping and reduces refraction from neighbouring Petri dishes which cause imaging artifacts. The black felt backing produced a dark background in scans which increased contrast between colonies and media in the images. Individual Petri dishes were cropped from the initial scan to fill the frame, resulting in (2376 × 2288) resolution images per petri dish. For best performance, images should be cropped so that Petri dishes fill the frame, as calculations of slice size are performed in the pipeline under this assumption. It is also recommended

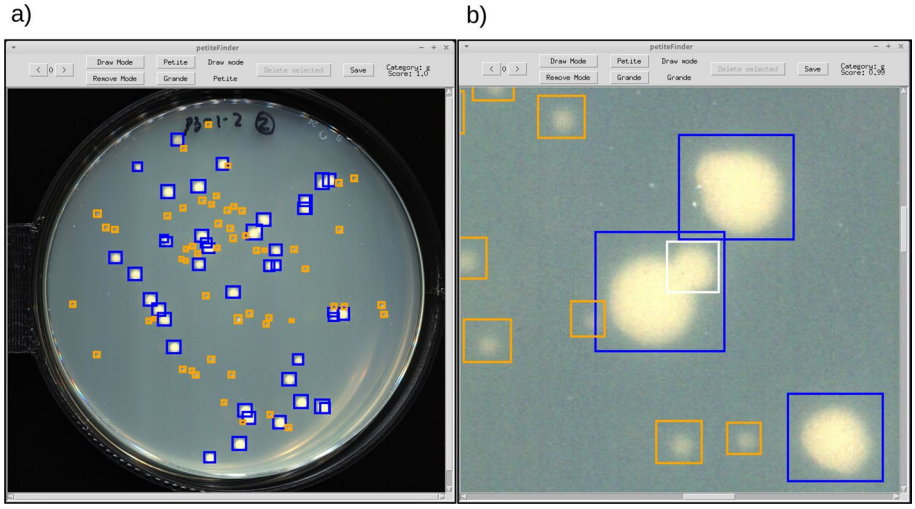

Fig. 3 An example of the interface in the amend function within *petiteFinder*. **a** The default view upon opening the GUI. Orange and blue boxes are Petites and Grandes, respectively. Arrow buttons in the top left can be used to move through images that have predictions. Hovering over colonies with a cursor reveals their class and probability score in the upper right. **b** An example where a user is zooming into an image to add a new Grande bounding box that is being drawn in white in the center of the frame. Upon finishing the drawing movement with the mouse, the bounding box switches to the appropriate colour (blue) in this case. Functions also exist to remove bounding boxes after being selected with the cursor

to have a resolution of at least (1000 × 1000) per Petri dish image to obtain comparable results to those reported in this study for similar sized colonies.

Media and growth conditions

Yeast colonies were grown on both complex and synthetic media to capture potential imaging variability in the most widely used growth media. The complex media was YPADG (1% yeast extract, 2% bacto-peptone, 0.1% glucose, 3% glycerol, 0.072% adenine hemisulfate). Synthetic media was SC-ura-trp (0.67% bacto yeast nitrogen base w/o amino acids, 0.1% glucose, 3% glycerol, 0.2% dropout powder lacking uracil and tryptophan). The shared characteristic across both media types that enables Petite identification is the carbon composition with reduced glucose which causes Petites to appear smaller and more translucent after 3–5 days of growth at 30 °C [4]. Before plating, liquid cultures were grown at 30 °C in a linear shaking water bath, while solid media growth took place in a forced air incubator at 30 °C. Across the labeled dataset, images were taken in a range from 3 to 5 days after growth, resulting in varying relative sizes of Petite/Grande colonies. However, Petites were discernible by eye after this timeframe in all images.

Yeast strains

All strains used in Petite frequency experiments were in the W303 background. Yeast cultured on SC-ura-trp (0.1% glucose, 3% glycerol carbon source) were matings between W303 *MATa leu*2-3,112 *can*1-100 *ura*3-1 *ade*2-1 *his*3-11,15 and W303 *MATα leu*2-3,112 *can*1-100 *ade*2-1 *his*3-11,15 *trp*1-1. Yeast cultured on YPADG was a haploid strain W303 *MATa leu*2-3,112 *can*1-100 *ura*3-1 *ade*2-1 *his*3-11,15.

Predictions with published semi-supervised methods

OpenCFU

Predictions on the test image set were performed with OpenCFU 3.9.1. Automatic bilateral thresholding was used alongside the auto outlier filter (default threshold of 30) and auto ROI masking. Minimum and maximum colony radii were 4 and 44 px, respectively. These radii correspond most closely to the minimum and maximum colony areas of 40 px^2 and 6000 px^2 used in our finely tuned semi-supervised pipeline (detailed in Additional File 1: Appendix A) to ensure a fair comparison.

CellProfiler

Predictions on the test image set were performed with CellProfiler 4.2.4. As a pipeline, we used a suggested yeast colony classification protocol [14] as a base, and modified it to provide a fair comparison with the other semi-supervised methods. Illumination correction with default parameters was applied to each color channel, and results were then combined, followed by alignment and masking. A binary mask of the plate (excluding its boundaries) was obtained through the resizing of a provided template image to fit the plate shape in our test set. During segmentation, a minimum and maximum area of 40 px^2 and 6000 px^2 were used to filter yeast colonies, similar to the other semi-supervised methods. Initially we applied the same eccentricity filtering (< 0.9) as in our semi-supervised method. However, the results demonstrated a very

low segmentation precision. Results were dramatically improved by using a lower eccentricity threshold of < 0.6. The classification of the colonies within CellProfiler was based on a colony size threshold of 1000 px^2 between Petite and Grande bins.

Results

Pipeline performance

To evaluate the performance of the pipeline, *petiteFinder* was applied to the images in the test set and compared against ground truth annotations. The test set consists of 17 petri dish images (4 YPADG, 4 SC-ura-trp nonideal, 9 SC-ura-trp ideal) with 1327 ground truth colonies. Of these 1327 colonies, 602 are Grande and 725 are Petite. On the test set, *petiteFinder* achieved a mean average precision at IOU $= 0.5$ of 0.96. Grande colonies had a precision and recall of 0.96 and 0.99, and Petite colonies a precision and recall of 0.96 and 0.98 (Table 1). These results are without any post-processing or user input. To improve accuracy even further, users can cull model predictions below a particular quality score.

To test what these accuracy metrics mean for computations based on colony detections, we compared Petite frequencies determined by *petiteFinder* and manual counting for each petri dish (Fig. 4a). The red points are predictions, and the black points are manual counting (ground truth) Petite frequencies. Across all test images, the average absolute difference in predicted versus ground truth Petite percentages (prediction error) is 1.7%, with a standard deviation of 1.2%. To understand the potential impact of this error on biological insights gained from *petiteFinder* predictions, we compare the prediction error to expected variation in Petite frequencies from experimental sampling. Assuming that the generation of Petite colonies is a Bernoulli process, the standard deviation in Petite frequency expected from sampling numerous Petri dishes with the same underlying Petite probability is $\sqrt{\frac{P(1-P)}{k}}$. Here P represents the probability of Petite colony production and k is the number of colonies per Petri dish. Taking P to be the ground truth Petite frequency, and $k = 78$, which is the average number of colonies per plate in the test set, we plot this binomial sampling error as a gray envelope around the ground truth measurements with $\pm$ the binomial standard deviation (gray envelope in Fig. 4a). Above Petite frequencies of 0.1, Fig. 4a demonstrates that *petiteFinder* predictions are within this sampling error envelope, suggesting that prediction errors are less impactful than sampling error itself. Below

Table 1 Summary of ground truth versus predicted annotations from *petiteFinder*. Ground truth annotations of 1327 colonies were compared to predicted annotations from *petiteFinder* across 17 images in the test set

mAP(0.5:0.95): 0.64	**mAP@0.5: 0.96**	**mAP@0.75: 0.62**
Category	**Precision**	**Recall**
Grande	0.96	0.99
Petite	0.96	0.98

The top row includes mean average precision computations (mAP) across IOU thresholds from 0.5 to 0.95 in 0.05 increments, at 0.5 IOU, and 0.75 IOU. Below this, precision $\frac{TP}{TP+FP}$ and recall $\frac{TP}{TP+FN}$ at 0.5 IOU have been computed for each colony class

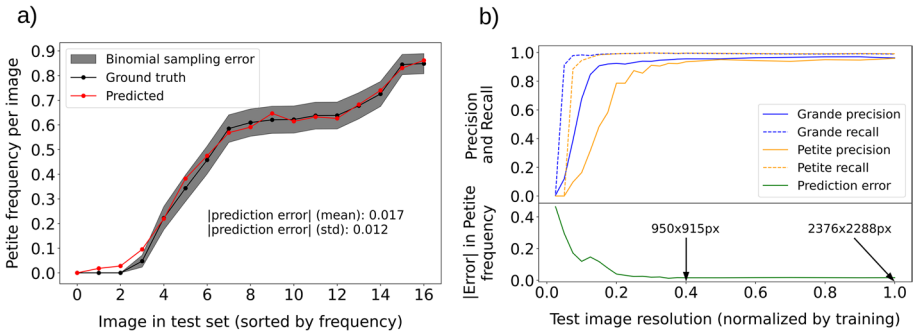

Fig. 4 *petiteFinder* performance compared to manual counting alongside a test of model robustness. **a** A plate-wise comparison of predicted and manually counted Petite colony frequencies. The red curves are the predictions and the black curves are manual counting. The average absolute deviation between predicted and ground truth Petite percentage (prediction error) is 1.7%. The gray envelope is the binomial sampling error (ground truth ± standard deviation), assuming that Petite production is a Bernoulli process with a probability equal to the ground truth frequency when sampling 78 colonies per plate image. **b** Top panel: Precision and recall of each colony category as a function of test image resolution normalized by the image resolution of the training data. Bottom panel: Absolute error in petite frequency as a function of test image resolution. The error is defined as the deviation between predicted and ground truth Petite frequency. Labels are also included to denote absolute image resolutions

0.1 petite frequency, where sampling error is negligible, predictions exist outside this envelope but still yield small absolute error compared to ground truth measurements ($< 5\%$ error in Petite percentage).

While we applied extensive data augmentation during training that would encompass a variety of imaging conditions, one parameter that remained fixed in both training and testing was the image resolution. Therefore, to understand how robust *petiteFinder* is to image resolution, we plotted model accuracy metrics while varying the image resolution of the test set in Fig. 4b. The x-axis in this plot should be interpreted as the fractional size in x and y dimensions of the test image relative to the training image resolution (2376×2288). Recall of the model in the top panel of Fig. 4b reveals that below 0.1 test image resolution (238×229) all colonies disappear according to the model due to their small size. As expected, this occurs first for Petites which are composed of fewer pixels. As we move towards lower resolutions starting at ~0.4 of test image resolution (950×915), we see that Petite colony precision drastically decreases due to misclassifications of small imaging artifacts as Petites. The same decrease in precision occurs for Grandes but at even lower test image resolutions. Petite frequency prediction error is also shown in the bottom panel as a function of test image resolution. Remarkably, at 0.4 test image resolution and above (950×915 or ~1M px^2), prediction accuracy is comparable to the results on our (2376×2288) resolution plate images. In general, the resolution threshold where Petite detection degrades most rapidly is dependent on experimental conditions that dictate Petite colony size. For our particular experimental conditions, this threshold indicates that a rapid degradation in Petite detection performance occurs below $4\times$ the resolution where Petite colonies vanish.

Comparing performance with semi-supervised methods

To provide context for the performance of *petiteFinder*, here we compare its performance on the test set to our own semi-supervised method tailored to Petite/Grande detection as well as the popular OpenCFU [13] and CellProfiler [14] pipelines.

As a stepping stone towards building *petiteFinder*, we developed a semi-supervised colony detection approach that incorporates Otsu's thresholding [29], watershed segmentation [30], and plate detection with a Hough transform [31]. We then relied on two features of Petite/Grande colonies that were biologically relevant to the classification problem, the size and intensity of colonies, and clustered colonies based on these features with average-linkage agglomerative clustering. The complete development of this method can be found in Additional File 1: Appendix A. Like most existing semi-supervised methods, the application of our semi-supervised pipeline required a choice of minimum/maximum colony sizes, which were selected to be 40 px^2/6000 px^2 by averaging the optimal choices for these parameters shown in Additional File 1: Table A.1, following a parameter grid search. The same minimum and maximum colony sizes were adopted for OpenCFU and CellProfiler to ensure a fair comparison. More details on the choices for other parameters in these pipelines can be found in Methods: Predictions with published semi-supervised methods.

First, we compare the colony segmentation performance of these semi-supervised methods to *petiteFinder*. The results are shown in Table 2, where precision and recall, which are class agnostic in the case of segmentation, are shown for IOU thresholds of 0.25 and 0.5. A small IOU threshold of 0.25 is important in this comparison because classical segmentation can produce bounding boxes smaller than the colonies themselves or the bounding boxes drawn by humans in the test set. OpenCFU and our semi-supervised approach exhibit comparable performance, with OpenCFU having a marginal advantage across all metrics and IOU thresholds. CellProfiler exhibits the lowest segmentation precision across semi-supervised approaches as a result of false positives due to refraction artifacts near the edges of Petri dishes (an example is shown in Additional file 1: Fig. B.2). The tradeoff, however, is that CellProfiler exhibits the highest segmentation recall across all semi-supervised methods. Examples of the failure modes of these semi-supervised methods can be found in Appendix B, Additional File 1: Figs B.1–B.5. In contrast to the semi-supervised methods, *petiteFinder* performs admirably, with better segmentation performance than the semi-supervised methods across all metrics. In

Table 2 A comparison of segmentation performance across existing semi-supervised colony detection methods, our own semi-supervised approach, and *petiteFinder*. Parameter regimes for each method are described in the text. Segmentation accuracy on the test dataset is displayed as class agnostic precision and recall at the two bounding box IOU thresholds of 0.25 and 0.5

Method (IOU)	Segmentation precision		Segmentation recall	
	(0.25)	(0.5)	(0.25)	(0.5)
OpenCFU	0.88	0.54	0.75	0.65
CellProfiler	0.56	0.32	0.84	0.75
Our semi-supervised	0.81	0.50	0.73	0.63
petiteFinder	**0.98**	**0.97**	**1.0**	**1.0**

Bold numbers indicate the maximum in each column

particular, *petiteFinder* improves segmentation precision at 0.25 IOU compared to the next best semi-supervised method by 11%, and recall by 19%.

Next, we compare colony detection performance, which is a combination of colony segmentation and classification. To do so with OpenCFU, which only provides segmentation, we applied the same unsupervised clustering of colony size/intensity as in our semi-supervised pipeline to the segmented regions. This reflects the canonical use of many of these segmentation pipelines, where users often apply their own classification methods to segmentation results. For CellProfiler, colonies were classified as Grande or Petite by applying a size threshold on the segmented mask. Results are shown in Table 3, where Grande and Petite precision and recall are shown for 0.25 and 0.5 IOU thresholds. Grande precision and recall are larger on average than for Petite colonies across all methods. This isn't too surprising, as small colonies have a greater potential to be filtered out from the background through size thresholding when comparable in size to artifacts or dust. Smaller colonies are also more likely to be eliminated through coarse graining during successive convolutions or when edge detection/texture filters are applied. OpenCFU coupled with our unsupervised clustering marginally outperforms *petiteFinder* in Grande precision, with an improvement of 2% in precision at 0.25 IOU. However, *petiteFinder* outperforms all other tested methods in Grande recall and Petite precision and recall. The Grande precision of *petiteFinder* is 27% better than the next best performing semi-supervised method at 0.25 IOU. More impressively, given the difficulty of detecting Petite colonies, *petiteFinder* improves Petite precision and recall by 59% and 18% over the next best performing semi-supervised pipeline at 0.25 IOU. Notably, this performance improvement is achieved without requiring any user input. This is in contrast to the finely tuned segmentation parameters that were adopted by both OpenCFU and CellProfiler from our semi-supervised colony detection approach.

Discussion

In this study we introduced *petiteFinder*, a computer vision tool to detect Grande and Petite colonies and determine Petite colony frequencies from images of Petri dishes. *petiteFinder* is the first computer vision pipeline tailored for this application to the

Table 3 A comparison of the object detection performance (segmentation + classification) across existing semi-supervised colony detection methods, our own semi-supervised approach, and *petiteFinder*. OpenCFU was modified by applying size/intensity average linkage clustering from our semi-supervised pipeline following its segmentation (referred to in table as + clust.). With CellProfiler, size filtering following segmentation was used to classify colonies as Grande or Petite. A version of CellProfiler with our unsupervised size/intensity clustering is also shown

Method (IOU)	Grande precision		Grande recall		Petite precision		Petite recall	
	(0.25)	(0.5)	(0.25)	(0.5)	(0.25)	(0.5)	(0.25)	(0.5)
OpenCFU + clust.	**1.00**	**0.97**	0.78	0.78	0.61	0.06	0.54	0.10
CellProfiler	0.97	0.95	0.76	0.75	0.40	0.07	0.83	0.47
CellProfiler + clust.	0.95	0.94	0.62	0.61	0.38	0.07	0.83	0.48
Our semi-supervised	0.99	0.93	0.60	0.58	0.46	0.06	0.52	0.13
petiteFinder	0.98	0.96	**0.99**	**0.99**	**0.97**	**0.96**	**0.98**	**0.98**

Parameter regimes for each method are described in the text. Grande and Petite precision and recall are shown with bounding box IOU thresholds of 0.25 and 0.5. Bold numbers indicate the maximum in each column

best of our knowledge. We showed that it performs comparably to human annotation, with errors that are smaller on average than the theoretical sampling error. Overall, the average difference in computed Petite percentages between *petiteFinder* predictions and manually annotated colonies was 1.7%. We also detailed best practices for using the tool, including minimum image resolutions and protocols to recreate imaging conditions. When *petiteFinder* is run on a GPU such as an Nvidia GTX 1070, each Petri dish image is processed in ∼2 s. Depending on colony density, this is a 10–100 fold improvement in throughput relative to manual counting without any user input. *petiteFinder* can also be run on a CPU, and, although the inference will be significantly slower (∼2 min per image), it still comes with the benefit of being completely automated. We believe that this advancement in throughput and detailed protocols for imaging and experimental conditions will drastically increase this assay's statistical power and repeatability.

While on average *petiteFinder* performed admirably, under 0.1 Petite frequency we see a larger prediction error than binomial sampling error from a distribution of cells with a fixed probability of becoming Petite. This is due to the sensitivity to false positives at low Petite frequencies. To address this issue and enable modifications to model predictions, we designed a GUI that allows experimentalists to visually explore and amend prediction outputs from *petiteFinder*. This 'amend' function lets users draw and remove bounding boxes around colonies following the initial prediction. Beyond performance and usability, we also emphasized how this computer vision problem of identifying Petite colonies highlighted ongoing difficulties in small object detection. We motivated changes to existing object detection architectures that were necessary to detect Petite colonies in this study, even with state-of-the-art architectures.

Regarding the development of this pipeline, a natural question is whether or not a supervised deep learning architecture is necessary to solve this problem with sufficient accuracy. Numerous popular semi-supervised approaches exist for colony segmentation, such as OpenCFU [13], CellProfiler [14], and AutoCellSeg [15]. These pipelines use a variety of thresholding and edge detection algorithms to segment colonies from the background of plate images, are immensely popular in the bioinformatics community, and perform comparably to humans in certain assays [13–15]. Therefore, is it necessary to leverage deep-learning to solve this problem? To address this question, we directly compared segmentation and classification performance between semi-supervised methods and *petiteFinder* on the test image set. We showed that *petiteFinder* outperforms all of the tested methods at colony segmentation. While *petiteFinder* was outperformed by 2% in Grande precision by a modified OpenCFU implementation, *petiteFinder* was an improvement of $\geq$ 18% across all other accuracy metrics compared to the next best semi-supervised methods. Most notably, *petiteFinder* improved Petite precision and Recall by 59% and 18%, which are the most important metrics when calculating Petite frequencies. These improvements place *petiteFinder* prediction error below theoretical sampling error in most regimes, which is not the case for semi-supervised methods applied to this problem (see Additional file 1: Figs. A.2b, A.5, B.4, B.5 for examples). Besides these performance improvements, *petiteFinder* also improves throughput as it requires no mandatory user input,

which is in stark contrast to the finely tuned semi-supervised methods that generally require user input per image to address varying experimental conditions. A general discussion of the user input requirements of these methods can be found in Additional File 1: Section B.1

Finally, it is important to comment on comparisons to existing deep-learning colony detection methods. A relevant comparison is the supervised red/white yeast colony detection pipeline described in [17]. This pipeline can be used for Petite frequency calculations in strains with an appropriate genetic background or with colonies that have been treated with tetrazolium. In yeast strains with specific mutations in the adenine synthesis pathway, an intermediate compound (initially white) accumulates and turns red when oxidized in respiring cells. It is also possible to treat cells in any genetic background with a tetrazolium-agar overlay to induce a transition from a white to red colony surface, specifically in Grande cells [10]. While the pipeline in [17] performs admirably, and could be modified for Petite detection with the red/white colony assay, there are two disadvantages to using the red/white assay over our experimental approach. First, the red/white assay is accompanied by a potentially cumbersome set of experimental constraints. The red/white assay requires constraints on the media (adenine limited media) and a specific genetic background of the strains, or chemical treatments that have to be carefully applied after colony growth. On the other hand, large/small colony detection only requires constraints on the media. Second, for red pigments to form, experimentalists often need to wait further after colony growth or chemical treatment which decreases experimental efficiency compared to our approach. With respect to the object detection pipeline in [17], one downside is that it requires post-processing of segmentation results, including heuristics on eccentricity and absolute size of colonies in pixels. Furthermore, assuming perfect segmentation in the study [17] (where segmentation accuracy is not reported), *petiteFinder* still achieves higher accuracy in Grande/Petite classification compared to white/red classification in experiments.

Conclusion

In this study we developed *petiteFinder*, an automated computer vision tool to detect Grande and Petite yeast colonies and determine Petite frequencies from images of Petri dishes. Colony detection with *petiteFinder* results in high accuracy Petite and Grande localization in images in a completely automated fashion. It achieves accuracy comparable to human annotation but at up to 100 times the speed and outperforms semi-supervised Grande/Petite colony classification approaches. By constructing this tool and providing details of experimental conditions, we hope this study will enable larger-scale experiments that rely on Petite colony frequencies to infer mitochondrial function in yeast.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-023-05168-5.

Additional file 1. Appendix A, B, and C for petiteFinder.

Acknowledgements
The authors thank numerous members of the Goyal lab for their thoughtful comments and discussions. They also thank members of the Humans Learning Machine Learning group (HLML) at the University of Toronto, in particular Jeremy Rothschild, for their comments on the manuscript. E.K. thanks Dr. Sarah Rauscher for funding support.

Author contributions
CJN and SG conceptualized the project. Both CJN and EK implemented, modified, and trained computer vision models in hyperparameter grid searches. CJN and EK also wrote the software for both the GUI and CLI. EK wrote tool installation and usage documentation. CJN performed the experiments and labeled data, evaluated model performance, created manuscript figures, and wrote the initial draft. EK edited the manuscript. CJN constructed and evaluated the semi-supervised approach in Additional File 1. S.G. acquired funding and provided feedback on the manuscript. All authors read and approved the final manuscript.

Funding
The authors (excluding E.K.) received funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant RGPIN-2015-0, the Simons Foundation Grant 326844 in the Mathematical Modeling of Living Systems, and funding for equipment from the Canadian Foundation for Innovation (CFI) Grant 32708.

Availability of data and materials
Source code, detailed documentation on installation and use, modifiable 3D print files, and all labeled data in this study are available at www.github.com/javathejhut/petiteFinder.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 9 September 2022 Accepted: 1 February 2023
Published online: 15 February 2023

References

1. Ephrussi B, Hottinguer H, Tavlitzki J. Action de l'acriflovine sur les levures: ii-etude genetique de mutant petite colonie. Ann Inst Pasteur. 1949;76:419–50.
2. Ephrussi B. Nucleo-cytoplasmic relations in micro-organisms–their bearing on cell heredity and differentiation. Oxford: Oxford at the Clarendon Press; 1953.
3. Contamine V, Picard M. Maintenance and integrity of the mitochondrial genome: a plethora of nuclear genes in the budding yeast. Microbiol Mol Biol Rev. 2000;64:281–315. https://doi.org/10.1128/mmbr.64.2.281-315.2000.
4. Dimitrov LN, Brem RB, Kruglyak L, Gottschling DE. Polymorphisms in multiple genes contribute to the spontaneous mitochondrial genome instability of saccharomyces cerevisiae s288c strains. Genetics. 2009;183:365–83. https://doi.org/10.1534/genetics.109.104497.
5. Ling F, Shibata T. Recombination-dependent mtdna partitioning: in vivo role of mhr1p to promote pairing of homologous dna. EMBO J. 2002;21:4730–40. https://doi.org/10.1093/emboj/cdf466.
6. Ling F, Shibata T. Mhr1p-dependent concatemeric mitochondrial dna formation for generating yeast mitochondrial homoplasmic cells. Mol Biol Cell. 2004. https://doi.org/10.1091/mbc.E03-07-0508.
7. Shibata T, Ling F. Dna recombination protein-dependent mechanism of homoplasmy and its proposed functions. Mitochondrion. 2007. https://doi.org/10.1016/j.mito.2006.11.024.
8. Ling F, Bradshaw E, Yoshida M. Prevention of mitochondrial genomic instability in yeast by the mitochondrial recombinase mhr1. Sci Rep. 2019. https://doi.org/10.1038/s41598-019-41699-9.
9. Karavaeva IE, Golyshev SA, Smirnova EA, Sokolov SS, Severin FF, Knorre DA. Mitochondrial depolarization in yeast zygotes inhibits clonal expansion of selfish mtdna. J Cell Sci. 2017;130:1274–84. https://doi.org/10.1242/jcs.197269.
10. Hess DC, Myers C, Huttenhower C, Hibbs MA, Hayes AP, Paw J, Clore JJ, Mendoza RM, Luis BS, Nislow C, Giaever G, Costanzo M, Troyanskaya OG, Caudy AA. Computationally driven, quantitative experiments discover genes required for mitochondrial biogenesis. PLoS Genet. 2009. https://doi.org/10.1371/journal.pgen.1000407.
11. Zamaroczy MD, Marotta R, Faugeron-fonty G, Goursot R, Mangin M, Baldacci G, Bernardi G. The origins of replication of the yeast mitochondrial genome and the phenomenon of suppressivity. Nature. 1981. https://doi.org/10.1038/292075a0.
12. Nunn CJ, Goyal S. Contingency and selection in mitochondrial genome dynamics. eLife. 2022;11:76557. https://doi.org/10.7554/eLife.76557.
13. Geissmann Q. Opencfu, a new free and open-source software to count cell colonies and other circular objects. PLoS ONE. 2013;8(2):1–10. https://doi.org/10.1371/journal.pone.0054072.
14. Bray MA, Vokes MS, Carpenter AE. Using cellprofiler for automatic identification and measurement of biological objects in images. Curr Protocols Mol Biol. 2015. https://doi.org/10.1002/0471142727.mb1417s109.

15. Khan AUM, Torelli A, Wolf I, Gretz N. Autocellseg: robust automatic colony forming unit (cfu)/cell analysis using adaptive image segmentation and easy-to-use post-editing techniques. Sci Rep. 2018. https://doi.org/10.1038/s41598-018-24916-9.
16. Siragusa M, Dall'Olio S, Fredericia PM, Jensen M, Groesser T. Cell colony counter called coconut. PLOS ONE. 2018;13(11):1–18. https://doi.org/10.1371/journal.pone.0205823.
17. Carl SH, Duempelmann L, Shimada Y, Bühler M. A fully automated deep learning pipeline for high-throughput colony segmentation and classification. Biol Open. 2020. https://doi.org/10.1242/bio.052936.
18. Dijkstra K, van de Loosdrecht J, Atsma WA, Schomaker LRB, Wiering MA. Centroidnetv2: a hybrid deep neural network for small-object segmentation and counting. Neurocomputing. 2021;423:490–505. https://doi.org/10.1016/j.neucom.2020.10.075.
19. Wang J, Chen K, Yang S, Loy CC, Lin D. Region proposal by guided anchoring. In Proceedings of the IEEE computer society conference on computer vision and pattern recognition. 2019. https://doi.org/10.1109/CVPR.2019.00308.
20. Akyon FC, Altinuc SO, Temizel A. Slicing aided hyper inference and fine-tuning for small object detection. 2022. arXiv:2202.06934 [Cs]. arXiv.org.
21. Ren S, He K, Girshick R, Sun J. Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017. https://doi.org/10.1109/TPAMI.2016.2577031.
22. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. 2016. https://doi.org/10.1109/CVPR.2016.90.
23. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S. Feature pyramid networks for object detection. In: Proceedings 30th IEEE conference on computer vision and pattern recognition, CVPR 2017. 2017. https://doi.org/10.1109/CVPR.2017.106.
24. Russell BC, Torralba A, Murphy KP, Freeman WT. Labelme: a database and web-based tool for image annotation. Int J Comput Vis. 2008. https://doi.org/10.1007/s11263-007-0090-8.
25. Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern. 1980;36(4):193–202. https://doi.org/10.1007/BF00344251.
26. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324. https://doi.org/10.1109/5.726791.
27. Xiao Y, Tian Z, Yu J, Zhang Y, Liu S, Du S, Lan X. A review of object detection based on deep learning. Multimed Tools Appl. 2020. https://doi.org/10.1007/s11042-020-08976-6.
28. Lin T-Y, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Dolláir, P. Microsoft coco: common objects in context. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition. 2015.
29. Otsu N. Threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern. 1979. https://doi.org/10.1109/tsmc.1979.4310076.
30. Digabel H, Lantuejoul C. Iterative algorithms. In: Proceedings of the 2nd European Symposium quantitative analysis of microstructures in material science, biology and medicine; 1978, 85–89.
31. Hough PVC. A method and means for recognition complex patterns; us patent: Us3069654a. US Patent. 1962.

Publisher's Note