BMC Bioinformatics

SOFTWARE

Open Access

CellSeg: a robust, pre-trained nucleus segmentation and pixel quantification software for highly multiplexed fluorescence images

Michael Y. Lee[1,2,3†], Jacob S. Bedia[4†], Salil S. Bhate[1,2,5], Graham L. Barlow[1,2], Darci Phillips[1,2,6], Wendy J. Fantl[4,7,8], Garry P. Nolan[2,7] and Christian M. Schürch[1,2,9*]

*Correspondence: christian.schuerch@med.uni-tuebingen.de
†Michael Y. Lee and Jacob S. Bedia are Co-first authors.
[1] Department of Microbiology and Immunology, Stanford University School of Medicine, Stanford, CA 94305, USA
Full list of author information is available at the end of the article

Abstract

Background: Algorithmic cellular segmentation is an essential step for the quantitative analysis of highly multiplexed tissue images. Current segmentation pipelines often require manual dataset annotation and additional training, significant parameter tuning, or a sophisticated understanding of programming to adapt the software to the researcher's need. Here, we present CellSeg, an open-source, pre-trained nucleus segmentation and signal quantification software based on the Mask region-convolutional neural network (R-CNN) architecture. CellSeg is accessible to users with a wide range of programming skills.

Results: CellSeg performs at the level of top segmentation algorithms in the 2018 Kaggle Data Challenge both qualitatively and quantitatively and generalizes well to a diverse set of multiplexed imaged cancer tissues compared to established state-of-the-art segmentation algorithms. Automated segmentation post-processing steps in the CellSeg pipeline improve the resolution of immune cell populations for downstream single-cell analysis. Finally, an application of CellSeg to a highly multiplexed colorectal cancer dataset acquired on the CO-Detection by indEXing (CODEX) platform demonstrates that CellSeg can be integrated into a multiplexed tissue imaging pipeline and lead to accurate identification of validated cell populations.

Conclusion: CellSeg is a robust cell segmentation software for analyzing highly multiplexed tissue images, accessible to biology researchers of any programming skill level.

Keywords: Deep learning, Segmentation, Image analysis, CODEX, Mask R-CNN, Multiplexed imaging, Pre-trained model

Introduction

Tissue imaging and single-cell analysis can reveal previously undetected biological structure and uncover subtle spatial relationships between cells. Recently, the development of antibody-based multiplexed imaging methods has enabled deep single-cell phenotyping of tissue microenvironments [1–9]. This analysis has been especially useful in cancer studies, where these imaging platforms have revealed nuanced tumor architecture and

interactions between tumor, immune and stromal cells, and healthy host tissue [10–16]. In such highly multiplexed tissue imaging studies, the quality and accuracy of downstream analyses depend critically on the precise identification and correct phenotypic assignment of single cells, which requires accurate demarcation of each cell's boundary and quantification of its marker expression. This is usually accomplished using an automated segmentation and signal quantification algorithm [17]. At a minimum, a segmentation algorithm takes an image as input and produces a set of masks denoting the boundary of each identified cell.

Commonly used segmentation algorithms include watershed (WTS) combined with thresholding [18, 19] and level-set techniques [20]. For example, WTS segmentation was recently used to identify cells in highly multiplexed fluorescence microscopy datasets of mouse and human tissues [3, 15, 16, 21]. However, these methods can be sensitive to noise within the image including blurred cell–cell contact boundaries and imaging artifacts such as antibody aggregates. Additionally, they are often not robust to variations in cell size or morphology and require significant parameter tuning for expected cell size, shape of nucleus, and cell density [22]. These limitations make their application to segmenting images of tumor tissue challenging, since cancers consist of a variety of cell shapes, sizes, and densities. Advances in deep learning architectures have transformed cell image analysis [23], and these models have been extended to applications in single-cell segmentation [24–31]. Leading among these algorithms is the Mask R-CNN architecture, which has previously shown positive performance on other segmentation tasks [26, 32]. However, deep learning algorithms usually require pre-labeled training data for the specific segmentation task, leading to substantial and time-consuming human input to obtain a high-quality segmentation.

While pre-trained deep learning architectures exist, such as StarDist [33] and Cellpose [26], an additional issue is the subsequent handling of the segmentation output to produce single-cell statistics used for downstream analysis, including pixel quantification. This results in either extra coding or exporting of masks to another image processing software where additional commands allow quantification of pixels in the segmented image to produce single-cell statistics. Further processing steps, including expanding mask boundaries and reducing noise in the statistics must be completed separately. Complete segmentation pipelines including CellProfiler [34] or ilastik [35] address these concerns but still often require hands-on user input, such as manual annotation or processing of segmentation results.

Here, we present CellSeg (https://michaellee1.github.io/CellSegSite/index.html), an easy-to-use, pre-trained, Mask R-CNN-based cell segmentation and pixel quantification software. Users supply a set of tissue images to CellSeg, and the software returns the segmented images and a table of single-cell statistics including each cell's location, nucleus size, and mean pixel values in each imaging channel. CellSeg is open-source and available for Windows, Mac, or Linux. It is capable of segmenting JPG, PNG, and TIFF images of any image size, subject to hardware requirements reported below. CellSeg is accessible to individuals of all programming levels and requires minimal user input. For most uses, the pre-trained CellSeg model requires no additional manual annotation of training data, no additional training, and limited parameter tuning to produce a high-quality segmentation. The software has been designed to work as a library, so more advanced Python

users can customize the pipeline to fit their needs. Applications detailed below demonstrate that CellSeg robustly segments highly multiplexed fluorescence images from a variety of healthy and cancerous tissues. CellSeg exceeds an established WTS segmentation pipeline both in terms of user friendliness as well as segmentation accuracy and performs at the level of two state-of-the-art deep learning segmentation algorithms.

Implementation

Overview of CellSeg pipeline

The CellSeg software is implemented in Python and run using Jupyter Notebook [36]. CellSeg first extracts a user-specified nucleus color channel for segmentation (Fig. 1, step 1). Through iterative visual inspection, we found increasing the brightness of the

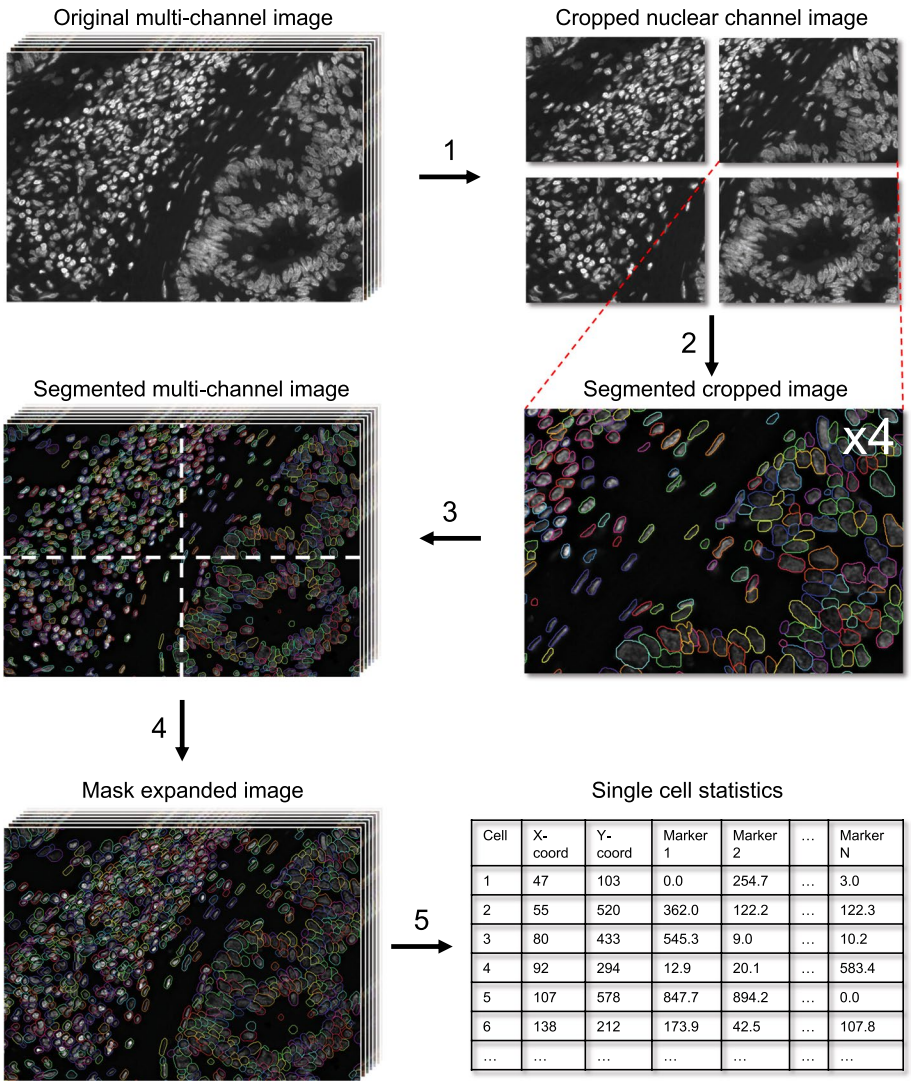


Fig. 1 CellSeg pipeline. Overview of CellSeg software with following steps. (1) Extract nuclear channel and crop images to segment. (2) Segment each image crop with CellSeg. (3) Stitch together segmented crops. (4) Expand boundaries of cells using mask expansion. (5) Perform lateral bleed compensation, then compute and output single-cell statistics for N markers

nuclear channel can improve segmentation performance, especially in images with weak nuclear stain signal. CellSeg therefore scales each nuclear image's brightness by a fixed constant computed from a reference image specified by the user. After scaling, CellSeg splits the nuclear stain image into several overlapping cropped images to accelerate segmentation (Fig. 1, step 1). Each image crop is segmented, and the resulting segmented crops are stitched into the full segmented multi-channel image (Fig. 1, steps 2–3). To eliminate erroneously segmented imaging artifacts, which often appear as high intensity speckles or clusters, CellSeg removes objects smaller than a user-specified threshold from the set of segmented cells.

After segmentation, two optional post-processing steps follow: mask expansion and lateral bleed compensation (Fig. 1, steps 4–5). Mask expansion extends the boundary of each segmented nucleus by a user-specified number of pixels to capture cell membrane fluorescent signal (Fig. 1, step 4). Lateral bleed compensation aims to correct fluorescent signal spillover between adjacent cells, an issue often seen in immunofluorescence imaging of dense tissues. The details of these steps and their performance are discussed below. After post-processing, CellSeg computes the mean pixel value for each marker over the set of pixels contained in each identified cell. In fluorescent images, these values can be seen as a proxy to the expression level of each imaged protein in each cell. CellSeg saves each cell's (X, Y) coordinate and pixel quantifications to a table in both comma-separated value (CSV) and flow-cytometry standard (FCS) formats (Fig. 1, step 5). This data output format is recognizable by popular downstream single-cell analysis software, including flow cytometry gating programs such as CellEngine (https://cellengine.com), Cytobank (https://www.cytobank.org), or FlowJo (https://www.flowjo.com), as well as cell clustering programs like VorteX [37]. For visual inspection of segmentation quality, the user can also optionally generate images of the segmented tissue with overlaid masks and a TIFF stack of mask regions of interest (ROIs) which can be viewed in other image analysis programs like Fiji/ImageJ [38].

User accessibility

CellSeg is an open-source segmentation software accessible to individuals with a range of programming skills. On the CellSeg webpage (https://michaellee1.github.io/CellSegSite/index.html), we created detailed tutorials for software installation, setup, and segmentation configuration. These tutorials assume no prior programming knowledge, and they walk the user through all phases of the CellSeg pipeline. The user has the option to run CellSeg using either a Jupyter Notebook containing a step-by-step walkthrough of the pipeline with comments or a fully automated script for segmenting several images. For more advanced users, the components of the pipeline can be used as a library. This allows users to incorporate algorithms from CellSeg into their personal segmentation pipelines or use CellSeg algorithms individually during exploratory image analysis. CellSeg can be run in the background on any computer with sufficient storage for the input image files with at least 16 GB RAM. While it does not require a GPU, CellSeg can be accelerated for those with access to hardware using the open-source package tensorflow-gpu [39], and instructions for GPU acceleration are provided on the webpage.

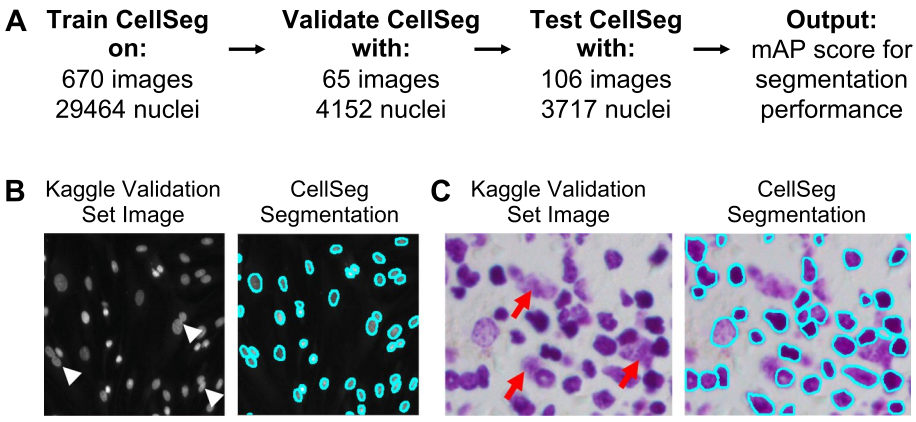
Training

CellSeg was trained on a dataset of fluorescent and brightfield biological microscopy images from the 2018 Kaggle Data Science Bowl containing 29,464 ground truth segmented nuclei (Fig. 2A) [40]. These images were acquired with variations in cell phenotype, size, image zoom, and brightness. Of note, CellSeg was not trained on any highly multiplexed tissue imaging data. Both the CellSeg architecture and training method were optimized for robustness to variations in cell size and morphology. Further details of CellSeg's architecture and training can be found in the "Methods" section.

Results and discussion

CellSeg architecture achieves high performance on Kaggle data challenge test set

After implementing CellSeg, we validated each step of the pipeline: architecture, segmentation, segmentation post-processing, and output. First, we quantitatively evaluated



D

Segmentation	mAP score	Average F1
nucleAIzer	0.6330	N/A
CellSeg	0.6320	0.7121
Competition 1st Place	0.6316	0.7120
CellProfiler	0.5281	0.6280
StarDist	0.4909	N/A
Cellpose	0.4723	N/A

Fig. 2 Training and Benchmarking CellSeg performance on the 2018 Kaggle data challenge. **A** Information on Kaggle dataset used to develop, train, and test CellSeg. CellSeg final performance was assessed on a test set provided by the Kaggle data challenge using mean average precision (mAP) score. **B** CellSeg segmentation of representative fluorescence image from the Kaggle test set. White arrowheads: cells with blurred nuclear boundaries **C** CellSeg segmentation of representative H&E-stained brightfield image. Red arrows: nuclear debris. **D** CellSeg performance compared to other top performing segmentation algorithms in data science bowl. Columns show mean average precision (mean AP) scores reported on Kaggle DSB2018 stage 2 test set and average F1 scores. For nucleAIzer, reported scores from the original publication [29] are displayed. For StarDist, brightfield and fluorescence images were segmented using 2D_versatile_he pre-trained model and 2D_versatile_fluo pre-trained model, respectively. For Cellpose, the pre-trained nuclei segmentation model was used (see "Methods" section for testing details)

our trained Mask R-CNN segmentation architecture using the 2018 Kaggle Data Science Bowl [40]. This challenge dataset enabled assessment of the performance of our trained Mask R-CNN architecture in comparison with top performers in this competition and more recently published neural network-based cell segmentation algorithms such as nucleAIzer [29], StarDist [33], and Cellpose [26].

CellSeg was tested for segmentation quality on a ground truth segmented validation set of 3717 nuclei from Kaggle (Fig. 2A). We found that CellSeg qualitatively performed well on fluorescence images where it accurately identified individual nuclei even when boundaries between two nuclei were blurred (Fig. 2B). CellSeg accurately identified many nuclei in the brightfield image stained with hematoxylin & eosin (H&E) (Fig. 2C), but overall performed better in fluorescent images, due potentially to the higher amounts of nuclear debris and other artifacts in the H&E-stained images. The Kaggle challenge used mean average precision (mAP) as the segmentation quality metric ("Methods" section). A higher mAP score corresponds to a more accurate segmentation on the Kaggle competition's test set. By the mAP metric, CellSeg performed among the top algorithms in the competition and comparably to nucleAIzer, while attaining a higher mAP score than both StarDist and Cellpose (Fig. 2D).

CellSeg outperforms an established WTS segmentation algorithm on a multi-tissue microarray

Next, we evaluated CellSeg's performance on a set of tissues representing a variety of cell and nuclear sizes, densities, and morphologies. Using a tissue microarray (TMA) comprised of human tumor and healthy tissues and imaged with the fluorescent nuclear marker DRAQ5 [15], we qualitatively assessed the performance of CellSeg against an established WTS algorithm that was tuned using hyperparameters selected by an expert pathologist for optimal segmentation quality [3]. We found that CellSeg was less sensitive to variations in nucleus size and morphology, outperforming the WTS algorithm on most tissues (Fig. 3). Both CellSeg and WTS correctly segmented immune cells and cells with spindly nuclei (Fig. 3A, B). While WTS tended to incorrectly segment large nuclei into several smaller masks in glioblastoma multiforme (GBM), hepatocellular carcinoma (HCC), and seminoma tissues, CellSeg correctly identified both large and small nuclei in the same images (Fig. 3C–E). CellSeg robustly identified nuclei with significant variations in brightness, as observed in the GBM and HCC tissues. CellSeg and WTS with expert annotation did not perform well on an image of T-cell acute lymphoblastic leukemia (T-ALL), a highly dense tumor composed of small lymphocytes with obscured nuclear boundaries (Fig. 3F). Manually segmenting such tissues is challenging even for expert pathologists. These findings underscore the importance of both tissue quality and clear separation between individual cells as key parameters for optimal segmentation.

(See figure on next page.)

Fig. 3 CellSeg performance on diverse human FFPE tissues. CellSeg performance on representative tissue images from a multi-tumor tissue microarray imaged with CODEX, all stains are DRAQ5 nuclear stain. **A.** Healthy spleen shows small cells. **B.** Dermatofibrosarcoma protuberans (DFSP) shows spindly nuclei. **C.** Glioblastoma multiforme (GBM) shows large, misshapen cells. **D.** Hepatocellular carcinoma (HCC) shows large, round cells. **E.** Seminoma shows a blend of large tumor cell nuclei and small nuclei from tumor-infiltrating lymphocytes. **F.** T-cell acute lymphoblastic leukemia (T-ALL) shows densely packed cells. Scale bar, 20 μm. Fluorescence intensity increased in original images for visualization purposes

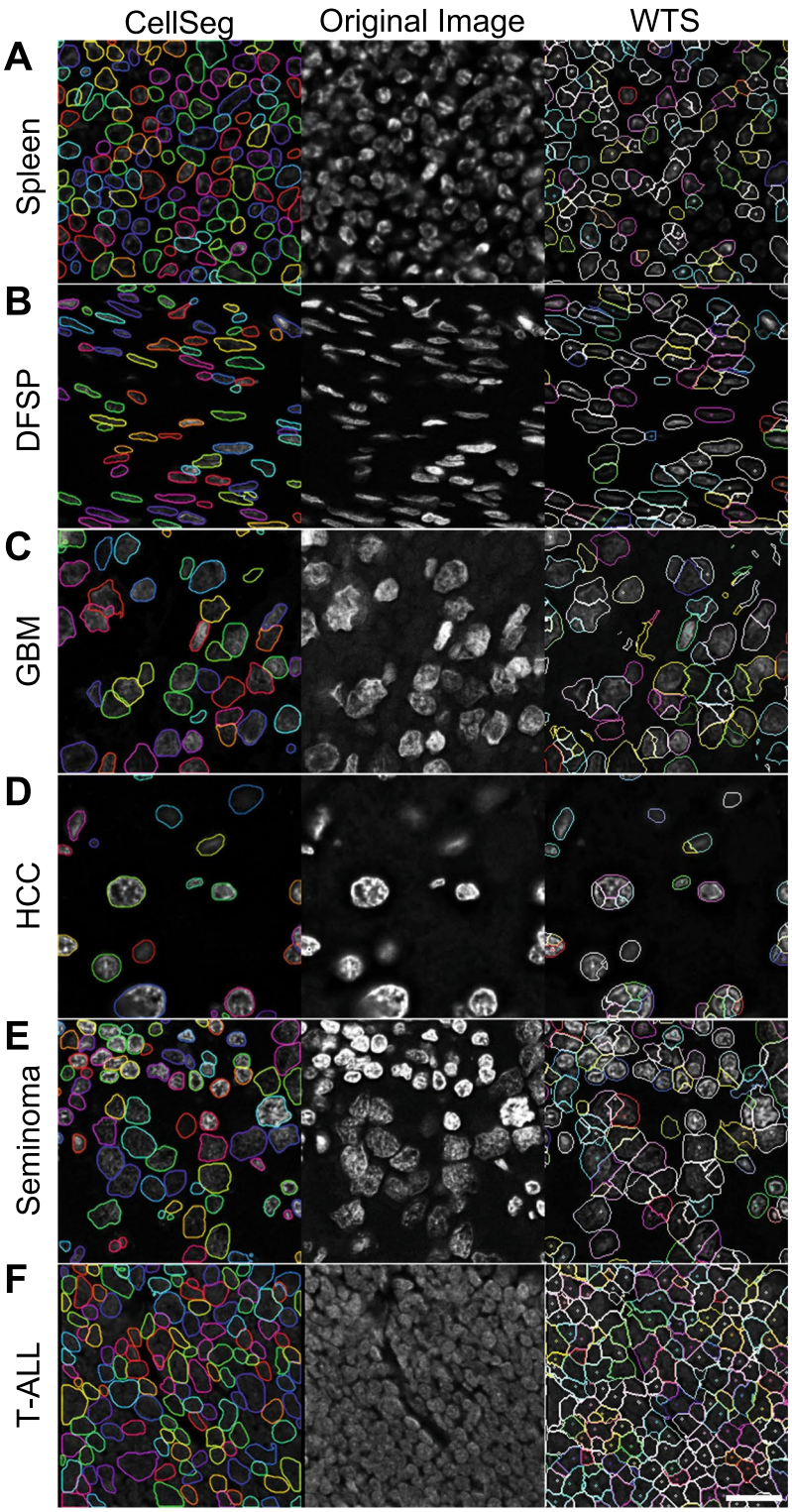


Fig. 3 (See legend on previous page.)

Overall, CellSeg performed at least as well as the established WTS algorithm on all tissues, while clearly outperforming it on three tissues (GBM, HCC, and seminoma).

CellSeg performs at the level of state-of-the-art neural network-based segmentation algorithms

Next, we qualitatively compared CellSeg's performance to two recently published neural network-based segmentation algorithms, StarDist [33] and Cellpose [26]. The six tissues from our TMA shown in Fig. 3 were segmented using pre-trained models of StarDist (2D versatile fluo) and Cellpose, and the segmentation masks were compared to masks generated by CellSeg. All three algorithms performed comparably well on spleen and HCC (Fig. 4A, D). Compared to CellSeg, StarDist segmented more objects with low fluorescence intensity in GBM and seminoma (Fig. 4C, E). However, StarDist also oversegmented many nuclei in DFSP, while CellSeg identified them accurately, suggesting that CellSeg is more robust to variations in nuclear morphology (Fig. 4B). Both StarDist and CellSeg identified more nuclei than Cellpose in DFSP, GBM, and seminoma (Fig. 4B, C, E). All three algorithms performed poorly on T-ALL (Fig. 4F). In summary, CellSeg performs at the level of state-of-the-art neural network-based segmentation algorithms when applied to a real-world dataset.

CellSeg post-processing steps improve downstream resolution of immune populations

In the development of CellSeg, we addressed two post-segmentation issues. First, because CellSeg is a nucleus segmentation algorithm, the boundary identifying a cell's nucleus often fails to capture the fluorescent signal of its plasma membrane where many of the protein markers used for cellular identification are located (e.g., CD45 denoting an immune cell, and EpCAM denoting an epithelial cell). Second, fluorescent imaging often results in spatial fluorescent spillover between adjacent cells, creating noise in quantification of protein expression [3]. To resolve these issues, two optional steps follow segmentation with CellSeg: (1) mask expansion (Fig. 5A, step 1) and (2) lateral bleed compensation (Fig. 5A, steps 2–3). Mask expansion extends the boundary surrounding each segmented nucleus by a user-defined number of pixels. This allows for quantification of the plasma membrane fluorescent signal. Next, lateral bleed compensation corrects for fluorescence spillover between adjacent cells. As described in Goltsev et al. [3], this algorithm computes the surface contact ratios between physically adjacent cells and uses this value to simultaneously boost signal from a cell and reduce spatial spillover noise from neighboring cells.

We tested the efficacy of the lateral bleed compensation algorithm with CellSeg on an immunofluorescence dataset imaged on the CO-Detection by indEXing (CODEX) platform. CODEX iteratively visualizes protein-antibody binding events, allowing for the quantification of more than 50 protein targets in formalin-fixed, paraffin-embedded (FFPE) or fresh-frozen tissue sections [3, 15]. Using CODEX, we recently imaged two TMAs containing 140 samples from 35 patients with colorectal cancer (CRC). In this study, WTS was used to segment the images [15]. This particular WTS segmentation used the same bleed compensation algorithm that we implemented for CellSeg. Single-cell marker quantifications from the WTS segmentation were extensively

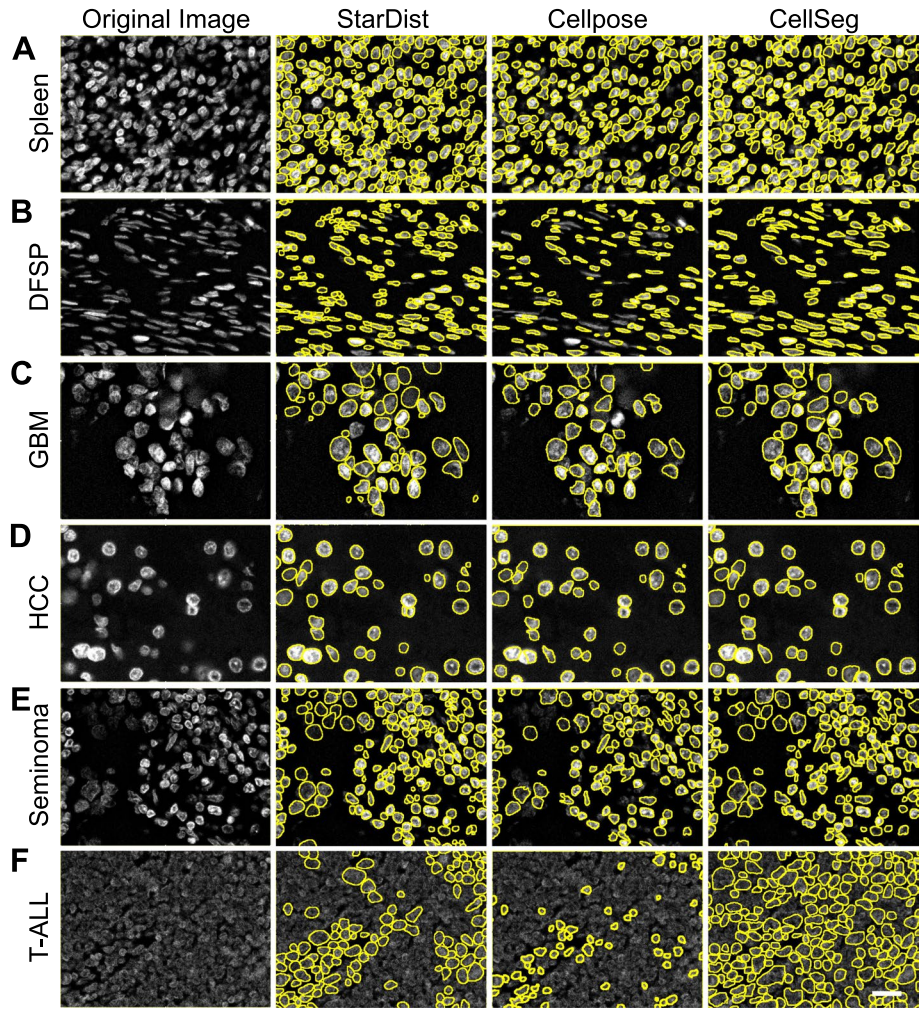


Fig. 4 CellSeg performs comparably to established deep learning-based segmentation algorithms on diverse human FFPE tissues**.** Representative images from tissues described in Fig. 3 are shown. **A.** StarDist, Cellpose, and CellSeg show comparable performance on spleen. **B.** StarDist oversegments several spindly nuclei in DFSP (arrows), while CellSeg and Cellpose segment nuclei accurately. **C.** StarDist and CellSeg segment more low intensity objects in GBM (arrows). **D.** all three algorithms perform similarly well on HCC. **E.** StarDist and CellSeg segment more low intensity objects in seminoma (arrows). **F.** All three algorithms perform relatively poorly on T-ALL. Scale bar, 20 µm

validated in this study, providing us with a baseline for the expected expression profiles of cell types against which we could validate our bleed compensation algorithm.

Using CellSeg, we segmented the 140 CRC images either with or without bleed compensation and gated the segmented data in CellEngine. In this dataset, the expression of certain pairs of imaged protein markers are expected to be mutually exclusive based on their known biology, including CD20/CD3, CD8/CD4, and cytokeratin/CD45. However, the presence of several densely populated immune cell regions, as observed in the CRC dataset, can lead to the erroneous identification of cells that are positive for both markers due to spatial fluorescent spillover. Applying an approach previously used to assess compensation [3], we measured the efficacy of

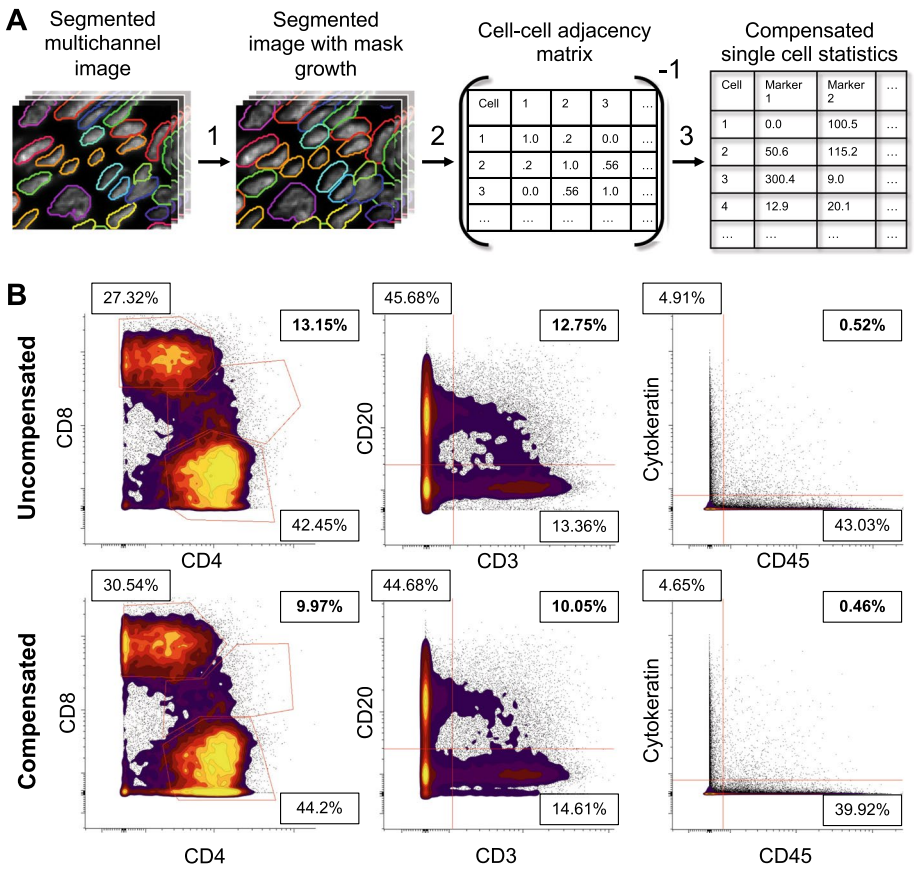


Fig. 5 Testing lateral bleed compensation on a CODEX dataset of colorectal cancer samples. **A.** Schematic demonstrating post-processing of CellSeg segmentation with following steps. (1) Grow cell boundaries by user defined number of pixels (growth of two pixels shown). (2) Compute inverse adjacency matrix from cell–cell. (3) Multiply inverted adjacency matrix by marker pixel intensity vector to obtain compensated single-cell pixel quantifications table. **B.** Effects of lateral bleed compensation on double-positive cell populations in the CRC dataset for three pairs of mutually exclusive markers (CD8 vs. CD4, Cytokeratin vs. CD45, CD20 vs. CD3). Data shown are from one of the two CRC TMAs (TMA A), with comparable bleed compensation performance for the other TMA (TMA B, data not shown)

bleed compensation by the observed reduction in the frequency of cells double positive for any of these pairs of markers (Fig. 5B). Lateral bleed compensation reduced the frequency of double positive cells in both the CD8/CD4 and CD20/CD3 pairs, with a lesser reduction of double positive cells in the cytokeratin/CD45 pair. Therefore, the lateral bleed compensation implemented in CellSeg improves the resolution of immune cell expression profiles in a dataset with densely packed immune cell regions.

Phenotyping using CellSeg output recapitulates validated cell populations in the CRC CODEX Dataset

In previously published work using the CODEX pipeline, WTS segmentation provided single-cell fluorescent intensity statistics that were used to assign cell phenotypes in the CRC dataset [15]. To assess whether CellSeg could replace WTS in the

CODEX pipeline, we performed a head-to-head comparison by gating segmented cells in all 140 samples with key phenotyping markers of major cell types as validated by an expert pathologist.

To confirm that the gated cell types derived from WTS and CellSeg occupied similar regions within the CRC tumors, we directly visualized these cell types on the CRC images. This analysis showed that CellSeg and WTS generated the same cell phenotypes with comparable spatial organization within the CRC tumors (Fig. 6A). Further validation was performed by examining the original fluorescent image displaying key phenotyping markers (Fig. 6B). Both CellSeg and WTS cell types, identified by gating, had the expected cell morphology and fluorescence marker profile when inspecting the corresponding regions of the fluorescent image.

For a more global quality assessment of our analysis, we correlated the cell types quantified by CellSeg with those identified by WTS. For each sample, we correlated the absolute number of cells in each gated population using the outputs of CellSeg and WTS (Fig. 6C). As examples, we depict gated T cells, macrophages, B cells, and tumor cells, all of which showed very strong positive correlations between the WTS and CellSeg segmented cell counts. The cell types generated from the CellSeg segmentation matched previously validated cell types generated from WTS in the CRC dataset. However, while cell counts from CellSeg and WTS were correlated, WTS segmentation generally resulted in higher numbers of tumor cells (Fig. 6C). This is likely due to over-segmentation of large tumor cell nuclei, as observed in GBM, HCC, and seminoma (Fig. 3C–E), suggesting that CellSeg is superior to WTS for tumor cell identification. These findings demonstrate that tissue analysis using CellSeg can recapitulate previously published findings in a multiplexed fluorescence imaging study.

Conclusions

In summary, we present CellSeg, a robust single-cell segmentation and quantification software for tissue images. Our software has been designed to be accessible to researchers of all programming skill levels. For novice programmers, we have created detailed tutorials on how to implement and use CellSeg (https://michaellee1.github.io/CellSegSite/index.html). For more advanced Python users, the components of the CellSeg pipeline function as a library to complete customized image analysis or segmentation tasks. As more sophisticated segmentation algorithms emerge, future researchers can use the CellSeg pipeline and combine it with their algorithm of choice. Importantly, our pre-trained segmentation algorithm works "out-of-the-box" for many single-cell segmentation tasks, without requiring any additional manual training by the user.

We validated each step of the CellSeg pipeline: architecture, segmentation, post-processing, and output. In a post-competition evaluation, the pre-trained CellSeg architecture scored among the top performers from the 2018 Kaggle data challenge. CellSeg also qualitatively outperformed an established segmentation algorithm on a multi-tissue TMA. Both qualitative and quantitative comparisons demonstrated that CellSeg performs comparably to state-of-the-art deep learning-based segmentation algorithms. Finally, using a CODEX CRC dataset, we showed that the post-processing steps in our CellSeg pipeline improved resolution of mutually exclusive cell populations while recapitulating previously published cell populations in the dataset. CellSeg is therefore a

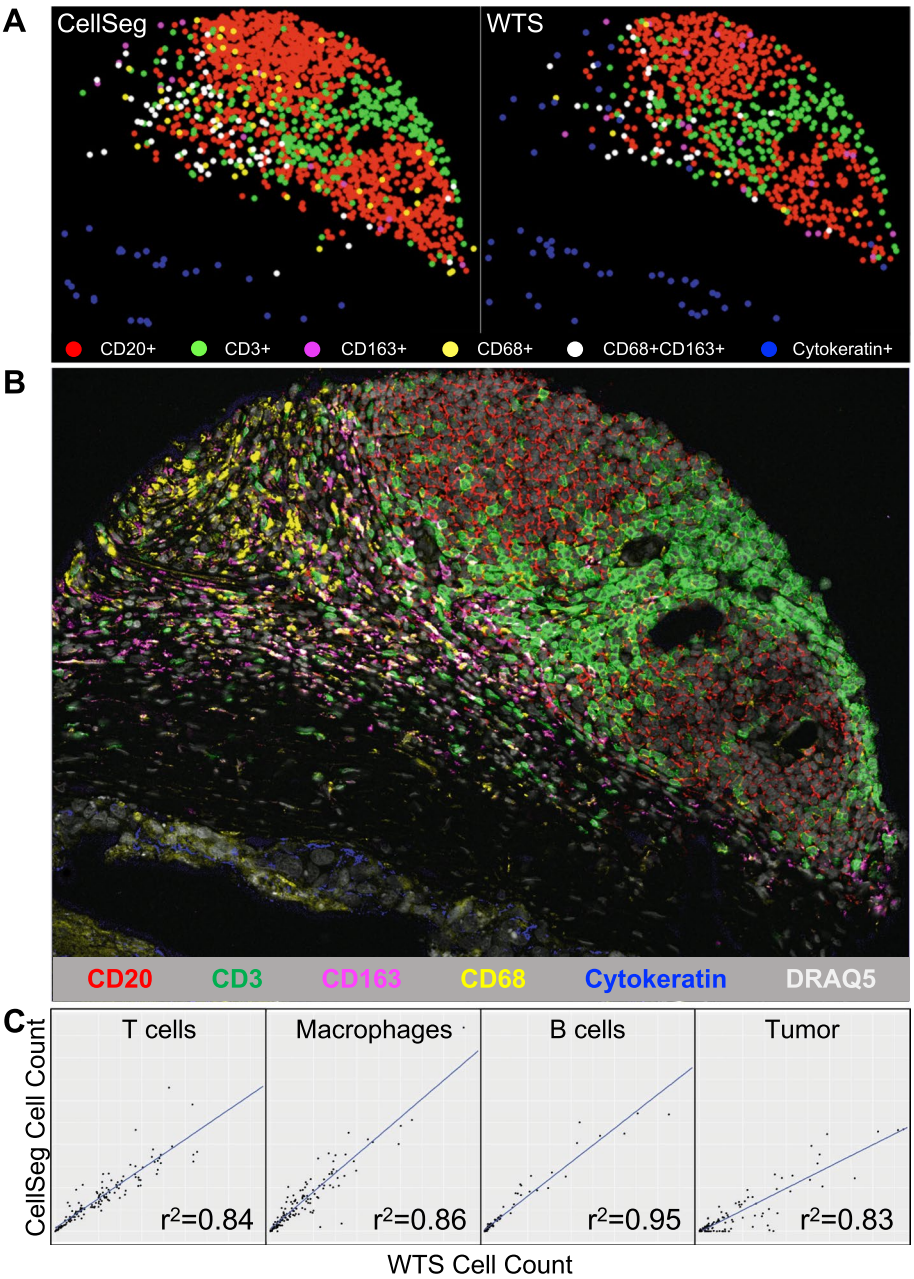


Fig. 6 Recapitulating previously identified cell populations using CellSeg. **A.** Visualization of identified populations in a representative CRC tissue. Points on scatter plots show positions of cells on the displayed tissue image in Fig. 5B. Population identity obtained through gating. **B.** Fluorescent image of a representative CRC tissue. Expression of six phenotyping markers used in Fig. 5A shown. **C.** Population correlation analysis between CellSeg and WTS. Each point corresponds to a TMA spot, where the X value is the gated population count from WTS and the Y value is the count computed from CellSeg. Least-squares regression line displayed along with r^2 value. T cells are defined as $CD45^+CD3^+CD20^-Cytokeratin^-$; macrophages as $CD45^+CD20^-CD3^-$ and $CD68^+$, $CD163^+$, or $CD68^+CD163^+$; B cells as $CD45^+CD20^+CD3^-Cytokeratin^-$; and tumor as $Cytokeratin^+CD45^-$

powerful tool that has shown robust performance on a wide variety of tissue segmentation tasks and should help researchers with their needs in cell segmentation and marker quantification in biological imaging data.

Methods

Architecture

CellSeg is based on the Matterport implementation of Mask R-CNN [32, 41]. CellSeg uses a slightly modified loss function during training. The original Mask R-CNN paper uses $L = L_{cls} + L_{box} + L_{mask}$, where the loss is the additive sum of class loss, bounding box loss, and mask loss as defined in the paper. A new hyperparameter was added to arrive at $L = \alpha L_{cls} + L_{box} + L_{mask}$. Through experimental analysis, it was found that reducing the contribution of class loss in our single-class model improved convergence during training. For the results in this paper, $\alpha = 0.5$.

Training

To prevent overfitting and to extend the training data, multiple image augmentation techniques were used, which contributed significantly to CellSeg's quantitative performance. The first was simple random field sampling. At train time, 512×512 pixel crops of the image were used, selected randomly from the image. Using the imgaug library version 0.2.9, contrast normalization, brightness, Gaussian blur, zooms scaling the X and Y axes independently, vertical and horizontal flips, and rotations were also modulated throughout training [43]. No augmentations were conducted at test time. The model was trained in minibatch sizes of 16 using stochastic gradient descent with momentum. Transfer learning was utilized for the dataset, with weights used from a Mask R-CNN model that trained on COCO, a segmentation challenge with 91 object types and 2.5 million labeled instances [44]. Auxiliary functions for training were adapted from the DeepRetina DSB2018 training scripts, although our model did not use their trained weights [42]. The network was trained for 150 epochs with decaying learning rate with the base model frozen, to let only the top layers train. Then, the full network was trained for 25 epochs at a very low learning rate. All training was done on an Nvidia GTX 1080 Ti GPU and a Dual Intel® Xeon® Silver 4114 10-core CPU, taking about 41 h to train.

Mean average precision metric

For each segmented image, mean average precision was computed using intersection over union (IoU) between segmented masks and ground truth masks as follows. IoU compares the overlap between the CellSeg segmentation of a cell and a ground truth manual segmentation. First, for each segmented mask, the pixel IoU, defined as the ratio of the overlap between ground truth mask A and CellSeg mask B to the total area that A and B cover was computed as $IoU(A, B) = (A \cap B)/(A \cup B)$. IoU values range from 0 to 1, with 1 denoting a perfectly segmented cell, i.e., the ground truth. The number of cells with IoU values exceeding threshold t were computed, where t ranges from 0.5 to 0.95 in increments of 0.05. At each t, a precision value Q(t) was calculated as: $Q(t) = TP(t)/(TP(t) + FP(t) + FN(t))$ where TP, FP, and FN were the number of true positives, false positives, and false negatives identified in an image, respectively. A TP is defined as an object with a pixel IoU above the threshold t. The Average Precision (AP) of an image was then computed as the mean over the thresholds:

$AP = (1/|\text{thresholds}|)\sum_t Q(t)$. Finally, the mean AP (mAP) was computed as the mean over the AP of each image in the test dataset.

Quantitative segmentation evaluation

Mask predictions were made on Kaggle DSB 2018 stage 2 test set images for CellSeg and uploaded to the DSB 2018 page on the Kaggle website (after the competition closed) to obtain mAP score. For StarDist (version 0.7.1), fluorescence and brightfield images were segmented using 2D_versatile fluo and 2D_versatile_HE pre-trained models, respectively. For Cellpose (version 0.6.5), fluorescence images were segmented using channels = [0,0] with all other parameters set to default. Brightfield images were segmented using parameters channels = [1,0], invert = True, and flow_threshold = 0.8 with all other parameters set to default. Predicted masks for each segmentation algorithm were saved and uploaded to Kaggle to obtain the mAP score.

TMA qualitative segmentation evaluation

CellSeg and an optimized WTS segmentation algorithm [3] were both used to segment six tissue images from a multi-tissue TMA. Representative images from the results of each segmentation were selected for Fig. 3. We used the best-focus image returned by the 3D WTS algorithm to compare segmentation results. Size parametrization for WTS was hand-verified by a board-certified surgical pathologist (C.M.S.). For qualitative comparison between CellSeg, StarDist, and CellPose in Fig. 4, 300×400 pixel patches were sampled randomly from each tissue image to visualize. StarDist (version 0.7.1) 2D_versatile_fluo pre-trained model was used for segmentation with default parameters. For Cellpose, the pre-trained model (version 0.6.5) with no modifications was used. Segmentation results were visualized with mask ROI overlays in ImageJ.

Mask expansion algorithms and lateral bleed compensation

We implemented two mask expansion algorithms. The first algorithm expands the boundary of each mask by a user-defined number of pixels. If this expansion leads to two overlapping masks, the algorithm assigns each pixel in the overlapping region to the mask whose center is closest to the pixel. The first mask expansion algorithm is computationally efficient and works well for tissue images with cells of similar size. However, the algorithm biases pixel assignment towards smaller masks, since the center of these masks are generally closer to the overlap region than the centers of larger masks. To correct for this, the second expansion algorithm iterates over the masks, expanding each mask by 1 pixel until it collides with another pre-existing mask boundary, at which point growth in that direction stops. The algorithm proceeds until each mask has been expanded by the user-defined number of pixels. This algorithm mitigates the need for assigning pixels based on distance to cell center at the cost of more computation time. Through iterative visual inspection of masks with and without mask expansion, we found that growth by 1 or 2 pixels is usually sufficient to capture most membrane protein signal. The lateral bleed compensation algorithm implemented in the CellSeg pipeline is the same as in previously published work from our group, readers are directed to the original paper for the details of the algorithm [3].

Benchmarking CellSeg

Segmentation of the CRC dataset and multi-tissue TMA was performed on a Dual Intel® Xeon® Silver 4114 10-core CPU. Resulting segmented data was gated in CellEngine (https://cellengine.com). When evaluating fluorescent bleed compensation on the CRC dataset, samples were aggregated by TMA, resulting in two gates for each cell population, one corresponding to each TMA. When performing population correlation analysis, cell types were gated by sample, resulting in tailored gates for the 140 samples. Gating was performed independently for the WTS dataset and the CellSeg dataset in a blinded fashion (J.S.B.). Resulting identified populations were validated by an expert in flow cytometry (C.M.S.). Population correlation analysis was performed in R Studio. The script is available upon request. Plots in Fig. 6a and c were generated using the R package ggplot2 [45]. The tissue image for Fig. 6b was obtained in Fiji/ImageJ [38].

Availability and requirements

Project Name: CellSeg. Project home page: https://michaellee1.github.io/CellSegSite/index.html. Operating system(s): Windows, MacOS, or Linux. Programming language: Python. Other requirements: Web browser, internet connection, Conda, Jupyter, minimum 16 GB RAM, other Python package dependencies listed on project home page. License: MIT. Restrictions for Non-academics: None.

Abbreviations

CODEX: CO-detection by indexing; CPU: Central processing unit; CRC: Colorectal cancer; CSV: Comma-separated values; DFSP: Dermatofibrosarcoma protuberans; FCS: Flow cytometry standard; FFPE: Formalin-fixed paraffin-embedded; GBM: Glioblastoma multiforme; GPU: Graphics processing unit; HCC: Hepatocellular carcinoma; IoU: Intersection over union; mAP: Mean average precision; R-CNN: Region-convolutional neural network; ROI: Region of interest; T-ALL: T-cell acute lymphoblastic leukemia; TMA: Tissue microarray; WTS: Watershed.

Acknowledgements

We thank Yu Xin Wang and Colin Holbrook (Baxter Laboratory for Stem Cell Biology, Stanford University School of Medicine) for constructive feedback on the software and Andrew Janowczyk (University of Lausanne and Swiss Institute of Bioinformatics, Switzerland, and Case Western Reserve University, Cleveland, OH) for critically reading the manuscript.

Authors' contributions

Conceptualization: M.Y.L., S.S.B., G.L.B., D.P. and C.M.S.; Methodology: M.Y.L. and J.S.B.; Software: M.Y.L. and J.S.B.; Validation: M.Y.L., J.S.B., S.S.B., G.L.B. and C.M.S.; Formal Analysis: M.Y.L. and J.S.B.; Investigation: M.Y.L. and J.S.B.; Resources: W.J.F., G.P.N. and C.M.S.; Writing – Original Draft: M.Y.L. and J.S.B.; Writing – Review and Editing: M.Y.L., J.S.B., S.S.B., G.L.B., D.P., W.J.F. and C.M.S.; Supervision: S.S.B., G.L.B. and C.M.S.; Project Administration: J.S.B.; Funding acquisition: S.S.B., D.P., W.J.F., G.P.N. and C.M.S. Reading and approval of the final manuscript: All authors.

Funding

This work was supported by the U.S. National Institutes of Health (2U19AI057229-16, 5P01HL10879707, 5R01GM10983604, 5R33CA18365403, 5U01AI101984-07, 5UH2AR06767604, 5R01CA19665703, 5U54CA20997103, 5F99CA212231-02, 1F32CA233203-01, 5U01AI140498-02, 1U54HG010426-01, 5U19AI100627-07, 1R01HL120724-01A1, R33CA183692, R01HL128173-04, 5P01AI131374-02, 5UG3DK114937-02, 1U19AI135976-01, IDIQ17X149, 1U2CCA233238-01, 1U2CCA233195-01 to G.P.N.; R21CA231280, 1R01CA234553-01A1, P01HL10879709 to W.J.F.); the U.S. Department of Defense (W81XWH-12–1-0591 to G.P.N.; W81XWH-14–1-0180 to W.J.F. and G.P.N.); the U.S. Food and Drug Administration (HHSF223201610018C, DSTL/AGR/00980/01 to G.P.N.); Cancer Research UK (C27165/A29073 to G.P.N.); the Bill and Melinda Gates Foundation (OPP1113682 to G.P.N.); the Cancer Research Institute; the Parker Institute for Cancer Immunotherapy; the Kenneth Rainin Foundation (2018–575); the Silicon Valley Community Foundation (2017–175329 and 2017–177799-5022); the Beckman Center for Molecular and Genetic Medicine; Juno Therapeutics, Inc. (122401); Pfizer, Inc. (123214); Celgene, Inc. (133826, 134073); Vaxart, Inc. (137364); and the Rachford & Carlotta A. Harris Endowed Chair (G.P.N.). W.J.F. was supported by the 2019 Cancer Innovation Award from the Stanford Cancer Institute, an NCI-designated Comprehensive Cancer Center; the Department of Urology at Stanford University; the BRCA Foundation; the V Foundation for cancer; a gift from the Gray Foundation; and a Parker Institute for Cancer Immunotherapy Bedside to Bench grant. C.M.S. was supported by the Swiss National Science Foundation (P300PB_171189, P400PM_183915). D.P. was supported by an NIH T32 Fellowship (AR007422), an NIH F32 Fellowship (CA233203), a Stanford Dean's Postdoctoral Fellowship, a Stanford Cancer Institute Fellowship, and Stanford's Dermatology Department. S.S.B. was supported by a Bio-X Stanford Interdisciplinary Graduate Fellowship and Stanford Bioengineering. G.L.B was supported by NIH 5U01AI101984 as well as an NIH T32 fellowship through the Stanford Molecular and Cellular Immunobiology Program

(5T32AI007290-34). The funding bodies played no role in the design of the study or collection, analysis, or interpretation of data, nor in writing the manuscript.

Availability of data and materials
Raw data and materials used in this study are published and available for download [15, 40]. The gating data and associated statistics shown in Figs. 5 and 6 are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
M.Y.L. is a co-founder of and has equity in Biodock, Inc. G.P.N. is a co-founder and stockholder of Akoya Biosciences, Inc., and inventor on patent US9909167. C.M.S. is a scientific advisor to, has stock options in, and has received research funding from Enable Medicine, Inc. The other authors declare that no competing financial interests exist.

Author details
[1]Department of Microbiology and Immunology, Stanford University School of Medicine, Stanford, CA 94305, USA. [2]Department of Pathology, Stanford University School of Medicine, Stanford, CA 94305, USA. [3]Department of Computer Science, Stanford, CA 94305, USA. [4]Department of Urology, Stanford University School of Medicine, Stanford, CA 94305, USA. [5]Department of Bioengineering, Stanford University School of Medicine, Stanford, CA 94305, USA. [6]Department of Dermatology, Stanford University School of Medicine, Stanford, CA 94305, USA. [7]Stanford Cancer Institute, Stanford University School of Medicine, Stanford, CA 94305, USA. [8]Department of Obstetrics and Gynecology, Stanford University School of Medicine, Stanford, CA 94305, USA. [9]Department of Pathology and Neuropathology, University Hospital and Comprehensive Cancer Center Tübingen, Tübingen, Germany.

Received: 20 April 2021 Accepted: 10 January 2022
Published online: 18 January 2022

References

1. Agasti SS, Wang Y, Schueder F, Sukumar A, Jungmann R, Yin P. DNA-barcoded labeling probes for highly multiplexed exchange-PAINT imaging. Chem Sci. 2017;8:3080–91.
2. Angelo M, Bendall SC, Finck R, Hale MB, Hitzman C, Borowsky AD, et al. Multiplexed ion beam imaging (MIBI) of human breast tumors. Nat Med. 2014;20:436–42.
3. Goltsev Y, Samusik N, Kennedy-Darling J, Bhate S, Hale M, Vazquez G, et al. Deep profiling of mouse splenic architecture with CODEX multiplexed imaging. Cell. 2018;174:968-981.e15.
4. Gut G, Herrmann MD, Pelkmans L. Multiplexed protein maps link subcellular organization to cellular states. Science. 2018;361.
5. Huang W, Hennrick K, Drew S. A colorful future of quantitative pathology: validation of Vectra technology using chromogenic multiplexed immunohistochemistry and prostate tissue microarrays. Hum Pathol. 2013;44:29–38.
6. Lin J-R, Izar B, Wang S, Yapp C, Mei S, Shah PM, et al. Highly multiplexed immunofluorescence imaging of human tissues and tumors using t-CyCIF and conventional optical microscopes. eLife. 2018;7.
7. Saka SK, Wang Y, Kishi JY, Zhu A, Zeng Y, Xie W, et al. Immuno-SABER enables highly multiplexed and amplified protein imaging in tissues. Nat Biotechnol. 2019;37:1080–90.
8. Schubert W, Bonnekoh B, Pommer AJ, Philipsen L, Böckelmann R, Malykh Y, et al. Analyzing proteome topology and function by automated multidimensional fluorescence microscopy. Nat Biotechnol. 2006;24:1270–8.
9. Wang Y, Woehrstein JB, Donoghue N, Dai M, Avendaño MS, Schackmann RCJ, et al. Rapid sequential in situ multiplexing with DNA-exchange-imaging in neuronal cells and tissues. Nano Lett. 2017;17:6131–9.
10. Ali HR, Jackson HW, Zanotelli VRT, Danenberg E, Fischer JR, Bardwell H, et al. Imaging mass cytometry and multiplatform genomics define the phenogenomic landscape of breast cancer. Nat Cancer. 2020;1:163–75.
11. Gerdes MJ, Sevinsky CJ, Sood A, Adak S, Bello MO, Bordwell A, et al. Highly multiplexed single-cell analysis of formalin-fixed, paraffin-embedded cancer tissue. Proc Natl Acad Sci USA. 2013;110:11982–7.
12. Giesen C, Wang HAO, Schapiro D, Zivanovic N, Jacobs A, Hattendorf B, et al. Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry. Nat Methods. 2014;11:417–22.
13. Jackson HW, Fischer JR, Zanotelli VRT, Ali HR, Mechera R, Soysal SD, et al. The single-cell pathology landscape of breast cancer. Nature. 2020;578:615–20.
14. Keren L, Bosse M, Marquez D, Angoshtari R, Jain S, Varma S, et al. A structured tumor-immune microenvironment in triple negative breast cancer revealed by multiplexed ion beam imaging. Cell. 2018;174:1373-1387.e19.
15. Schürch CM, Bhate SS, Barlow GL, Phillips DJ, Noti L, Zlobec I, et al. Coordinated cellular neighborhoods orchestrate antitumoral immunity at the colorectal cancer invasive front. Cell. 2020;182:1341-1359.e19.
16. Phillips D, Matusiak M, Gutierrez BR, Bhate SS, Barlow GL, Jiang S, et al. Immune cell topography predicts response to PD-1 blockade in cutaneous T cell lymphoma. Nat Commun. 2021;12:6726.
17. Schüffler PJ, Schapiro D, Giesen C, Wang HAO, Bodenmiller B, Buhmann JM. Automatic single cell segmentation on highly multiplexed tissue images. Cytometry A. 2015;87:936–42.

18. Zhou X, Li F, Yan J, Wong STC. A novel cell segmentation method and cell phase identification using markov model. IEEE Trans Inf Technol Biomed. 2009;13:152–7.
19. Padfield D, Rittscher J, Roysam B. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. Med Image Anal. 2011;15:650–68.
20. Maška M, Daněk O, Garasa S, Rouzaut A, Muñoz-Barrutia A, Ortiz-de-Solorzano C. Segmentation and shape tracking of whole fluorescent cells based on the Chan-Vese model. IEEE Trans Med Imaging. 2013;32:995–1006.
21. Phillips D, Schürch CM, Khodadoust MS, Kim YH, Nolan GP, Jiang S. Highly multiplexed phenotyping of immunoregulatory proteins in the tumor microenvironment by CODEX tissue imaging. Front Immunol. 2021;12:1763.
22. Xing F, Yang L. Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review. IEEE Rev Biomed Eng. 2016;9:234–63.
23. Godinez WJ, Hossain I, Lazic SE, Davies JW, Zhang X. A multi-scale convolutional neural network for phenotyping high-content cellular images. Bioinformatics. 2017;33:2010–9.
24. Van Valen DA, Kudo T, Lane KM, Macklin DN, Quach NT, DeFelice MM, et al. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLOS Comput Biol. 2016;12:e1005177.
25. Al-Kofahi Y, Zaltsman A, Graves R, Marshall W, Rusu M. A deep learning-based algorithm for 2-D cell segmentation in microscopy images. BMC Bioinform. 2018;19:365.
26. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. Nat Methods. 2021;18:100–6.
27. Greenwald NF, Miller G, Moen E, Kong A, Kagel A, Dougherty T, et al. Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning. Nat Biotechnol. 2021;1–11.
28. Bannon D, Moen E, Schwartz M, Borba E, Kudo T, Greenwald N, et al. DeepCell Kiosk: scaling deep learning–enabled cellular image analysis with Kubernetes. Nat Methods. 2021;18:43–5.
29. Hollandi R, Szkalisity A, Toth T, Tasnadi E, Molnar C, Mathe B, et al. nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer. Cell Syst. 2020;10:453-458.e6.
30. Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF, editors. Medical image computing and computer-assisted intervention—MICCAI 2015. Cham: Springer; 2015. p. 234–41.
31. Isensee F, Jaeger PF, Kohl SAA, Petersen J, Maier-Hein KH. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. Nat Methods. 2021;18:203–11.
32. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. In: 2017 IEEE international conference on computer vision (ICCV). 2017. p. 2980–8.
33. Schmidt U, Weigert M, Broaddus C, Myers G. Cell Detection with Star-convex Polygons. arXiv:180603535 [cs]. 2018;11071:265–73.
34. McQuin C, Goodman A, Chernyshev V, Kamentsky L, Cimini BA, Karhohs KW, et al. CellProfiler 3.0: next-generation image processing for biology. PLOS Biol. 2018;16:e2005970.
35. Berg S, Kutra D, Kroeger T, Straehle CN, Kausler BX, Haubold C, et al. ilastik: interactive machine learning for (bio) image analysis. Nat Methods. 2019;16:1226–32.
36. Kluyver T, Ragan-Kelley B, Perez F, Granger B, Bussonier M, Frederic J, et al. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press; 2016. p. 87–90.
37. Samusik N, Good Z, Spitzer MH, Davis KL, Nolan GP. Automated mapping of phenotype space with single-cell data. Nat Methods. 2016;13:493.
38. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, et al. Fiji: an open-source platform for biological-image analysis. Nat Methods. 2012;9:676–82.
39. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. arXiv:160304467 [cs]. 2016.
40. Caicedo JC, Goodman A, Karhohs KW, Cimini BA, Ackerman J, Haghighi M, et al. Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl. Nat Methods. 2019;16:1247–53.
41. Abdulla W. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. 2017.
42. Lopez-Urrutia A. Deep Retina 3th place solution to Kaggle's 2018 Data Science Bowl. GitHub.
43. Jung AB. https://imgaug.readthedocs.io/en/latest/. Imgaug. 2019. https://imgaug.readthedocs.io/en/latest/.
44. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors., et al., Computer vision—ECCV 2014. Cham: Springer; 2014. p. 740–55.
45. Wickham H. ggplot2: Elegant graphics for data analysis. 2nd edition. Springer; 2016.

Publisher's Note