
Article:
Wiggins, L. orcid.org/0000-0003-4615-2379, Lacy, S., Park, G. et al. (6 more authors) (2025) CellPhePy: a python implementation of the CellPhe toolkit for automated cell phenotyping from microscopy time-lapse videos. Journal of Microscopy. ISSN 0022-2720

https://doi.org/10.1111/jmi.13416

Received: 12 December 2024 | Revised: 12 December 2024 | Accepted: 1 April 2025

DOI: 10.1111/jmi.13416

Journal of Microscopy RMS

RESEARCH ARTICLE

CellPhePy: A python implementation of the CellPhe toolkit for automated cell phenotyping from microscopy time-lapse videos

Laura Wiggins[1] | **Stuart Lacy**[2] | **Graeme Park**[3,4] | **Joanne Marrison**[3,4] | **Ben Powell**[5] | **Beth Cimini**[6] | **Peter O'Toole**[3,4] | **Julie Wilson**[5] | **William J. Brackenbury**[3,4]

[1]Department of Materials Science and Engineering, University of Sheffield, Sheffield, UK

[2]Wolfson Atmospheric Chemistry Laboratories, University of York, York, UK

[3]York Biomedical Research Institute, University of York, York, UK

[4]Department of Biology, University of York, York, UK

[5]Department of Mathematics, University of York, York, UK

[6]Imaging Platform, Cambridge, Massachusetts

Correspondence
Laura Wiggins, Department of Materials Science and Engineering, University of Sheffield, Sheffield, UK.
Email: l.wiggins@sheffield.ac.uk

Funding information
BBSRC, Grant/Award Numbers: BB/S507416/1, BB/Y513970/1; MRC, Grant/Award Number: MR/X018067/1; Wellcome Trust, Grant/Award Number: 310891/Z/24/Z

Abstract

We previously developed the CellPhe toolkit, an open-source R package for automated cell phenotyping from ptychography time-lapse videos. To align with the growing adoption of python-based image analysis tools and to enhance interoperability with widely used software for cell segmentation and tracking, we developed a python implementation of CellPhe, named CellPhePy. CellPhePy preserves all of the core functionality of the original toolkit, including single-cell phenotypic feature extraction, time-series analysis, feature selection and cell type classification. In addition, CellPhePy introduces significant enhancements, such as an improved method for identifying features that differentiate cell populations and extended support for multiclass classification, broadening its analytical capabilities. Notably, the CellPhePy package supports CellPose segmentation and TrackMate tracking, meaning that a set of microscopy images are the only required input with segmentation, tracking and feature extraction fully automated for downstream analysis, without reliance on external applications. The workflow's increased flexibility and modularity make it adaptable to different imaging modalities and fully customisable to address specific research questions. CellPhePy can be installed via PyPi or GitHub, and we also provide a CellPhePy GUI to aid user accessibility.

KEYWORDS

cell phenotyping, image analysis, machine learning, microscopy, open-source, segmentation, timelapse, tracking

1 | INTRODUCTION

Recent years have seen the proliferation of open-source software for the analysis of microscopy images. These tools provide a diverse range of capabilities, ranging from segmentation and phenotypic feature extraction using CellProfiler,[1] CellPose,[2] StarDist[3] and Ilastik,[4] to cell tracking for monitoring temporal changes to cellular phenotypes using software such as TrackMate,[5] CellTracker[6] and DeepCellTracking.[7] With increasingly specialised analysis

requirements, it is becoming more common to integrate multiple tools within a workflow, allowing researchers to take advantage of each software's strengths for specific tasks. A key limitation of this approach is the variety of programming languages used by different software, which introduces additional complexity and necessitates advanced programming skills to integrate these tools effectively into a cohesive workflow. Napari[8] addresses this issue by providing a platform that supports analysis plugins, enabling tools from disparate fields to be used in combination with each other with ease. Notably, Napari is a python-based platform, which has prompted community efforts to port existing software into compatible python plugins. This trend extends beyond Napari itself, as the growing popularity of python for image analysis tasks has led to a broader movement in the community to create python versions of various tools and libraries, as well as increasing functionality to use tools in headless mode within scripts without the need for external graphical user interfaces, exemplified by tools such as Ilastik and CellPose. The integration of Docker containers further streamlines this process by allowing users to package tools along with their dependencies, ensuring consistent and reliable execution in headless environments, and tools such as BiaPy[9] have successfully made use of this.

We previously developed the CellPhe toolkit,[10] an open-source R package for automated cell phenotyping from ptychography time-lapse videos. Recognising the current trend towards python-based analysis and the increased potential of interoperability with existing bioimage analysis tools, we sought to develop a python-based implementation of CellPhe, called CellPhePy, that retains its core functionality of single-cell phenotypic feature extraction, time-series analysis, feature selection and cell type classification. We have also made significant enhancements to the code, including a novel approach for feature selection that ensures the identification of discriminating features is more robust. Additionally, our implementation extends beyond binary classification and offers support for multi-class classification, allowing for the simultaneous characterisation and classification of multiple cell types. CellPhePy also facilitates automated CellPose segmentation and TrackMate tracking, stages that previously required manual intervention through the FIJI graphical user interface. This improvement ensures that the only necessary prerequisite for analysis is a set of microscopy time-lapse images, with all additional steps carried out automatically. CellPhePy can be downloaded and installed via PyPi or from the CellPhePy GitHub repository where you can also find a Jupyter notebook for running the CellPhePy workflow and visualising outputs. We also offer a graphical user interface (GUI) that can be easily installed through the terminal or using Docker. This GUI enables users to run the entire CellPhePy pipeline with minimal coding required, streamlining the process and making it more accessible to a broader user base.

2 | RESULTS

2.1 | The CellPhePy python package

The original CellPhe R package required users to segment and track cells through external software before importing the data into the CellPhe pipeline. Although it supported TrackMate tracking tables, this process required users to be proficient with external tools and introduced additional time investment, creating a barrier to entry for using CellPhe. To address this issue, the CellPhePy package automates the process of CellPose segmentation and TrackMate tracking, meaning that a stack of time-lapse images is the only required input (Figure 1). As well as saving the user's time by removing the need to manually use external software, providing a fully automated interface can save computational time too as it facilitates the deployment of CellPhe on more powerful resources such as High Performance Computing (HPC) clusters or Cloud compute. Following segmentation, the user can export their segmentation masks as well as single-cell regions of interest (ROIs) that are compatible with ImageJ's ROI manager to visualise segmentation performance. Furthermore, CellPhePy's tracking function allows users to select the most suitable TrackMate algorithm for their application, choosing from SimpleSparseLAP, SparseLAP, Kalman, AdvancedKalman, NearestNeighbor, or Overlap. The output file from this segmentation and tracking pipeline is then compatible with the remaining CellPhe functions within the package. These encompass calculation of phenotypic features from each cell on each frame of the time-lapse, as well as summarisation of single-cell time series to quantify temporal changes to cellular phenotypes. Cell populations can then be analysed through CellPhe's feature selection method of 'separation scores'[10] to identify discriminatory variables, integrated XGBoost[11] for cell type classification, and hierarchical clustering to identify heterogeneous phenotypes within a sample. Enhanced modularity allows users to customise the segmentation and tracking workflow by importing their own segmentation masks or tracking tables for use within the CellPhe pipeline. Additionally, users can integrate their own single-cell features into the code before performing time series summarisation, enabling the analysis of temporal changes in custom features. This flexibility provides users with greater control over the workflow, allowing for tailored analyses to suit their specific needs.

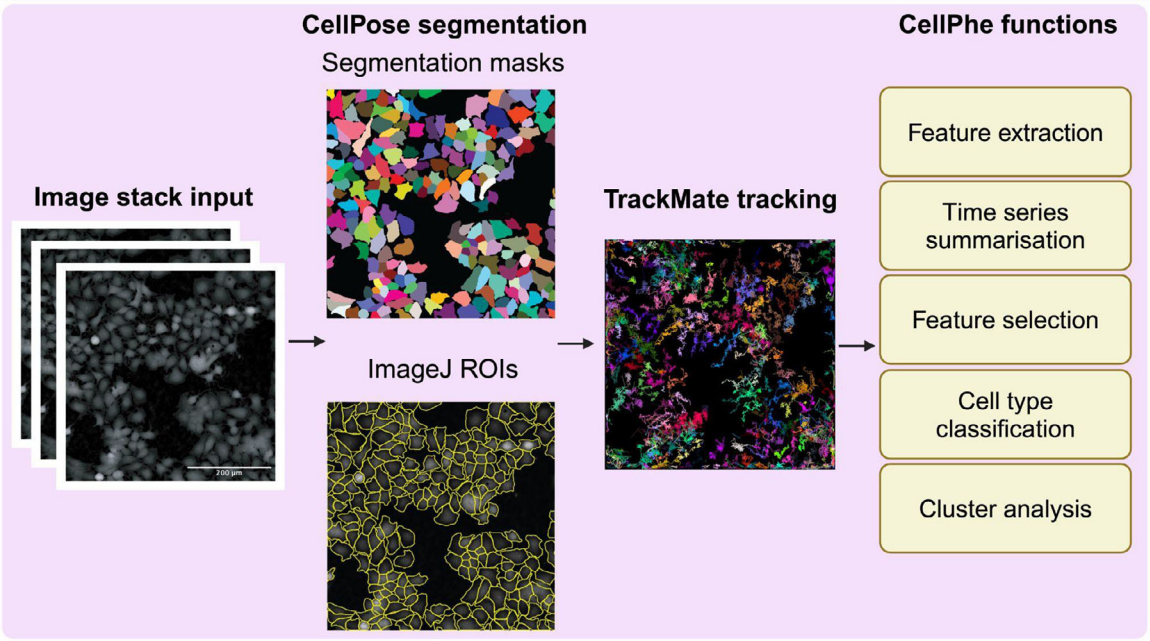

FIGURE 1 A summary of the CellPhePy package workflow. A stack of microscopy time-lapse images is the required input for the CellPhePy workflow. Each image is then automatically segmented using CellPose, where segmentation masks and ImageJ compatible ROI files can be output to assess segmentation performance. Cells are then automatically tracked through TrackMate, and tracking information fed into CellPhe's own functions for downstream analysis.

The CellPhePy python package is available to install through PyPi at https://pypi.org/project/cellphe/ or through GitHub at https://github.com/uoy-research/CellPhePy. Note that the PyPi version 0.4.0 was used to obtain the results presented here. The GitHub page features detailed documentation in the README and a comprehensive tutorial that guides users through the workflow, from importing images to classifying phenotypic changes induced by drug treatment. In addition, a Jupyter notebook is provided, allowing users to execute the workflow with minimal coding intervention. The notebook highlights customisable parameters and offers the ability to visually inspect outputs, enabling users to explore and experiment with the results interactively.

2.2 | Improved identification of discriminatory features

The CellPhePy package includes an improved method for feature selection, enabling more reliable identification of features that effectively discriminate between different cell populations. The CellPhe pipeline determines whether a feature is discriminatory or not through calculation of a separation score that minimises within-group variance while maximising between-group variance, a higher separation score represents a more discriminatory feature. The original CellPhe release provided a method for determining the optimal separation threshold through classification, where the threshold was set to the lowest value that did not result in significant loss of classification accuracy. This approach involved testing classification results for discrete threshold values, which was computationally expensive and limited to fixed points. It also depended on the choice of classification model and its parameters, making it sensitive and potentially inconsistent across different setups. To address these limitations, the CellPhePy package now employs the elbow method, a proven technique for determining the optimal number of clusters in cluster analysis.[12] In this context, the elbow of the plot of ordered separation scores is identified to determine the minimum number of features that provide the greatest separation between cell populations. This approach ensures that the most discriminatory features are selected efficiently, avoiding overfitting while maximising the distinction between groups.

An example of this approach is provided in Figure 2, where the aim of the analysis is to identify the phenotypic features that best discriminate between strongly metastatic MDA-MB-231 cells and weakly metastatic MCF-7 cells (Figure 2A). A total of 1083 features are extracted through CellPhePy, and a plot of all features, coloured by feature category (size, shape, texture, movement or density) is provided in Figure 2B. Through the elbow method, the optimal separation threshold was determined to be 0.09, with 74 features having separation score greater than

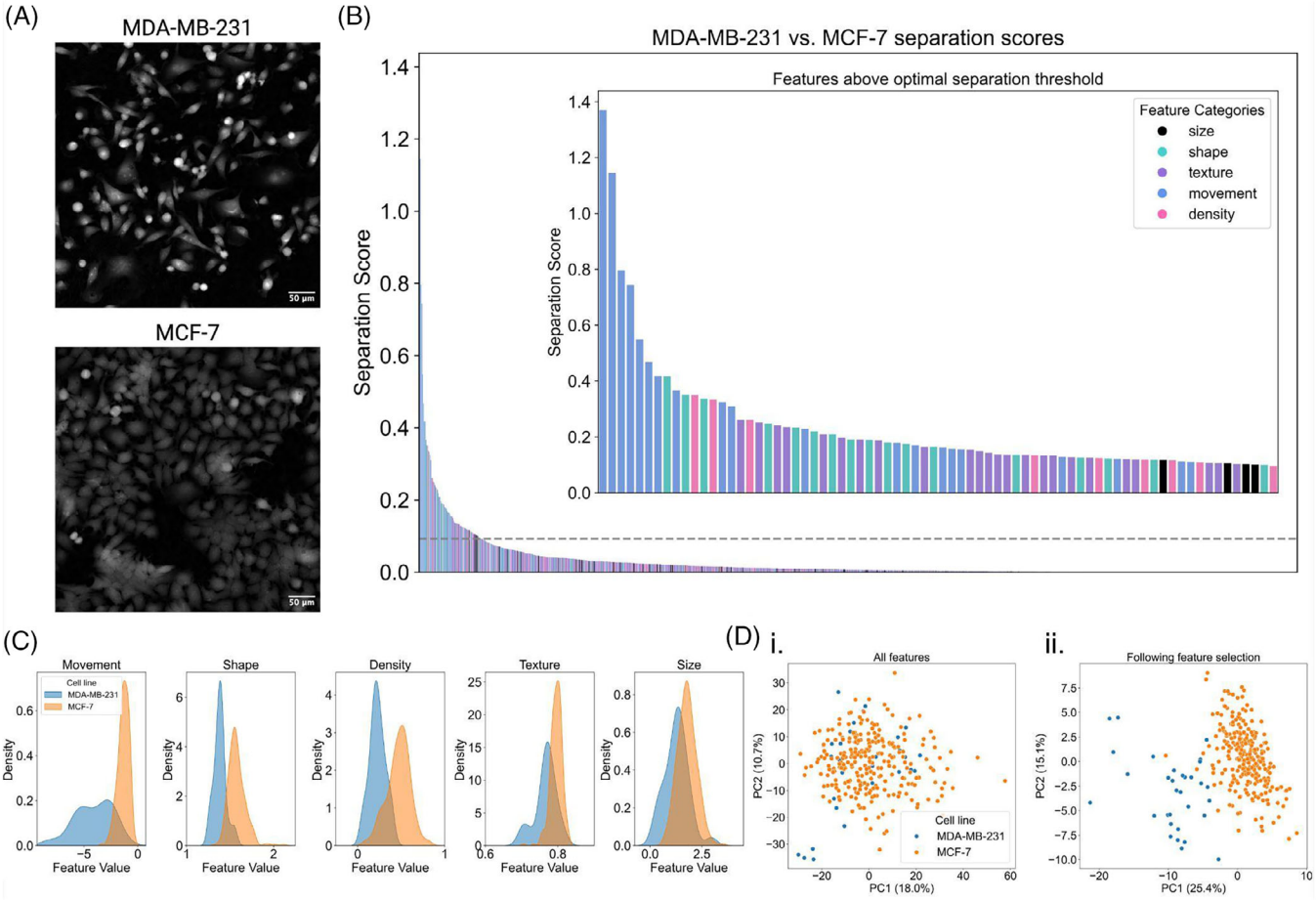

FIGURE 2 Identifying phenotypic features that characterise MDA-MB-231 and MCF-7 cells. (A) Snapshots taken from ptychographic timelapse videos of MDA-MB-231 and MCF-7 monocultures. (B) Barplot of separation scores for all 1083 phenotypic features extracted through CellPhe, coloured by feature category (size, shape, texture, movement, density). The grey dashed line represents the optimal separation threshold (0.09) calculated through the elbow method, with the inset plot providing a zoom in of the 74 features with separation scores greater than or equal to this threshold. (C) Kernel density plots of the top-scoring features for each category, demonstrating relationship between separation score and separation of feature distributions. (D) PCA scores plots when (i) all 1083 features were included and (ii) only the 74 highest-scoring features were included, showing greater separation of cell lines following feature selection.

or equal to this threshold. The top scoring features were primarily related to cellular movement, characterising increased migration of MDA-MB-231 cells in comparison to MCF-7. Such features quantified temporal changes to displacement, surface area covered by cell trajectories, velocity and track length. Kernel density plots of the top-scoring features for each category are shown in Figure 2C, demonstrating how the separation scores correlate with the degree of overlap between the feature distributions of the two cell populations. To illustrate how the 74 retained features collectively characterise cell types, we compare PCA score plots using all 1083 features versus only the 74 highest-scoring features (Figure 2D). When all features are included, the plots show significant overlap between the cell populations, with no clear separation even along the first principal component. After feature selection, however, there is distinct separation between the cell populations, with greatest separation achieved along the first principal component.

2.3 | CellPhePy enables multiclass characterisation and classification

While the initial release of CellPhe successfully enabled the characterisation and classification of two cell types, biological experiments often require the analysis of multiple cell types simultaneously. To accommodate this, we adapted the calculation of separation scores in CellPhe to identify features that provide optimal separation across multiple cell types at once. In addition, CellPhe's original approach to classification was through an ensemble of LDA, Random Forest and SVM classifiers, where final classification was based on majority vote, treating each

classifier's predictions equally. While this works for binary classification, where one class will always receive a majority vote, it was not suitable for multiclass cases. To address this, we replaced the discrete three classifier ensemble with XGBoost, which is an ensemble of decision trees that can inherently handle multiclass problems as well as automatically weighting its members' predictions.,

CellPhePy's multiclass classification capabilities offer flexibility in defining and adjusting groupings to suit the specific goals of an analysis. In our example, we classify and characterise four breast cancer cell lines (MDA-MB-231, MCF-7, SkBr3, and BT-474) alongside a healthy breast epithelial cell line (MCF-10A). Figure 3B(i) illustrates how multiclass feature selection effectively separates these cell lines, with an optimal separation threshold of 0.45, retaining 97 phenotypic features. When the cell lines are relabelled as 'cancer' for the breast cancer lines and 'healthy' for MCF-10A, separation scores are recalculated to reflect disease status rather than individual cell lines. This relabelling results in an optimal separation threshold of 0.2, with 74 features identified as most discriminatory, further improving group separation in Figure 3B(ii). Lastly, with the healthy cell line removed, the remaining breast cancer cell lines are relabelled by their clinical molecular subtype: TNBC for MDA-MB-231 and MDA-MB-468, Luminal A for MCF-7, and Her2+ for SkBr3. Separation scores are recalculated to capture molecular subtype distinctions, yielding an optimal separation threshold of 0.28 and 141 features achieving separation scores above this threshold. Figure 3B(iii) demonstrates the success of this approach in label-free characterisation of molecular subtypes based on phenotypic data.

CellPhePy's classification capabilities were applied across three different analytical aims: cell line classification, disease status classification, and molecular subtype classification. For each of these objectives, an XGBoost model was trained using the relevant features, and the model was then used to classify unseen data (Figure 3C). For this analysis, classification accuracy percentages are reported as the TPR, that is, the percentage of cells within a class correctly classified as that class. For cell line classification, accuracy percentages were as follows: MCF-10A = 90%, MCF-7 = 94%, MDA-MB-231 = 61%, MDA-MB-468 = 90%, SkBr3 = 98% (sample sizes are detailed in the Methods). For disease status classification, accuracy scores were Cancer = 97%, Healthy = 85%. For molecular subtype classification, accuracy scores were Her2+ = 98%, Luminal A = 98%, TNBC = 89%. This demonstrates how CellPhePy's flexible classification approach can be easily adapted to different analytical objectives, making it suitable for a range of applications, including diagnosis, drug screening, and biomarker discovery.

2.4 | A CellPhePy GUI for enhanced user accessibility

We developed a CellPhePy GUI to make the toolkit accessible to all users, regardless of their programming expertise. The user is required to provide the path to a folder containing a stack of microscopy time-lapse images on the landing page. Following this, they can click a button to process the images which will perform CellPose segmentation, TrackMate tracking, extraction of single-cell features, and CellPhe's time series summarisation. The resulting feature tables can then be used as input for the 'Single Population' and 'Multiple Populations' tabs, which provide dashboards for analysing individual populations or comparing multiple populations, respectively (Figure 4). The GUI is run locally, giving users full control over their images without the need to upload them anywhere. It is cross-platform and operates within a web browser, ensuring accessibility on different operating systems. The program is openly available on GitHub (https://github.com/uoy-research/CellPhe-dashboard) and can be run directly using python, or using Docker.

3 | DISCUSSION

The findings of '2020 BioImage Analysis Survey: community experiences and needs for the future'[13] highlighted the bioimage analysis community's desire for developers to improve tool accessibility through comprehensive documentation, intuitive user interfaces, and more user-friendly plugins or packages. In addition, developers were encouraged to make their tools interoperable with popular existing platforms. These recommendations motivated the refactoring of CellPhe to CellPhePy, with the transition from R to python enabling the tool to integrate with existing python-based tools such as CellPose. Furthermore, incorporation of PyImageJ[14] enabled headless automation of TrackMate tracking, eliminating the previous requirement to handle this externally through the ImageJ GUI.

Increased modularity of CellPhePy ensures that users are not restricted to a single method for cell segmentation and tracking. For example, users are now able to provide their own labelled masks generated through a software of their choice. Our headless TrackMate tracking code also supports all TrackMate tracking algorithms, including SimpleSparseLAP, SparseLAP, Kalman, Advanced-Kalman, NearestNeighbor and Overlap, allowing users to tailor analysis workflows for their specific samples and research needs. This flexibility makes CellPhePy a versatile analysis tool for a wide range of imaging modalities,

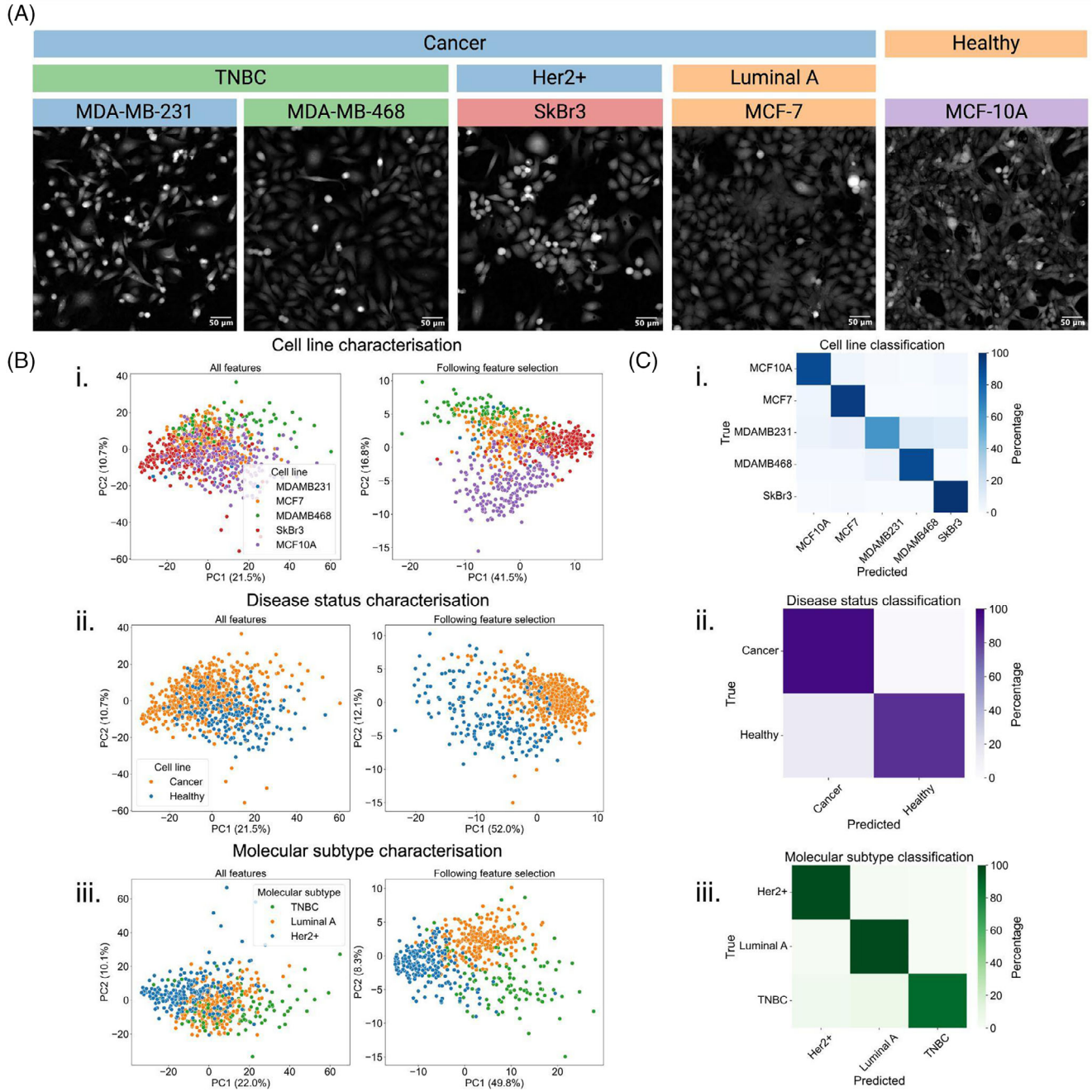

FIGURE 3 An example of multiclass classification through CellPhePy. (A) Snapshots taken from ptychographic timelapse videos of MDA-MB-231, MDA-MB-468, SkBr3, MCF-7 and MCF-10A monocultures. Labels used for the analysis included in this figure are provided in the image headings, coloured as they appear in the PCA scores plots in b. (B) PCA scores plots before and after feature selection for three characterisation applications: (i) cell line characterisation, (ii) disease status characterisation and (iii) molecular subtype characterisation, demonstrating greater separation of cell populations following custom feature selection. (C) Confusion matrices of test set classification accuracy percentages for XGBoost classification of (i) cell lines, (ii) disease status, and (iii) molecular subtype.

including ptychography, brightfield and fluorescence. Users can simply replace CellPhePy's default use of CellPose's Cyto model with another built-in CellPose model listed on the CellPose documentation page, or use an alternative segmentation method more appropriate for their sample and imaging modality. Users can also add their own custom features within CellPhePy's feature extraction script, enabling temporal analysis of more sample- and modality-specific features such as changes in intensity or co-localisation of fluorescence markers, for example.

Extending multiclass cell type characterisation and classification capabilities within CellPhePy significantly broadens its potential applications, enabling analysis within more complex experimental setups like large-scale

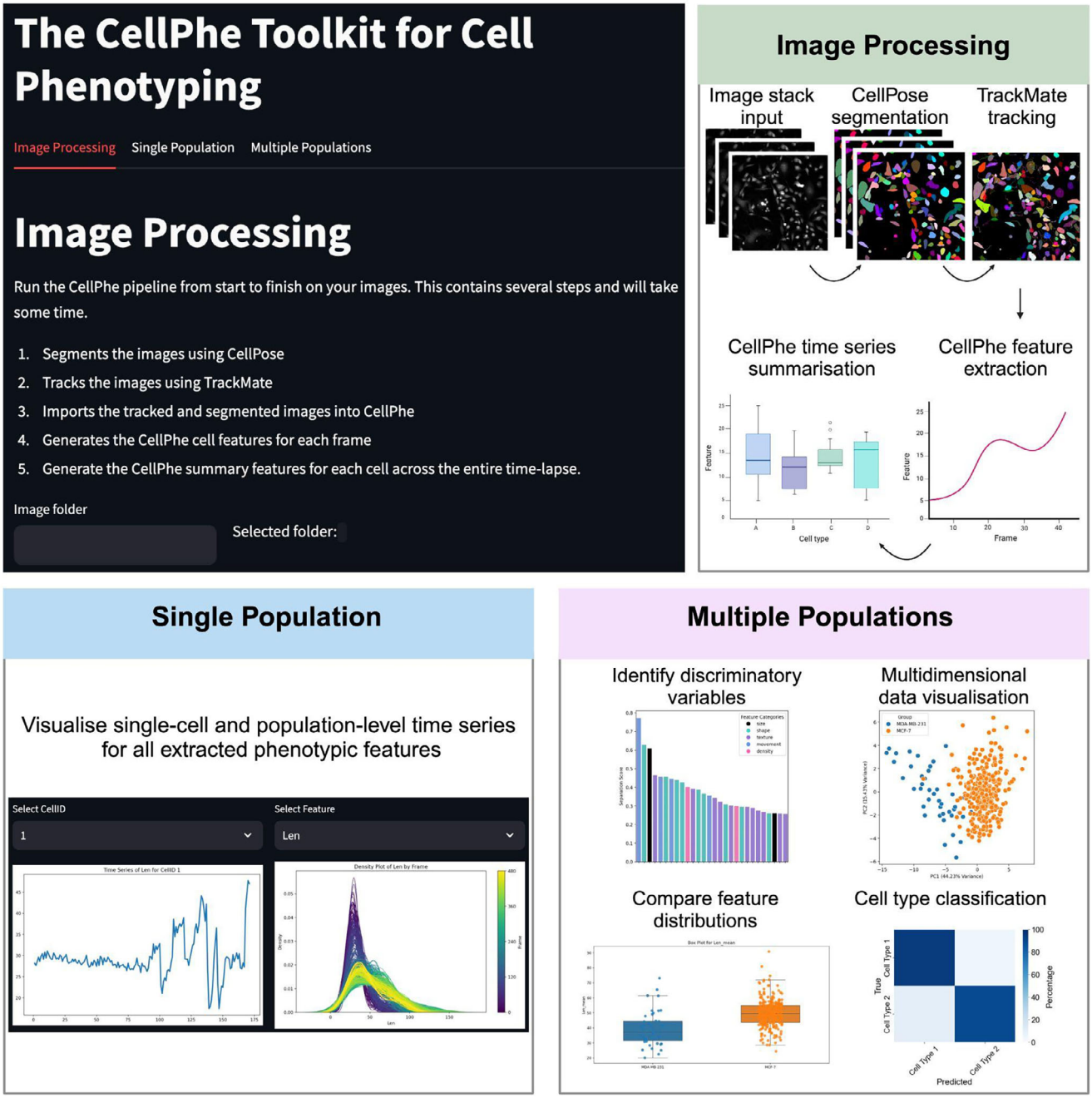

FIGURE 4 The CellPhePy GUI for enhanced user accessibility. An overview of the three tabs within the CellPhePy GUI: Image Processing, Single Population and Multiple Populations. The image processing tab requires the user to specify a folder path to their stack of microscopy time-lapse images, then automates CellPose segmentation, TrackMate tracking and CellPhe feature extraction. The Single Population tab can be used to visualise single-cell feature time series as well as temporal population-level feature changes. The Multiple Populations tab enables multi-class characterisation and classification of cell types, allowing users to visualise which features are important in distinguishing between their cell populations.

drug screening assays or co-culture experiments. Data can also be repurposed by redefining classes to handle different research questions. For instance, as demonstrated here, groups can be redefined to reveal phenotypic differences between cancerous and healthy states, as well as to distinguish molecular subtypes, thereby increasing its utility for disease diagnosis or study of heterogeneity. We emphasise that multiclass classification may not always be the most suitable approach for a given research question, and this will be reflected in suboptimal results. For example, here we demonstrated that classifying all breast cell lines individually led to poor classification accuracy for MDA-MB-231 cells, with a proportion being misclassified as MDA-MB-468 cells due to shared

characteristics between these TNBC cell lines. In this case, a more effective approach would be to use a secondary classifier that focuses on identifying differences between MDA-MB-231 and MDA-MB-468, specifically highlighting the phenotypic distinctions between these two cell lines.

In conclusion, herein we have provided a description of the new python CellPhePy implementation, providing a more adaptable, easier to use version of software for wider uptake of time-lapse imaging data analysis across modalities.

4 | MATERIALS AND METHODS

4.1 | Cell culture

The MDA-MB-231 cells were a gift from M. Djamgoz, Imperial College, London. SkBr3 cells were a gift from J. Rae, University of Michigan and MCF-10A cells were a gift from N. Maitland, University of York. MDA-MB-468 and MCF-7 cells were from ATCC. The molecular identity of all cell lines was verified by short tandem repeat analysis.[15] Authenticated cell stocks were stored in liquid nitrogen and thawed for use in experiments. Thawed cells were subcultured 4–5 times prior to discarding and thawing a new stock to ensure that the molecular identity of cells was retained throughout. All cell lines, except for MCF-10A, were cultured separately in Dulbecco's Modified Eagle Medium (DMEM) supplemented with 5% fetal bovine serum (FBS) and 4 mM L-glutamine. MCF-10A cells were cultured in DMEM/F12 (Invitrogen) with 5% heat-inactivated horse serum, 0.5 μg/mL hydrocortisone, 20 ng/mL human EGF, 10 μg/mL insulin, and 100 ng/mL cholera toxin. To minimise imaging artefacts, FBS was filtered using a 0.22 μm syringe filter before use. Cells were incubated at 37°C in plastic filter-cap T-25 flasks and were split at a 1:6 ratio during passaging. No antibiotics were added to the cell culture medium. Cells were confirmed to be free of mycoplasma before use in experiments through routine DAPI testing at monthly intervals.

4.2 | Image acquisition and exportation

On the day of imaging, cells were placed onto the Phasefocus Livecyte 2 (Phasefocus Limited, Sheffield, UK) and incubated for 30 min before image acquisition to allow for temperature equilibration. A 500 μm × 500 μm field of view per well was imaged to capture as many cells, and therefore data observations, as possible. Selected wells were imaged in parallel for ~22 h at 20× magnification with 6-min intervals between frames, resulting in full time-lapses of 222 frames per imaged well. For phase images, Phasefocus' Cell Analysis Toolbox® software was utilised for image processing, such as use of the rolling ball algorithm to reduce background noise, as well as image exportation.

4.3 | Cell type classification

For the multiclass classification examples presented in this study, the training data comprised 43 MDA-MB-231, 73 MDA-MB-468, 213 SkBr3, 205 MCF-10A, and 268 MCF-7 cells. The corresponding test data included 36 MDA-MB-231, 48 MDA-MB-468, 161 SkBr3, 205 MCF-10A, and 294 MCF-7 cells. Classification models were trained, and test sets were evaluated using CellPhePy's `classify_cells()` function. A confusion matrix summarising classification accuracy percentages was generated using the `sklearn.metrics.confusion_matrix()` function.

Train and test data were divided into different groupings to facilitate three analytical objectives: cell line classification, disease status classification, and molecular subtype classification. In these examples, the data were split as follows:

- Cell line classification: MDA-MB-231, MDA-MB-468, SkBr3, MCF-10A, and MCF-7.
- Disease status classification: Cancer (MDA-MB-231, MDA-MB-468, SkBr3, MCF-7) vs. Healthy (MCF-10A).
- Molecular subtype classification: TNBC (MDA-MB-231, MDA-MB-468) vs. Her2+ (SkBr3) vs. Luminal A (MCF-7).

4.4 | CellPhePy

4.4.1 | Multiclass separation scores

Let G be the number of classes. n_g denotes the sample size of class g, $\underline{x}_g$ denotes the sample mean of a feature for class g and s_g^2 denotes the sample variance of a feature for class g.

The grand mean of a feature is given by:

$$\bar{\bar{x}} = \frac{1}{N}\sum_{g=1}^{G} n_g \bar{x}_g \text{ where } N = n_1 + n_2 + n_3 + \cdots + n_G$$

The feature's between-class variance is given by:

$$V_B = \frac{1}{N-G}\sum_{g=1}^{G} n_g(\bar{x}_g - \bar{\bar{x}})^2$$

And the feature's within-class variance is given by:

$$V_W = \frac{1}{N-G}\sum_{g=1}^{G} (n_g - 1)\, s_g^2$$

The feature's separation score can then be calculated by:

$$\text{Separation} = \frac{VB}{VW}$$

4.5 | Determining optimal separation score threshold

To determine the optimal subset of features, we applied an elbow method based on feature separation scores. Separation scores for all features were sorted in descending order, and a straight line was interpolated between the maximum and minimum scores. For each feature, the vertical distance between its separation score and the interpolated line was calculated. The 'elbow point' was identified as the feature with the maximum distance from the line, representing the transition from steep score improvement to a plateau. We then define the optimal separation score threshold as this elbow point, with all downstream classification tasks including only features with separation scores greater than or equal to the optimal threshold.

4.6 | CellPose segmentation

CellPose integration is provided through the official python package: cellpose (https://pypi.org/project/cellpose/, version 3.1.0)

4.7 | TrackMate tracking

Object tracking is provided by the TrackMate ImageJ plugin, interfaced via the PyImagej python package (https://pypi.org/project/pyimagej/, version 1.5.0). This package leverages the scyjava python package (https://pypi.org/project/scyjava/, version 1.10.0) to interface between the Java Virtual Machine running ImageJ, and python code.

4.8 | The CellPhePy GUI

The CellPhePy GUI is developed using the Streamlit framework, available as a python package (https://pypi.org/project/streamlit/, version 1.41.0). The GUI can be run natively as a python script, or as part of a Docker image that bundles all the dependencies.

AUTHOR CONTRIBUTIONS

Conceptualisation: Laura Wiggins, William J. Brackenbury, Peter O'Toole, Beth Cimini, Stuart Lacy. *Data curation*: Laura Wiggins, Graeme Park, Joanne Marrison, Stuart Lacy. *Formal analysis*: Laura Wiggins. *Funding acquisition*: Laura Wiggins, William J. Brackenbury, Peter O'Toole, Beth Cimini, Ben Powell. *Software*: Laura Wiggins, Stuart Lacy, Beth Cimini. *Supervision*: Laura Wiggins, William J. Brackenbury, Peter O'Toole. *Validation*: Laura Wiggins, Stuart Lacy. *Visualisation*: Laura Wiggins. *Writing – original draft*: Laura Wiggins. *Writing – review and editing*: Laura Wiggins, William J. Brackenbury, Stuart Lacy, Beth Cimini, Ben Powell.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

The CellPhePy python package is available to install through PyPi at https://pypi.org/project/cellphe/ or through GitHub at https://github.com/uoy-research/CellPhePy. The CellPhePy GUI is also hosted on GitHub (https://github.com/uoy-research/CellPhe-dashboard) and can be installed through the terminal or Docker.

REFERENCES

1. Mcquin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., Doan, M., Ding, L., Rafelski, S. M., Thirstrup, D., Wiegraebe, W., Singh, S., Becker, T., Caicedo, J. C., & Carpenter, A. E. (2018). CellProfiler 3.0: Next-generation image processing for biology. *PLoS Biology*, *16*, e2005970.
2. Stringer, C., Wang, T., Michaelos, M., & Pachitariu, M. (2021). Cellpose: A generalist algorithm for cellular segmentation. *Nature Methods*, *18*, 100–106.
3. Schmidt, U., Weigert, M., Broaddus, C., & Myers, G. (2018). Cell detection with star-convex polygons. In Frangi, A., Schnabel, J., Davatzikos, C., Alberola-López, C., Fichtinger, G. (Eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018* (pp. 265–273). Springer International Publishing.
4. Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., Eren, K., Cervantes, J. I., Xu, B., Beuttenmueller, F., Wolny, A., Zhang, C., Koethe, U., Hamprecht, F. A., & Kreshuk, A. (2019). Ilastik: Interactive machine learning for (bio)image analysis. *Nature Methods*, *16*(12), 1226–1232.
5. Ershov, D., Phan, M.-S., Pylvänäinen, J. W., Rigaud, S. U., Le Blanc, L., Charles-Orszag, A., Conway, J. R. W., Laine, R. F., Roy, N. H., Bonazzi, D., Duménil, G., Jacquemet, G., & Tinevez, J.-Y. (2022). TrackMate 7: Integrating state-of-the-art segmentation algorithms into tracking pipelines. *Nature Methods*, *19*, 829–832.
6. Piccinini, F., Kiss, A., & Horvath, P. (2016). CellTracker (not only) for dummies. *Bioinformatics*, *32*, 955–957.
7. Schwartz, M. S., Moen, E., Miller, G., Dougherty, T., Borba, E., Ding, R., Graf, W., Pao, E., & Van Valen, D. (2019). Caliban: Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. *bioRxiv*. https://doi.org/10.1101/803205
8. Chiu, C.-L., Clack, N., & the napari community. (2022). Napari: A python multi-dimensional image viewer platform for the research community. *Microscopy and Microanalysis*, *28*, 1576–1577.

9. Franco-Barranco, D., Román, J. A. A.-S., Hidalgo-Cenalmor, I., Backová, L., González-Marfil, A., Caporal, C., Chessel, A., Gómez-Gálvez, P., Escudero, L. M., Wei, D., Muñoz-Barrutia, A., & Arganda-Carreras, I. (2024). BiaPy: A unified framework for versatile bioimage analysis with deep learning. https://doi.org/10.1101/2024.02.03.576026
10. Wiggins, L., Lord, A., Murphy, K. L., Lacy, S. E., O'toole, P. J., Brackenbury, W. J., & Wilson, J. (2023). The CellPhe toolkit for cell phenotyping using time-lapse imaging and pattern recognition. *Nature Communications*, *14*, 1854.
11. Brownlee, J. (2016). *XGBoost with Python: Gradient boosted trees with XGBoost and Scikit-Learn*. Machine Learning Mastery.
12. Shi, C., Wei, B., Wei, S., Wang, W., Liu, H., & Liu, J. (2021). A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2021*, 31.
13. Jamali, N., Dobson, E. T., Eliceiri, K. W., Carpenter, A. E., & Cimini, B. A. (2022). 2020 BioImage Analysis Survey: Community experiences and needs for the future. *Biological Imaging*, *1*, e4.
14. Rueden, C. T., Hiner, M. C., Evans, E. L., Pinkert, M. A., Lucas, A. M., Carpenter, A. E., Cimini, B. A., & Eliceiri, K. W. (2022). PyImageJ: A library for integrating ImageJ and Python. *Nature Methods*, *19*, 1326–1327.
15. Masters, J. R., Thomson, J. A., Daly-Burns, B., Reid, Y. A., Dirks, W. G., Packer, P., Toji, L. H., Ohno, T., Tanabe, H., Arlett, C. F., Kelland, L. R., Harrison, M., Virmani, A., Ward, T. H., Ayres, K. L., & Debenham, P. G. (2001). Short tandem repeat profiling provides an international reference standard for human cell lines. *Proceedings of the National Academy of Sciences of the United States of America*, *98*, 8012–8017.

How to cite this article: Wiggins, L., Lacy, S., Park, G., Marrison, J., Powell, B., Cimini, B., O'Toole, P., Wilson, J., & Brackenbury, W. J. (2025). CellPhePy: A python implementation of the CellPhe toolkit for automated cell phenotyping from microscopy time-lapse videos. *Journal of Microscopy*, 1–10. https://doi.org/10.1111/jmi.13416