https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

Technical Note: colab_zirc_dims: a Google-Colab-based Toolset for Automated and Semi-automated Measurement of Mineral Grains in LA-ICP-MS Images Using Deep Learning Models

Michael C. Sitar[1], Ryan J. Leary[2]

[1]Department of Geosciences, Colorado State University, Fort Collins, CO, 80523-1482, USA
[2]Department of Earth and Environmental Science, New Mexico Institute of Mining and Technology, Socorro, NM, 87801, USA

Correspondence to: Michael C. Sitar (mcsitar@colostate.edu)

Abstract

Collecting grain measurements for large detrital zircon age datasets is time-consuming, but a growing number of studies suggest such data are essential to understanding complex roles of grain size and morphology in grain transport and as indicators for grain provenance. We developed the colab_zirc_dims Python package to automate deep-learning-based segmentation and measurement of mineral grains from scaled images captured during laser ablation at facilities that use Chromium targeting software. The colab_zirc_dims package is implemented in a collection of freely accessible, ready-to-run Google Colab notebooks with additional functionalities for dataset preparation and semi-automated grain segmentation and measurement using a simple graphical user interface. Our automated grain measurement algorithm approaches human measurement accuracy when applied to a manually measured n = 5,004 detrital zircon dataset, but persistent errors necessitate semi-automated measurement for production of publication-quality datasets. We estimate that our semi-automated grain segmentation workflow will enable users to collect grain measurements for large ($n \geq 5,000$), applicable datasets in under a day of work, and we hope that the colab_zirc_dims toolset allows more researchers to augment their detrital geochronology datasets with grain measurements.

1 Introduction

Despite an increasing number of studies on the subject, the degree to which detrital geochronology datasets are affected by sample and mineral grain size remains unresolved. Several detrital zircon studies have documented substantial grain size-dependent mineral fractionation leading to biased detrital age spectra and erroneous provenance interpretations (e.g., Lawrence et al., 2011; Ibañez-Mejia et al., 2018; Augustsson et al., 2018; Cantine et al., 2021), whereas several other studies have identified provenance-dependent grain size relationships in detrital samples but have found little evidence of age spectra biasing by selective transport processes (e.g., Muhlbauer et al., 2017; Leary et al., 2020, accepted). Because the number of studies characterizing grain size of detrital zircon datasets remains relatively small, especially compared to the number of studies employing detrital zircon geochronology, we likely lack the necessary volume and diversity of datasets to understand under which specific circumstances zircon transport processes will bias age spectra and interpreted provenance (Leary et al., accepted). A principal challenge in collecting such data has been that few automated approaches have been published, and manual collection of grain dimensions from large detrital datasets has been a barrier to such studies.

Zircon grains can be measured manually using analogue methods prior to LA-ICP-MS but doing so is prohibitively time consuming. Grains may also be imaged, characterized, and measured via scanning electron microscope before or after analysis, but this too incurs time and instrumentation costs that increase with sample size, and such analyses are not standard at most labs. Many LA-ICP-MS facilities using Teledyne-Photon machines laser ablation systems with proprietary Chromium (Teledyne Photon Machines, 2020) targeting software save reflected light images of samples during analysis with scaling and shot location metadata files and provide these files to facility users. Images from these facilities may be full-sample mosaics captured prior to analyses or single, grain-centred per-shot images captured during ablation. The former are provided by the University of Arizona LaserChron Center (ALC) and the latter by the University of California, Santa Barbara (UCSB) Petrochronology Center. Many researchers who have not otherwise imaged their large-n detrital mineral datasets do have access to these files, and these can be used to locate and manually measure detrital mineral grains using the offline version of the Chromium targeting software (Leary et al., 2020).

Three limitations to manual grain measurement in Chromium (Leary et al., 2020) are a) grains may be partially exposed at the surfaces of epoxy mounts, so measurements are minimum, rather than true dimensions, b) this method is extremely time consuming, and c) this method can only produce one-dimensional (i.e., length) measurements. The first problem is inherent to reflected light images, but the latter two can be mitigated and solved, respectively, via automated two-dimensional grain-image segmentation and measurement of segmentation results. Deep learning methods, wherein training-optimizable models are used to algorithmically extract information from data (e.g., images) with minimal pre-processing (Alzubaidi et al., 2021), are at the cutting edge of accuracy in image segmentation and so allow grain image segmentation to be automated to a greater degree than other methods (e.g., thresholding).

We developed the colab_zirc_dims Python package, which contains code to automatically segment and measure mineral grains from Chromium-scaled LA-ICP-MS reflected light images using deep learning instance segmentation (i.e., where grains are treated as separate objects and distinguished from one another) models. Such models are computationally expensive to run and can be quite slow without a good graphics processing unit (GPU), so we implemented our code in Google Colab notebooks for maximally accessible public use. Google Colab is a free service that allows users to run Jupyter notebooks (i.e., Kluyver et al., 2016) on cloud-based virtual machines with high-end GPUs. Because its user interface is Colab-notebook-based, colab_zirc_dims is not a per-se application but a set of simplified, highly interactive scripts that rely on a backend of code in the colab_zirc_dims package. Deep-learning-based techniques are increasingly geologic image segmentation tasks (e.g., fission track counting (Nachtergaele and De Grave, 2021), cobble measurement (Soloy et al., 2020), and photomicrograph grain segmentation (e.g., Bukharev et al., 2018; Jiang et al., 2020; Latif et al., 2022)), but the colab_zirc_dims package and processing notebooks represent, to the best of our knowledge, the first deep-learning-based approach to per-grain detrital mineral separate measurement.

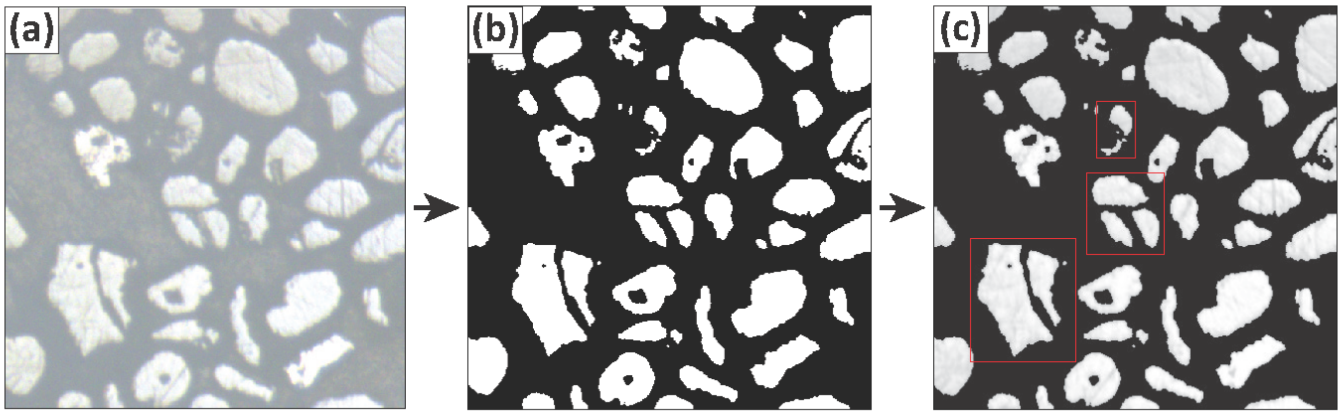
2 Established image segmentation techniques and related software

Automated segmentation of mineral grains in LA-ICP-MS images can be achieved with some success using relatively simple image segmentation techniques such as k-means clustering, edge detection, and intensity thresholding. Otsu's thresholding method (i.e., Otsu, 1979), wherein image pixels are automatically segmented into background and foreground classes with via maximization of inter-class intensity variance, is particularly well-suited for reflected light images because mineral grains appear as a bright phase against an epoxy background (Fig. 1). Though grain segmentations produced through Otsu thresholding are often accurate, they tend to split single fractured grains into multiple sub-grains (Fig. 1c) and can be wildly inaccurate where image artefacts affecting pixel intensity (e.g., anomalous bright spots) are present. These problems are common to automated segmentation techniques, and edge detection methods additionally contend with mis-segmentations along artefactitious edges where sub-image boundaries appear within larger, otherwise uniform mosaic images. Because deep learning models can be optimized through training to ignore image artefacts and intra-grain fractures, they are likely the best available tool for achieving fully automated mineral grain segmentations with near-human accuracy.

Some existing software applications enable measurement of mineral grains in images with varying degrees of automation. The offline version of the Chromium LA-ICP-MS targeting application supports loading and viewing of scaled alignment images and shot locations; users can manually measure the axial dimensions of analysed grains using a ruler-like "measure" tool (Leary et al., 2020; Teledyne Photon Machines, 2020). The ZirconSpotFinder module of the MATLAB-based AgeCalcML application likewise supports loading and viewing of Chromium-scaled LA-ICP-MS alignment images, but also implements semi-automated grain segmentation using user-selected thresholds, filtering of segmented grains by surface area, and export of area-filtered shot lists (Sundell et al., 2020). AnalyZr, a new application designed specifically for measurement of zircon grains in images, combines Otsu thresholding with a novel boundary separation algorithm to automatically segment grains and allows users to edit the resulting segmentations before exporting automatically-generated, grain-specific dimensional analyses (Scharf et al., 2022). Because AnalyZr supports loading of grain image .png files from any source with manual capture of image scale, it can be used to extract more detailed per-grain information (e.g., zonal area from cathodoluminescence images, unobscured grain dimensions from transmitted light images) than is obtainable using only reflected light images (Scharf et al., 2022). AnalyZr's manual scaling implementation and thresholding-based segmentation algorithm also, however, necessitate substantial human involvement in producing accurate grain segmentations and measurements. The colab_zirc_dims package and notebooks are likely better suited for rapid measurement of mineral grains in applicable (i.e., with Chromium-scaled images) large-n datasets due to their automated image loading, scaling, and generally accurate deep-learning-based automated segmentation capabilities.

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

Figure 1. A visualization of image thresholding segmentation using Otsu's method (Otsu, 1979) and its inherent problems in the context of reflected light detrital zircon grain images. (a) An original, unaltered LA-ICP-MS reflected light image. (b) A binary image resulting from segmentation of the original image into foreground (white) and background (black) classes using Otsu's method. (c) The original image with "background" masked out using the binary image; fractured single grains that have been erroneously split into multiple grains are highlighted in red.



3 Methods

3.1 Dependencies

The colab_zirc_dims package was written in Python 3.8 and relies on some non-standard Python packages (Van Rossum, 2020). Pillow, OpenCV, and Matplotlib are respectively used for image loading, image display, and to create and save verification segmentation images; Matplotlib was additionally used to create figures for this manuscript (Umesh, 2012; Bradski, 2000; J. D. Hunter, 2007). NumPy is used for array operations and conversions, and pandas is used in some contexts for data organization and export (Harris et al., 2020; McKinney, 2010). The ".measure" module of Scikit-Image is used to produce unscaled dimensional analyses from segmented grain masks and to extract mask outlines for conversion into user-editable polygons (van der Walt et al., 2014). Interactivity in colab_zirc_dims processing notebooks is implemented using IPython (Pérez and Granger, 2007). Detectron2, which is a deep learning library that was developed by Facebook and is itself built on Google's PyTorch, was used for model construction and training and is used to deploy models within colab_zirc_dims processing notebooks (Detectron2; Paszke et al., 2019).

Our notebook-based implementation of the colab_zirc_dims package relies, obviously, on Google Colab to run. We recognize that Google Colab is an unconventional platform for final deployment of scientific computing algorithms and that it does have some significant disadvantages (e.g., runtimes will automatically disconnect if left idle for too long) versus local, application-based deployment. Nevertheless, we believe that Google Colab's benefits in this use-case outweigh its disadvantages, especially with regards to accessibility. The colab_zirc_dims notebooks can be run using otherwise-expensive GPUs by anyone with a Google account, regardless of their local hardware or prior Python experience. We also mitigate potential connection-related issues by implementing automatic saving to Google Drive during automated and semi-automated grain-image processing: if a user's runtime disconnects, they can simply re-connect and resume work from the last sample processed before disconnection.

3.2 Deep learning models

3.2.1 Mask RCNN

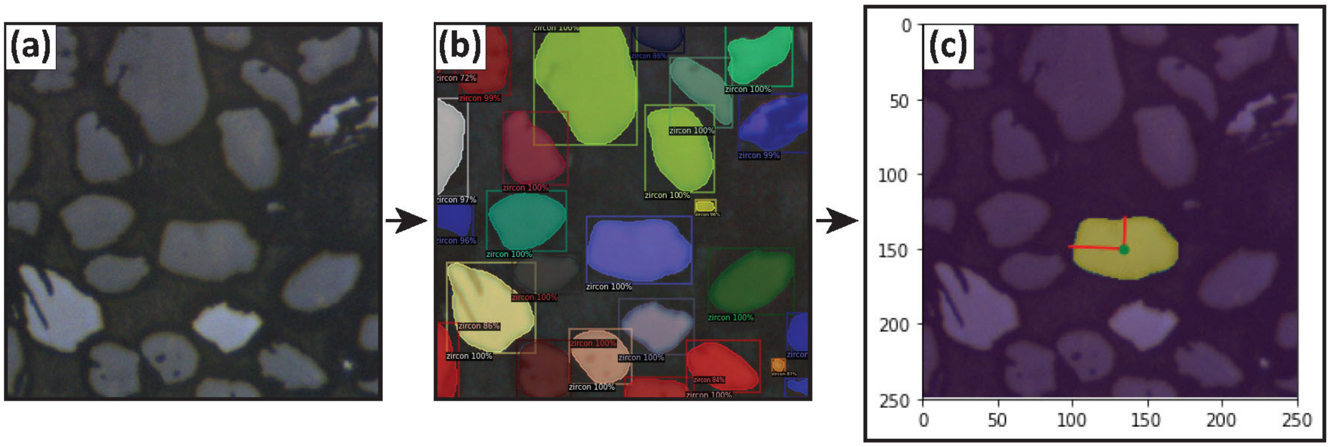
Mask RCNN is an instance segmentation (i.e., Figs. 2a, 2b) convolutional neural network (CNN) model architecture that was developed by Facebook and, when first released, achieved state of the art results when trained and tested on the MS COCO ("Common Objects in Context") instance segmentation baseline dataset (He et al., 2018; Lin et al., 2015). Image inputs in Mask RCNN models initially pass through a backbone neural network, which extracts a feature map containing high level information (e.g., possible objects) from the image (Ren et al., 2016; He et al., 2018). If the model includes a Feature Pyramid Network (FPN), this feature map will be enhanced via convolutional upsampling to improve resolution of small objects (Lin et al., 2016). The feature map is passed to a region proposal network, which generates variably-scaled anchor boxes encompassing the entire map before predicting whether each anchor box contains an actual (i.e., non-background) object and fitting bounding boxes to predicted regions of interest (ROIs) potentially containing objects (Ren et al., 2016; He et al., 2018). Lower-confidence ROIs are dropped at this stage if their degree of overlap with other possible ROIs for the same object is beyond a modifiable non-max suppression (NMS) threshold (Ren et al., 2016). The feature map is subsequently resampled for each remaining ROI by an "ROIAlign" layer to extract ROI-specific feature maps that are aligned pixel-to-pixel with the input image (He et al., 2018). Each ROI-specific feature map is finally passed to both an object detection head layer and a mask head network, which together produce the final model outputs: per-object classifications, fitted bounding boxes, and segmentation masks (He et al., 2018; Ren et al., 2016).

Components and branches of models available in Detectron2 are to a large extent modular and can be swapped with components or branches with different (e.g., newer) designs. We trained and tested Mask RCNN FPN models using two different backbone architectures: ResNet, and Swin (He et al., 2015; Liu et al., 2021). ResNet networks, which are standard for Detectron2 Mask RCNN implementations, use a variable number of residually connected blocks of batch-normalized convolutional layers to produce feature maps (He et al., 2015). ResNets are generally classified by their total number of convolutional layers (i.e., ResNet-101 has 101 layers); we tested Mask-RCNN models with both ResNet-50 and ResNet-101 backbones (Table 1). Whereas ResNet is somewhat dated, the Swin network architecture is near-cutting-edge and achieved state-of-the-art results on several baseline datasets including MS COCO when it was released last year (Liu et al., 2021). Swin models use transformer blocks to extract information (i.e., features) from iteratively upscaled and merged image patches within shifting self-attention windows (Liu et al., 2021). We trained and tested one otherwise-standard Mask RCNN FPN model with an unofficial Detectron2 implementation (Ye et al., 2021) of Swin-T as a backbone (Table 1).

Figure 2. A visualization of the colab_zirc_dims deep-learning-based mineral grain segmentation and measurement process. (a) The original image extracted from an ALC mosaic image, centred on a detrital zircon grain. (b) The results (bounding boxes, probability scores, and masks) of instance segmentation of the original image using a Mask RCNN model (M-R101-C), as displayed by the

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022
© Author(s) 2022. CC BY 4.0 License.

Detectron2 "visualizer" module. (c) The resulting colab_zirc_dims verification image, scaled in µm and displaying the identified central grain mask (yellow), mask centroid (green), and measured long and short grain axes (red).



155 **Table 1. Architectures, pre-training and training parameters, and number of iterations for trained model checkpoints picked for evaluation on the full Leary et al. (accepted) dataset.**

Full evaluated model name	Abbreviated model name	Architecture	Backbone	Pre-training dataset	Training image augmentation	Training iterations[b]
101_COCO_base_6.0k	M-R101-C	Mask RCNN	ResNet-101	MS COCO	Yes	6000
50_COCO_base_6.0k	M-R50-C	Mask RCNN	ResNet-50	MS COCO	Yes	6000
101_from_scratch_8.0k	M-R101-S	Mask RCNN	ResNet-101	None	Yes	8000
50_from_scratch_4.0k	M-R50-S	Mask RCNN	ResNet-50	None	Yes	4000
50_from_scratch_no_augs_4.0k	M-R50-S-NA	Mask RCNN	ResNet-50	None	No	4000
mask_rcnn_swint_7.0k	M-ST-C	Mask RCNN	Swin-T[a]	MS COCO	Yes	7000
centermask2_4.0k	C-V-C	Centermask2	VoVNetV2	MS COCO	Yes	4000

[a] Unofficial Detectron2 implementation (Ye et al., 2021).

[b] Number of two-image training iterations completed for model selected for detailed evaluation on test dataset.

3.2.2 Centermask2

Centermask2 is a Detectron2-specific update to the Centermask instance segmentation CNN model architecture; Centermask is somewhat similar to Mask RCNN but differs in some important aspects (Lee and Park, 2020). In Centermask models, multi-scale feature maps extracted by backbone networks with FPN are passed directly to a Fully Convolutional One-Stage Object Detector (FCOS) model which identifies, classifies, and fits bounding boxes to objects without an intermediate (e.g., in Mask RCNN) anchor box prediction stage (Lee and Park, 2020; Tian et al., 2019). Object-specific masks are then predicted by a mask head which focuses on apparently information-rich pixels (i.e., spatial attention) from input images (Lee and Park, 2020). By reducing the number of steps involved in predicting and segmenting objects, Centermask models are able to out-perform Mask RCNN models on baseline datasets such as MS COCO in accuracy, especially with regard to object bounding box

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

predictions (Lee and Park, 2020). The standard backbone for Centermask models is VoVNetV2, which uses staged feature map downsampling ("one-shot aggregation") with residual connections to predict features somewhat more accurately than ResNet (Lee and Park, 2020; Lee et al., 2019). We trained and tested one Centermask2 model equipped with an FPN VoVNetV2-99 (i.e., with 99 convolutional layers) backbone (Table 1).

3.2.3 Training

Deep learning models can be trained either from scratch with randomly initialized weights (i.e., training-optimizable parameters which determine model behaviour) or from a pre-trained baseline. Most of the weights in well-trained instance segmentation models will be broadly applicable to distinguishing objects (e.g., from 80 classes including "person", "car", and "fork" in MS COCO) from backgrounds, so pre-trained models can be adapted to segment new object classes by retraining only some (e.g., the last backbone convolutional layer) on new data (Lin et al., 2015; Alzubaidi et al., 2021). Such adaptation of pre-trained weights (i.e., "transfer learning") generally allows retraining of models for new applications with smaller datasets, less training, or some combination thereof (Alzubaidi et al., 2021). We re-trained four models from MS-COCO-pre-trained weights (Table 1) and trained three additional Mask-RCNN ResNet-FPN models from scratch (Table 1) to verify that pre-training was appropriate for achieving accurate grain segmentations (Detectron2; Lee and Park, 2020).

Table 2. A summary of the dataset used to train the deep learning models presented in this manuscript for reflected light mineral grain segmentation.

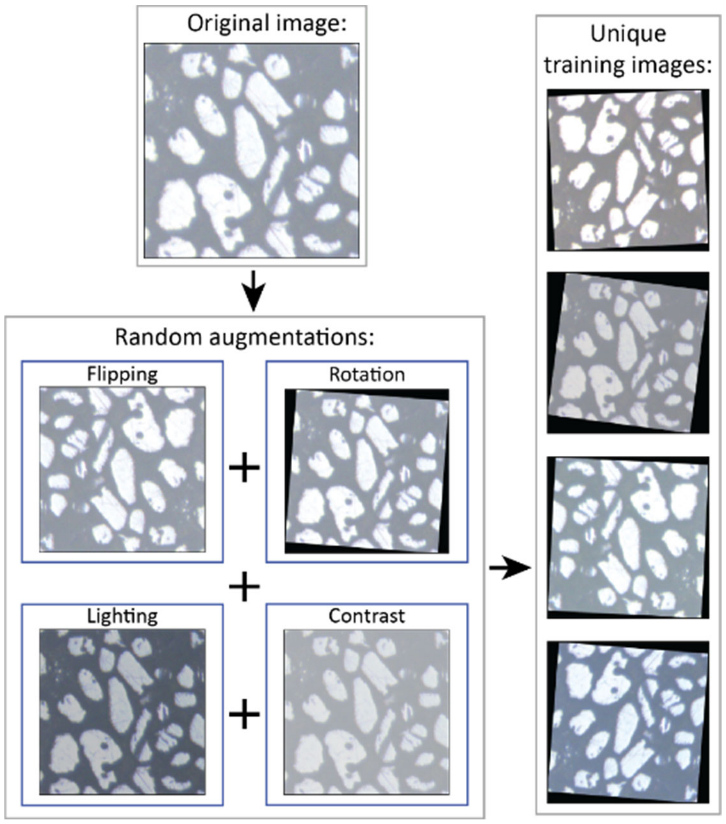
Set	Images	Zircon grains
Training	112[a]	1555
Validation	31[b]	480

[a] 18 from UCSB, 94 from ALC.

[b] 12 from UCSB, 19 from ALC.

We assembled a training dataset (with training and validation sub-sets; Table 2) of reflected light detrital zircon LA-ICP-MS images and manually segmented all mineral grains visible in the images using VIA Image Annotator (Dutta and Zisserman, 2019). ALC images (Table 2) were algorithmically extracted at varying, sample-dependent scales and resolutions (194 by 194 pixels to 398 by 398 pixels) from mosaic images captured during analyses of detrital zircon from the Eagle and Paradox basins, USA; dates and Chromium-derived manual grain measurements resulting from these analyses were published by Leary et al. (2020). UCSB images (Table 2) were captured during analyses of detrital zircon from units in east-central Nevada, USA and added to the dataset as-is, with respective resolutions and scales of 1280 by 1024 pixels and 0.43 µm/pixel. Training and validation images were selected semi-randomly: most were picked at random using the Python 3.8 "Random" module (Van Rossum, 2020) and some additional images were hand-selected in order to ensure that models were exposed to common image artefacts and atypical grain morphologies during training. Although some training and validation images contain likely detrital apatite grains in addition to zircon, we segmented all visible mineral grains into a single "zircon" class to avoid harming our models' generalization abilities in the presence of varying image exposure levels. Our models are consequently likely applicable to segmentation of all reflected-light bright-phase minerals but are unable to distinguish these minerals from one another. Because our dataset is small by deep learning standards, we used Detectron2-based random augmentations (Fig. 3) during training of all but one (control) model (Table 1) to ensure that models did not see the exact same image twice and so improve learning and mitigate potential overfitting.

Figure 3. A graphical summary of the image augmentation method used during model training. Additional uniform (i.e., to 800 pixels for Mask RCNN ResNet models) and random (i.e., to 400 to 800 pixels for Mask RCNN Swin-T and Centermask2) short-edge resizing augmentations were also applied.

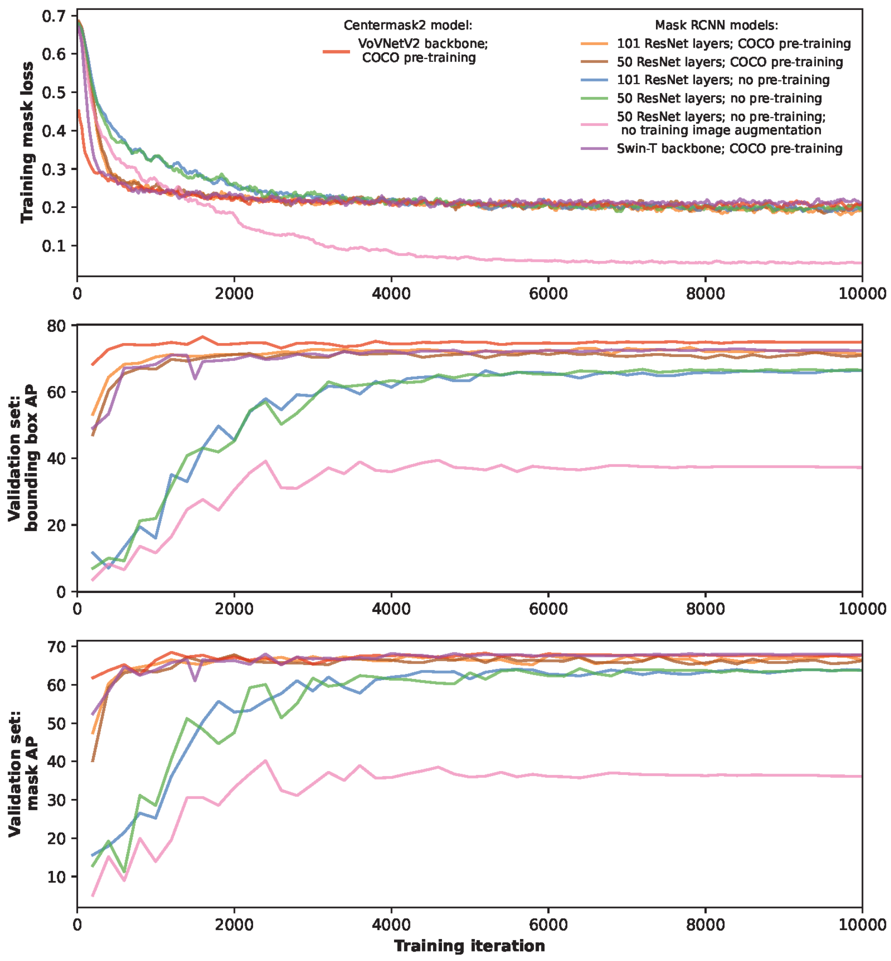


We trained each of our models in Google Colab using Detectron2 for at least 11,000 total two-image iterations with model-dependent learning rate schedules, all of which incorporated a 1,000 iteration warmup period and stepped 50% learning rate reductions at variable (generally 1,000 iteration) intervals. MS COCO-style evaluations of the validation set (Fig. 4) were run every 200 iterations, and checkpoint model weights were saved at ~1000 iteration intervals for further evaluation against outside data. Losses during training were calculated using model-default loss functions (He et al., 2018; Lee and Park, 2020; Detectron2); cross-entropy mask loss is a component of all model loss functions and so can be used for one-to-one inter-model training loss comparison (Fig. 4). We did not implement weight decay during training of any models except Mask RCNN with Swin-T backbone, where we used a starting weight decay parameter of 0.05 with 75% reduction at every stepped learning rate drawdown. Grain boundaries visible in reflected light images should not overlap, so we identified optimally reduced NMS

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

thresholds (0.06 for Mask RCNN and 0.4 for Centermask2) for generating accurate single-grain detections and masks and implemented these thresholds both during training and in colab_zirc_dims processing notebooks. We have followed the example of Nachtergaele and De Grave (2021) and include in the data repository for this manuscript our training dataset and Google Colab notebooks for training new Mask RCNN (ResNet and Swin-T backbone) and Centermask2 models to allow for future model improvements via training parameterization or implementation of new architectures.

Figure 4. Plots showing values of training cross-entropy mask loss and validation set MS-COCO-style AP metrics (mean average precision; calculated from multi-scale, multi-confidence-level intersection over union means) for bounding boxes and masks during model training.



3.3 Dimensional analysis of mineral grains

The initial step in dimensional analysis of grains using colab_zirc_dims is standardized loading of grain images for segmentation such that differently formatted image-datasets can be processed using a single set of algorithms. Shot-centred single images (e.g., from UCSB) can be passed to models for segmentation as-is, but segmentation of grains from mosaic image datasets (e.g., from ALC) is performed on scaled, shot-centred sub-images extracted from mosaics using shot coordinate metadata. Grain-centred images are segmented by a deep learning model, and the resulting segmentations (e.g., Fig. 2b) are passed to an algorithm that attempts to identify and return a "central" mask corresponding to the shot target grain LA-ICP-MS analysis (Fig. 2c). If no mask is found at the actual centre of the image, as may be the case in slightly misaligned images, the algorithm searches radially outwards until either a mask is identified or the central ~10% of the image has been checked. If a central grain is found, its dimensions are analysed using the "regionprops" function from the scikit-image "measure" module (van der Walt et al., 2014), and the resulting measurements and properties are, where applicable, scaled from pixels to µm or µm^2 using a Chromium-metadata-derived scale factor. Successful grain-image processing by the colab_zirc_dims grain segmentation and measurement algorithm will return the following grain-mask properties: *area*, *convex area*, *eccentricity*, *equivalent diameter*, *perimeter*, *major axis length*, *minor axis length*, and *circularity*. Circularity is calculated from scikit-image-derived area and perimeter measurements using Eq. (1); this is a notably simpler and likely less robust calculation than would be required for grain roundness (i.e., Resentini et al., 2018). Calculations for other returned properties can be found in the scikit-image documentation (van der Walt et al., 2014).

Equation 1:

$$Circularity = \frac{4\pi * Area}{Perimeter^2}$$

4 colab_zirc_dims

4.1 The colab_zirc_dims package

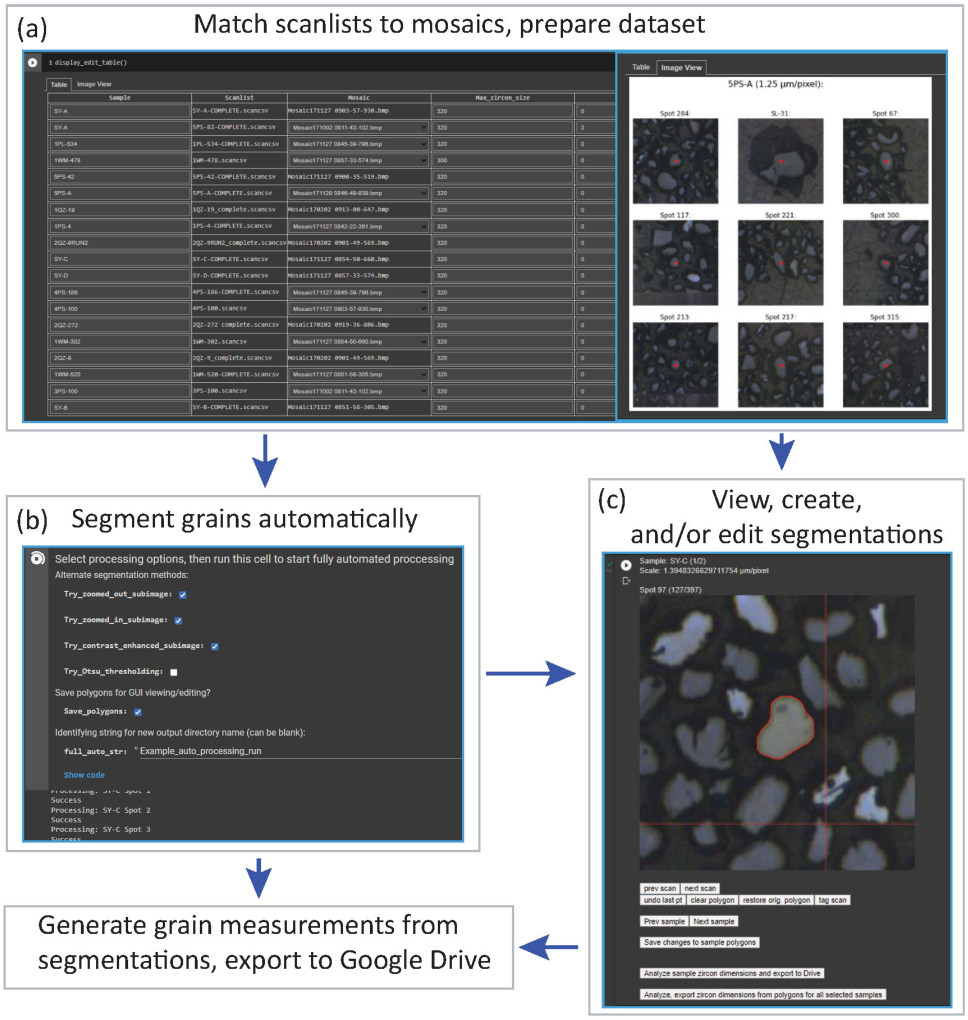
Code for loading and parsing Chromium alignment and shot list files, segmenting and measuring grains using deep learning models, and interacting with Colab notebooks using widgets is contained within the colab_zirc_dims package, which we have made available on the Python Package Index (Python Package Index - PyPI, 2022) for easy installation to local and virtual (e.g., Google Colab) machines. Some colab_zirc_dims modules (e.g., utilities for reading Chromium metadata files and basic segmentation functions) will work in local installations, but the package is only fully functional in Colab virtual runtimes.

4.2 Dataset organization

Before using colab_zirc_dims notebooks to automatically or semi-automatically measure grains, users must upload their dataset (i.e., image and metadata files) as a project folder on Google Drive. Required formats for colab_zirc_dims project folders are simple but necessarily differ slightly between dataset types (e.g., ALC mosaics or UCSB per-shot images), and they are thoroughly documented in the processing notebook for each type of dataset. Once a project folder has been created in a user's Google Drive, they can proceed either directly to Colab-notebook-based processing in the case of per-shot image datasets or to an additional, likewise notebook-based dataset preparation step in the case of mosaic image datasets.

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

255 **Figure 5. A graphical summary of interfaces and workflow options available in colab_zirc_dims processing notebooks. (a) The interactive ALC mosaic dataset preparation (matching mosaic images to .scancsv shot lists, setting sub-image sizes, adjusting misalignment, etc.) interface. (b) The automated segmentation options interface for ALC datasets. (c) The GUI for semi-automated segmentation and editing of model- or human-generated segmentations.**


(a) Match scanlists to mosaics, prepare dataset
Table
Image View
5PS-A (1.25 µm/pixel):
Spot 284
SL-31
Spot 67
Spot 117
Spot 221
Spot 300
Spot 213
Spot 217
Spot 315
(b) Segment grains automatically
Show code
(c) View, create, and/or edit segmentations
prev scan
next scan
undo last pt
clear polygon
restore orig. polygon
tag scan
Prev sample
Next sample
Save changes to sample polygons
Analyze sample zircon dimensions and export to Drive
Analyze, export zircon dimensions from polygons for all selected samples
Generate grain measurements from segmentations, export to Google Drive

```
1 display_edit_table()
```

Sample	Scanlist	Mosaic	Max_zircon_size	
SY-A	SY-A-COMPLETE.scancsv	Mosaic171127 0903-57-930.bmp	320	0
SY-A	5PS-82-COMPLETE.scancsv	Mosaic171002 0811-43-102.bmp	320	3
1PL-534	1PL-534-COMPLETE.scancsv	Mosaic171127 0845-39-796.bmp	320	0
1WM-478	1WM-478.scancsv	Mosaic171127 0857-33-574.bmp	300	0
5PS-42	5PS-42-COMPLETE.scancsv	Mosaic171127 0900-35-519.bmp	320	0
5PS-A	5PS-A-COMPLETE.scancsv	Mosaic171129 0846-40-939.bmp	320	0
1QZ-19	1QZ-19_complete.scancsv	Mosaic170202 0913-00-647.bmp	320	0
1PS-4	1PS-4-COMPLETE.scancsv	Mosaic171127 0842-22-381.bmp	320	0
2QZ-9RUN2	2QZ-9RUN2_complete.scancsv	Mosaic170202 0901-49-569.bmp	320	0
SY-C	SY-C-COMPLETE.scancsv	Mosaic171127 0854-50-660.bmp	320	0
SY-D	SY-D-COMPLETE.scancsv	Mosaic171127 0857-33-574.bmp	320	0
4PS-186	4PS-186-COMPLETE.scancsv	Mosaic171127 0845-39-796.bmp	320	0
4PS-100	4PS-100.scancsv	Mosaic171127 0903-57-930.bmp	320	0
2QZ-272	2QZ-272 complete.scancsv	Mosaic170202 0919-36-806.bmp	320	0
1WM-302	1WM-302.scancsv	Mosaic171127 0854-50-660.bmp	320	0
2QZ-9	2QZ-9_complete.scancsv	Mosaic170202 0901-49-569.bmp	320	0
1WM-520	1WM-520-COMPLETE.scancsv	Mosaic171127 0851-58-305.bmp	320	0
3PS-100	3PS-100.scancsv	Mosaic171002 0811-43-102.bmp	320	0
SY-B	SY-B-COMPLETE.scancsv	Mosaic171127 0851-58-305.bmp	320	0

Select processing options, then run this cell to start fully automated proccessing
Alternate segmentation methods:

Try_zoomed_out_subimage: ✓
Try_zoomed_in_subimage: ✓
Try_contrast_enhanced_subimage: ✓
Try_Otsu_thresholding: ☐

Save polygons for GUI viewing/editing?

Save_polygons: ✓

Identifying string for new output directory name (can be blank):

full_auto_str: " Example_auto_processing_run

```
Success
Processing: SY-C Spot 2
Success
Processing: SY-C Spot 3
```

Sample: SY-C (1/2)
Scale: 1.3948326629711754 µm/pixel

Spot 97 (127/397)

4.3 Colab notebooks

4.3.1 Dataset preparation tools

260 As we note in Sect 3.3, segmentation and measurement of grains in mosaic image datasets requires extraction of shot-specific sub-images from larger mosaics using shot locations in corresponding .scancsv shot metadata files. Information on which mosaic file in a project folder matches which .scancsv file must consequently be provided by users for processing. Because deep learning models struggle to identify and segment grains when they cannot see all grain boundaries (e.g., if sub-images are smaller than grains), sub-image extraction also requires a user-provided, mosaic-specific sub-image size for accurate

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022
© Author(s) 2022. CC BY 4.0 License.

segmentations and measurements. Colab_zirc_dims processing notebooks read the aforementioned information from "mosaic_info" .csv files stored in project folders. Though these "mosaic_info" files can be created and uploaded to project folders manually, they can also be generated quickly and easily using the "*Mosaic_Match*" colab_zirc_dims notebook (Fig. 5a) that we provide. The "*Mosaic_Match*" notebook implements code that automatically finds matches between shot lists and mosaics in a project folder and allows to users to generate, modify, and export "mosaic_info" tables (Fig 5a). Users can view sample shot locations and sub-images using a "Display" function (Fig. 5a), thus allowing interactive mis-alignment correction, adjustment of sub-image sizes, and, in cases where multiple possible mosaics could match a single .scancsv file, identification and selection of the correct mosaic from a dynamically populated dropdown menu. After exporting a "mosaic_info" .csv file, users can proceed to fully or semi-automated segmentation and measurement of their dataset (Figs. 5b, 5c).

4.3.2 Fully automated segmentation and measurement

We provide Colab notebooks for automated and semi-automated processing of both mosaic image ("*Mosaic_zircon_process*") and per-shot image ("*Single_shot_image_zircon_process*") datasets; these notebooks are respectively currently set up to fully support processing of ALC and UCSB datasets but will likely work with datasets from other facilities sans-modification. The per-shot image notebook additionally supports loading and processing of any grain-centred reflected light grain images without Chromium scaling metadata, in which case users can provide custom per-sample scaling information in a .csv file or use a default scale of 1 μm/pixel.

Deep learning segmentation model weights are selected by users from a dropdown menu and downloaded to Colab runtimes from an Amazon Web Services repository (provided by us) prior to model initialization and processing. After weight file download and model initialization, users can select options for automated processing (Fig. 5b). These options include whether to attempt segmentation with various alternate methods (e.g., zooming out slightly or increasing image contrast before reapplying the model, or, as a last resort, using Otsu thresholding) if segmentation is initially unsuccessful, and whether to save polygons approximating model-produced masks for viewing or modification in the colab_zirc_dims graphical user interface (GUI) (Fig 5b). During automated processing, grain dimensional analyses (Sect. 3.3) in per-sample .csv files along with verification mask image .png files (e.g., Fig. 2c) are saved and exported to the user's Google Drive.

4.3.3 Colab-based GUI for semi-automated segmentation and measurement

Because our deep learning models cannot fully match human accuracy when segmenting grains, we provide a simple, Colab-based GUI (Fig. 5c) extended from code in the Tensorflow Object Detection API (TensorFlow Developers, 2022) that allows users to view, modify, and save polygon-based grain segmentation masks. These polygon-masks are can either be loaded from a previous automated or GUI processing session or generated on-the-fly on a per-sample basis. After viewing or re-segmenting part or all of a dataset, users can send their grain segmentations for measurement and export (Sect. 4.3.2); grain dimension exports from the GUI will include additional tags indicating whether each grain was segmented by a human or by a deep learning model.

https://doi.org/10.5194/gchron-2022-12
Preprint. Discussion started: 5 May 2022

5 Accuracy evaluations

We assessed the accuracy of our segmentation models by automatically processing the Leary et al. (accepted) mosaic image dataset using all trained segmentation models with the colab_zirc_dims "*Mosaic_Process*" notebook and comparing resulting automated measurements with the manual (measured with the Chromium "Measure" tool) per-grain axial measurements from the same dataset. We picked "best" model checkpoints (Table 1) where models achieved apparent local maxima in validation accuracies (i.e., Fig. 4) and local minima in various measurement error metrics (e.g., failure rate and absolute long axis error; Table 3).

Table 3. Evaluation of error versus human measurements for automated measurements produced by deep learning models on the Leary et al. (accepted) dataset, with segmentation via Otsu's thresholding method as a baseline. The best results on each metric are shown in bold type.

Model / method	**n[b]**	**Failure rate (%)**	**Average[a] error (µm)**		**Average[a] absolute error (µm)**		**Average[a] error (%)**		**Average[a] absolute error (%)**		**≥ 20% absolute error rate (%)**		**Grain extent underestimate rate[c] (%)**	**Average[a] segmentation time per spot[d] (s)**
			Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	Long axis	Short axis		
M-R101-C	5003	**0.02**	-1.2	-1.0	6.1	**4.4**	-0.45	-0.93	7.96	**8.83**	8.20	9.85	10.11	0.119 [e]
M-R50-C	4992	0.24	-2.0	-1.7	6.2	4.5	-1.47	-2.38	8.04	8.98	8.21	10.94	11.52	**0.072** [e]
M-R101-S	4931	1.46	-2.2	-1.2	7.7	5.7	-1.31	-0.87	9.75	11.45	11.60	15.43	15.37	0.107[e]
M-R50-S	4988	0.32	-3.6	-2.3	7.6	5.5	-3.13	-3.45	9.46	10.91	11.65	15.92	17.46	0.087 [e]
M-R50-S-NA	4749	5.10	-2.6	**-0.2**	13.1	8.6	-1.01	1.82	17.18	18.16	23.23	23.08	21.75	0.108 [e]
M-ST-C	4993	0.22	-2.7	-2.4	**6.0**	4.5	-2.79	-4.03	**7.71**	8.96	8.45	11.40	13.18	0.129 [e]
C-V-C	4998	0.12	**-0.7**	-0.7	**6.0**	**4.4**	**0.12**	**-0.27**	7.85	9.00	**7.40**	**9.76**	**9.80**	0.114 [f]
Otsu thresholding	5004	0.00	-3.4	-4.0	11.9	9.7	-3.27	-5.50	16.02	21.69	18.11	26.84	27.34	0.020

[a] Arithmetic mean.

[b] Number of scan-images out of a total of 5004 where a "central" grain mask could be identified with confidence ≥ 70% for Mask RCNN models or ≥ 50% for Centermask2 (C-V-C) model.

[c] Proportion of grains where new measurements differ by ≤ -20% along long and/or short axes from published (Leary et al., accepted) measurements.

[d] Time for model/method to segment (or fail to segment) an image and return a measurable mask. Actual per-spot processing times will be higher due to additional automated mask measurement and image saving time.

[e] Measured in Colab Notebook with NVIDIA T4 GPU.

[f] Measured in Colab Notebook with NVIDIA P100 GPU.

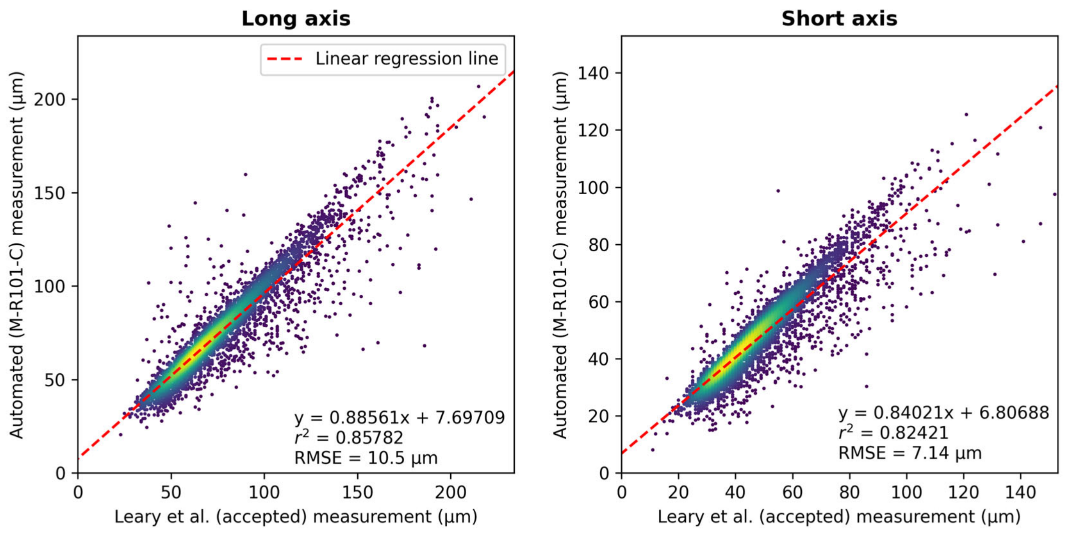
5.1 Impact of model architecture and training parameters

Some of our results are unsurprising: models trained from scratch with training image augmentation fail to match pre-trained models in validation set segmentation (Fig. 4) and full dataset (Leary et al., accepted) measurement accuracies (Table 3) despite converging on the same training loss values (Fig. 4). Our model trained from scratch without training image augmentation (M-R50-S-NA) additionally underperforms pure Otsu's method thresholding in segmentation measurement accuracy in several metrics with a concurrently high rate of failure to recognize grains in images (Table 3), likely due to poor generalization ability. Pre-training and training image augmentation are essential to training models that accurately and consistently segment mineral grains.

Our three best-performing models on the full dataset are M-R101-C, C-V-C, and M-ST-C, and we find that M-R50-C produces faster but somewhat less accurate measurements (Table 3). C-V-C achieves the highest measurement accuracy in most absolute error metrics as well as impressive, though likely not particularly robust (e.g., M-R50-S-NA; Table 3), < 0.5% average long and short axis errors. In combination with its high validation bounding box AP (Fig. 4), this suggests that Centermask2's anchor-free detection algorithm allows C-V-C to predict grain extents prior to segmentation with greater accuracy than Mask RCNN models (Table 3). M-ST-C notably fails to outperform our other models by most metrics (Table 3) despite its state-of-the-art Swin-T backbone, and we attribute this to our small training dataset; the basic Swin architecture tends to perform best when given large amounts of training data (Lee et al., 2021). We expect that all our models could improve with more training data and that Centermaks2 and Mask RCNN models with Swin backbones could significantly outperform Mask RCNN ResNet models given a much larger training dataset or one that requires models to distinguish between grains of different minerals. Although it is slightly less accurate overall than C-V-C, M-R101-C has the lowest grain identification failure rate (Table 3) and we observe that it can accurately segment grains from artefactitious (e.g., very blurry) images where other models fail. This suggests that model M-R101-C has the best generalization ability of our trained models, and, though users are encouraged to experiment with different models, we have made M-R101-C the default colab_zirc_dims segmentation model to maximize potential applicability to images from LA-ICP-MS facilities not represented in our training or test datasets.

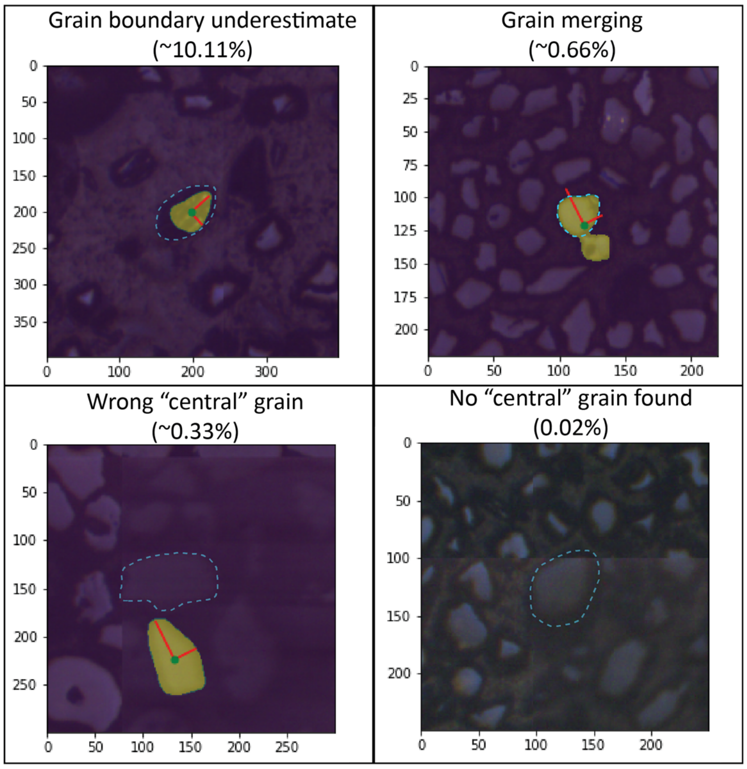
5.2 Machine error

Figure 6. Automated (M-R101-C) and manual (Leary et al., accepted) measurement plots for long and short grain axes with linear regression lines plotted and gaussian KDE density shown via heatmap.



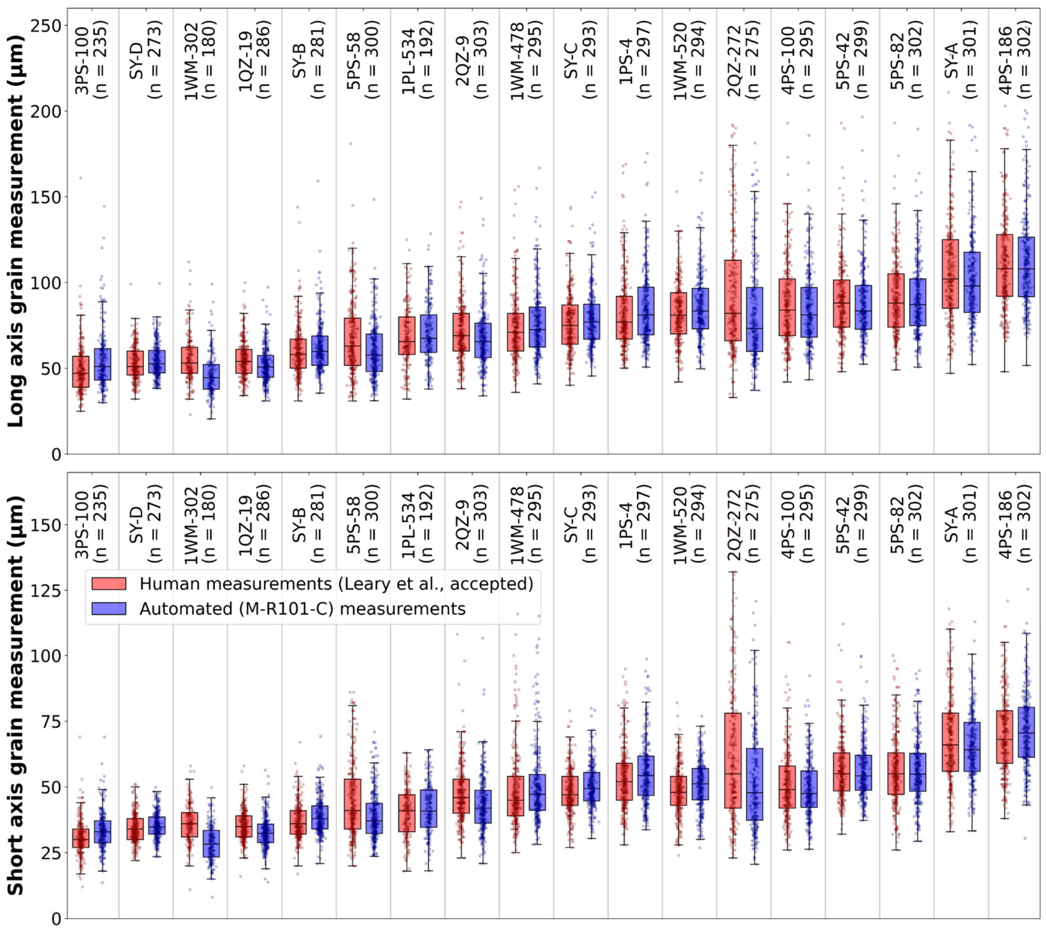
Per-grain automated (M-R101-C) measurements for the full Leary et al. (accepted) dataset generally hew close to ground-truth measurements but skew slightly negative (Fig. 6). The apparent dominant cause of this negative skew is under-segmentation of grains that are incompletely exposed at the surface of epoxy mounts but whose full grain areas are interpretable by humans from "shadows" visible in the (mostly) reflected light images (Fig. 7). We did not train our models to interpret beyond clearly visible grain boundaries and they consequently fail to reproduce human measurements for these grains, but models could likely be trained to do so without diminished accuracy on "normal" grains given a larger, interpretively segmented training dataset. Positive measurement errors are relatively rare (Fig. 6) but are probably mainly attributable to segmentation masks that merge different grains (Fig. 7). Failure to identify the correct "central grain" in images (Fig. 7) is likewise rare but may cause positive, negative, or negligible measurement error depending on the respective sizes of the target and mistakenly identified grains. Cases where no grain could be identified are exceedingly rare (Table 3, Fig. 7) and do not contribute directly to measurement error but, like all identified errors, necessitate manual re-segmentation of grains for production of accurate measurements.

Figure 7. Examples of automated (M-R101-C) segmentation mask error modes with estimated occurrence rates, with axes scaled in µm and correct grain segmentations outlined in light blue. Rates for "grain boundary underestimate" and "no central grain found" errors are estimated from analysis of the entire Leary et al. (accepted) dataset (i.e., Table 3), and "grain merging" and "wrong central grain" rates are based on manual error identification and classification in an n = 301 sample of the full dataset (i.e., Table 4).



We find that automated processing using colab_zirc_dims and our default model (M-R101-C) can approximately reproduce aggregate long and short grain axis length distributions for most samples in the Leary et al. (accepted) mosaic image and measurement dataset (Fig. 8). Systemic negative errors along both grain axes are concentrated within four samples (1WM-302, 5PS-58, 2QZ-9, and 2QZ-272; Fig. 8). We found that grains in these samples were consistently underexposed above mount surfaces and that "grain extent underestimate" (Table 3; Fig. 7) segmentation errors were as a result sufficiently common to negatively impact sample axis length distributions. Based on these results, we believe that fully automated measurement using colab_zirc_dims is a viable method for rapid approximation of grain size distributions in optimal (e.g., with well-exposed grains) samples. Due to low but significant segmentation error rates (Fig. 7), manual segmentation verification and correction (i.e., semi-automated measurement) is, however, necessary for production of publication-quality grain measurement datasets. Assuming time requirements of 26 minutes total to automatically generate segmentation masks, one second per grain to manually check masks, and 20 seconds to correct each mis-segmentation, and, conservatively (Fig. 7), that 15% of grains must be re-segmented via GUI, we estimate that it would take about six hours to semi-automatically collect zircon grain measurements for the full (n = 5,004) Leary et al. (accepted) dataset using colab_zirc_dims.

Figure 8. A sample-by-sample boxplot comparison of human (Leary et al., accepted) and automated (M-R101-C) measurements along long and short grain axes. Boxes extend from Q1 to Q3, and whiskers extend from Q1 - 1.5 * (Q3 - Q1) to Q3 + 1.5 * (Q3 + Q1); sample medians are indicated by black horizontal lines.



5.3 Human error

Automated measurement error metrics (e.g., Table 3) likely encompass some error that would be present even if grains were manually segmented, due to differential interpretations of grain areas between researchers, especially in samples with under-exposed grains. In an n = 301, randomly picked sample-proportionate grain subsample from the Leary et al. (accepted) dataset, we find that our default automated segmentation model (M-R101-C) achieves similar axial measurement absolute error metrics to the first author (M.S.) of this manuscript (Table 4). Though free of clear mis-segmentations (e.g., grain merging; Fig. 7) and mostly free of interpretive grain extent underestimates, the first author's measurements skew higher than dataset measurements (Table 4). Apparent over-interpretations of grain extents by the first author likely reflect different image display conditions (e.g., higher zoom and different contrast) during manual re-segmentation versus those present during collection of dataset measurements. Various features of colab_zirc_dims, namely automated segmentation of most grains and uniform image display conditions during manual segmentation of other grains, may enhance grain measurement dataset reproducibility in addition to collection speed.

Table 4. A comparison between human (first author) and automated (model M-R101-C) measurement errors in an n = 301, sample-proportionate random subsample of the Leary et al. (accepted) image-measurement dataset. The best results for each metric are shown in bold type.

Segmented by	n	Average[a] error (µm)		Average[a] absolute error (µm)		Average[a] error (%)		Average[a] absolute error (%)		RMSE (%)		≥ 20% absolute error rate (%)		Grain extent underestimate rate[b] (%)
		Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	Long axis	Short axis	
Human[c]	301	4.0	3.9	**6.6**	4.8	6.19	8.70	8.83	10.65	**12.55**	14.06	**8.64**	14.29	**1.66**
M-R101-C	301	**-1.8**	**-1.2**	6.9	**4.6**	**-1.13**	**-1.37**	**8.56**	**9.10**	13.33	**12.87**	9.30	**9.30**	10.30

[a] Arithmetic mean.

[b] Proportion of grains where new measurements differ by ≤ -20% along long and/or short axes from published (Leary et al., accepted) measurements.

[c] Manual grain segmentation by the first author using the colab_zirc_dims GUI; measurements derived from the resulting masks.

6 Future developments

The colab_zirc_dims package and Google Colab notebooks make it significantly faster and easier to augment an appropriate LA-ICP-MS dataset with grain measurements. We will continue to maintain and update colab_zirc_dims, and in the future hope to test and, if necessary, modify our code to extend full support to datasets from facilities beyond ALC and UCSB, possibly including those using targeting software other than Chromium. Though individual researchers are our intended userbase for colab_zirc_dims, we believe that deep learning models also hold great potential utility for LA-ICP-MS facilities. Such facilities are well-resourced to create larger training datasets than ours and could implement trained models in a variety of applications including provision of per-spot grain measurements as a standard data product, fully automated spot picking, and possibly automated phase identification.

7 Conclusions

We have trained a set of deep learning models to segment detrital zircon grain instances from reflected light images and developed code that uses the models to rapidly extract per-grain dimensional measurements from images saved during LA-ICP-MS analyses at facilities using Chromium targeting software. We present this code as the colab_zirc_dims Python package, and we implement it in a collection of interactive, ready-to-run Google Colab notebooks. These notebooks are highly accessible and allow users to automatically or semi-automatically process datasets utilizing freely available high-end GPUs.

Based on evaluations of our trained models against manual grain measurements from an in-press, large-n detrital zircon dataset, we conclude that transfer learning and training image augmentation are essential to training deep learning models that can accurately segment and identify mineral grains given a relatively small training dataset. We find that Centermask2-VoVNetV2 and Mask RCNN models with ResNet-101 and Swin-T backbones achieve the greatest measurement accuracies on the evaluated dataset and expect that greater-still accuracies could be reached with more training data. All our trained models are available to colab_zirc_dims users for application to their datasets.

The colab_zirc_dims deep-learning-based automated measurement algorithm approaches human measurement accuracy on a sample-by-sample basis and can be used to rapidly approximate grain size distributions for samples with well-exposed zircon grains, without any human involvement. Our semi-automated segmentation workflow allows researchers to create manually reviewed and corrected grain size measurements for large-n datasets in under a day. We believe that colab_zirc_dims makes it drastically easier to augment applicable LA-ICP-MS datasets with grain measurements, and we hope that allowing more researchers to do so will expand our understanding of the relationships between zircon dimensions and age in varied environments. We also hope to extend full colab_zirc_dims support to datasets that do not currently work with its processing notebooks in the future and encourage users to share samples of such datasets with the first author.

Code availability

The colab_zirc_dims source code, small example datasets, and links to pre-formatted template project folders and the latest versions of colab_zirc_dims Google Colab notebooks are available at the colab_zirc_dims GitHub page (Sitar, 2022): https://github.com/MCSitar/colab_zirc_dims. Additional code for reproducing error evaluations and figures presented in this manuscript using new or included automatically generated measurements is included in the supplementary data repository (Sitar and Leary, 2022): https://doi.org/10.5281/zenodo.6412303.

Data availability

The full Leary et al. (accepted) dataset of images and measurements that we used for model evaluation, our training dataset, model training scripts with copies of our trained model weights, and full measurement and evaluation data supporting the results presented in our manuscript can be found in the supplementary data repository (Sitar and Leary, 2022): https://doi.org/10.5281/zenodo.6412303.

Author contribution

M.S. wrote the first draft of the manuscript and both authors contributed to subsequent drafts. M.S. segmented the training dataset, trained the models, developed the code, and evaluated model-derived measurements. R. L. provided contextualized image and measurement datasets for model training and evaluation and feedback for improvement of the code and processing notebooks.

Competing interests

The authors declare that they have no conflict of interest.

Acknowledgements

We would like to thank Kurt Sundell for insights into imaging systems at the Arizona LaserChron Center. Michael Sitar is additionally grateful to John Singleton for the leeway to finish this project. Previously unpublished UCSB training image data were collected in collaboration with Alaina Rosenthal-Guillot, with assistance from Andrew Kylander-Clark. Work by Michael Sitar was completed concurrently with work supported by grants from the National Science Foundation (award # 2115719) and USGS EDMAP Program (award # G21AC10493).

References

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, J. Big Data, 8, 53, https://doi.org/10.1186/s40537-021-00444-8, 2021.

Python Package Index - PyPI: https://pypi.org/, last access: 13 April 2022.

Augustsson, C., Voigt, T., Bernhart, K., Kreißler, M., Gaupp, R., Gärtner, A., Hofmann, M., and Linnemann, U.: Zircon size-age sorting and source-area effect: The German Triassic Buntsandstein Group, Sediment. Geol., 375, 218–231, https://doi.org/10.1016/j.sedgeo.2017.11.004, 2018.

Bradski, G.: The OpenCV Library, Dr Dobbs J. Softw. Tools, 2000.

Bukharev, A., Budennyy, S., Lokhanova, O., Belozerov, B., and Zhukovskaya, E.: The Task of Instance Segmentation of Mineral Grains in Digital Images of Rock Samples (Thin Sections), in: 2018 International Conference on Artificial Intelligence

Applications and Innovations (IC-AIAI), 2018 International Conference on Artificial Intelligence Applications and Innovations (IC-AIAI), 18–23, https://doi.org/10.1109/IC-AIAI.2018.8674449, 2018.

Cantine, M. D., Setera, J. B., Vantongeren, J. A., Mwinde, C., and Bergmann, K. D.: Grain size and transport biases in an Ediacaran detrital zircon record, J. Sediment. Res., 91, 913–928, https://doi.org/10.2110/jsr.2020.153, 2021.

TensorFlow Developers.: TensorFlow, Zenodo, https://doi.org/10.5281/zenodo.5949169, 2022.

Dutta, A. and Zisserman, A.: The VIA Annotation Software for Images, Audio and Video, Proc. 27th ACM Int. Conf. Multimed., 2276–2279, https://doi.org/10.1145/3343031.3350535, 2019.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E.: Array programming with NumPy, Nature, 585, 357–362, https://doi.org/10.1038/s41586-020-2649-2, 2020.

He, K., Zhang, X., Ren, S., and Sun, J.: Deep Residual Learning for Image Recognition, ArXiv151203385 Cs, 2015.

He, K., Gkioxari, G., Dollár, P., and Girshick, R.: Mask R-CNN, ArXiv170306870 Cs, 2018.

Ibañez-Mejia, M., Pullen, A., Pepper, M., Urbani, F., Ghoshal, G., and Ibañez-Mejia, J. C.: Use and abuse of detrital zircon U-Pb geochronology—A case from the Río Orinoco delta, eastern Venezuela, Geology, 46, 1019–1022, https://doi.org/10.1130/G45596.1, 2018.

J. D. Hunter: Matplotlib: A 2D Graphics Environment, Comput. Sci. Eng., 9, 90–95, https://doi.org/10.1109/MCSE.2007.55, 2007.

Jiang, F., Li, N., and Zhou, L.: Grain segmentation of sandstone images based on convolutional neural networks and weighted fuzzy clustering, IET Image Process., 14, 3499–3507, https://doi.org/10.1049/iet-ipr.2019.1761, 2020.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C.: Jupyter Notebooks – a publishing format for reproducible computational workflows, in: Positioning and Power in Academic Publishing: Players, Agents and Agendas, 87–90, 2016.

Latif, G., Bouchard, K., Maitre, J., Back, A., and Bédard, L. P.: Deep-Learning-Based Automatic Mineral Grain Segmentation and Recognition, Minerals, 12, 455, https://doi.org/10.3390/min12040455, 2022.

Lawrence, R. L., Cox, R., Mapes, R. W., and Coleman, D. S.: Hydrodynamic fractionation of zircon age populations, GSA Bull., 123, 295–305, https://doi.org/10.1130/B30151.1, 2011.

Leary, R., Smith, M. E., and Umhoefer, P.: Mixed eolian-longshore sediment transport in the Late Paleozoic Arizona Pedregosa basin, USA: a case study in grain-size analysis of detrital zircon datasets, J. Sediment. Res., accepted for publication.

Leary, R. J., Smith, M. E., and Umhoefer, P.: Grain-Size Control on Detrital Zircon Cycloprovenance in the Late Paleozoic Paradox and Eagle Basins, USA, J. Geophys. Res. Solid Earth, 125, e2019JB019226, https://doi.org/10.1029/2019JB019226, 2020.

Lee, S. H., Lee, S., and Song, B. C.: Vision Transformer for Small-Size Datasets, ArXiv211213492 Cs, 2021.

Lee, Y. and Park, J.: CenterMask : Real-Time Anchor-Free Instance Segmentation, ArXiv191106667 Cs, 2020.

Lee, Y., Hwang, J., Lee, S., Bae, Y., and Park, J.: An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection, ArXiv190409730 Cs, 2019.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P.: Microsoft COCO: Common Objects in Context, ArXiv14050312 Cs, 2015.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S.: Feature Pyramid Networks for Object Detection, https://doi.org/10.48550/arXiv.1612.03144, 2016.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B.: Swin Transformer: Hierarchical Vision Transformer using Shifted Windows, https://doi.org/10.48550/ARXIV.2103.14030, 2021.

McKinney, W.: Data Structures for Statistical Computing in Python, in: Proceedings of the 9th Python in Science Conference, 56–61, https://doi.org/10.25080/Majora-92bf1922-00a, 2010.

Muhlbauer, J. G., Fedo, C. M., and Farmer, G. L.: Influence of textural parameters on detrital-zircon age spectra with application to provenance and paleogeography during the Ediacaran–Terreneuvian of southwestern Laurentia, GSA Bull., 129, 1585–1601, https://doi.org/10.1130/B31611.1, 2017.

Nachtergaele, S. and De Grave, J.: AI-Track-tive: open-source software for automated recognition and counting of surface semi-tracks using computer vision (artificial intelligence), Geochronology, 3, 383–394, https://doi.org/10.5194/gchron-3-383-2021, 2021.

Otsu, N.: A Threshold Selection Method from Gray-Level Histograms, IEEE Trans. Syst. Man Cybern., 9, 62–66, https://doi.org/10.1109/TSMC.1979.4310076, 1979.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, ArXiv191201703 Cs Stat, 2019.

Pérez, F. and Granger, B. E.: IPython: a System for Interactive Scientific Computing, Comput. Sci. Eng., 9, 21–29, https://doi.org/10.1109/MCSE.2007.53, 2007.

Ren, S., He, K., Girshick, R., and Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, ArXiv150601497 Cs, 2016.

Resentini, A., AndÒ, S., and Garzanti, E.: Quantifying Roundness of Detrital Minerals By Image Analysis: Sediment Transport, Shape Effects, and Provenance Implications, J. Sediment. Res., 88, 276–289, https://doi.org/10.2110/jsr.2018.12, 2018.

Scharf, T., Kirkland, C. L., Daggitt, M. L., Barham, M., and Puzyrev, V.: AnalyZr: A Python application for zircon grain image segmentation and shape analysis, Comput. Geosci., 162, 105057, https://doi.org/10.1016/j.cageo.2022.105057, 2022.

Sitar, M. C.: MCSitar/colab_zirc_dims: colab_zirc_dims v1.0.8, Zenodo, https://doi.org/10.5281/ZENODO.6410745, 2022.

Sitar, M. C. and Leary, R. J.: colab_zirc_dims: full results, datasets, and replication code repository, https://doi.org/10.5281/zenodo.6412304, 2022.

Soloy, A., Turki, I., Fournier, M., Costa, S., Peuziat, B., and Lecoq, N.: A Deep Learning-Based Method for Quantifying and Mapping the Grain Size on Pebble Beaches, Remote Sens., 12, 3659, https://doi.org/10.3390/rs12213659, 2020.

Sundell, K., Gehrels, G. E., Quinn, D. P., Pecha, M., Giesler, D., Pepper, M., George, S., and White, A.: AGECALCML: AN OPEN-SOURCE MATLAB-BASED DATA REDUCTION PLATFORM FOR LA-ICP-MS GEOCHRONOLOGY AND GEOCHEMISTRY DATA FROM THE ARIZONA LASERCHRON CENTER, GSA 2020 Connects Online, 358944, https://doi.org/10.1130/abs/2020AM-358944, 2020.

Teledyne Photon Machines: Chromium 2.4, Teledyne Photon Machines, 2020.

Tian, Z., Shen, C., Chen, H., and He, T.: FCOS: Fully Convolutional One-Stage Object Detection, ArXiv190401355 Cs, 2019.

Umesh, P.: Image Processing in Python, CSI Commun., 23, 2012.

Van Rossum, G.: The Python Library Reference, release 3.8.2, Python Software Foundation, 2020.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and contributors, the scikit-image: scikit-image: Image processing in Python, PeerJ, 2, e453, https://doi.org/10.7717/peerj.453, 2014.

Detectron2: https://github.com/facebookresearch/detectron2.

Ye, H., Yang, Y., and L3str4nge: SwinT_detectron2: v1.2, Zenodo, https://doi.org/10.5281/ZENODO.6468976, 2021.