Longitudinal Analysis of Midgut
LAM v1.0

User Manual

Arto I. Viitanen

Hietakangas Laboratory

University of Helsinki

Table of Contents

1 Description

Longitudinal Analysis of Midgut (LAM) is a tool for reducing the dimensionality of microscopy image–obtained data, and for subsequent quantification of variables and feature counts while preserving regional context. LAM's intended use is to analyze whole *Drosophila melanogaster* midguts or their sub-regions for phenotypical variation due to differing nutrition, altered genetics, etc. Key functionality is to provide statistical and comparative analysis of variables along the whole length of the midgut for multiple sample groups. Additionally, LAM has algorithms for the estimation of feature-to-feature nearest distances and for the detection of cell clusters, both of which also retain the regional context. The analysis is performed after image processing and feature detection. Consequently, LAM requires coordinate data of the features as input.

The analysis is based on approximation of the samples from end to end through the creation of vectors onto which features can be projected by minimal distance estimation. The vectors are divided into user-defined number of bins that can be readily compared between samples and sample groups. The feature-specific point of projection along the vector is linked to all relevant input data through feature identifiers, subsequently allowing quantification of cells and their differing variables along the length of the sample. All individual samples' quantification data is joined with their sample groups to find the groups' general characteristics, which can then be compared group-by-group in a statistical analysis. The joining of the data is based on anchoring of the vectors to each other via a user-provided input coordinate that corresponds to a distinguishable, real biological segment of the samples, e.g. R3–region. In practice, the anchoring means that all samples are inserted into a data matrix where all anchoring points are located at the same index position, with variable lengths of the vector at either side. This is necessary due to the inherent proportional variation within the midgut that leads to a compounding error when moving further along the bins. Consequently, insertion into the matrix guarantees that the samples are in 'focus' at the anchoring site and that the correct biological regions are compared against each other.

LAM has been tested on larval and adult midguts, but usage can most likely be extended to other tissues where longitudinal quantification could be applicable. The analysis method is for the most part restricted to planar geometries, and consequently most functionalities of LAM do not offer resolution on the Z-axis. The limitation to XY-coordinates should be sufficient for most research questions regarding the midgut, as it is pseudostratified and has only minute layering of cells.

Notification:

While the positional input data that LAM accepts can indicate any sort of feature that is to be counted and analyzed, **for the purpose of the user manual any individual observations will be referred to as cells.** LAM also accepts multiple data folders for each sample, and while each of these folders does not necessarily contain data from one microscopy channel, **these separate data will be referred to as channels for simplicity.** Microscopy channels will be referred to as such.

2 Installation of Dependencies

LAM is developed in Python 3.7 environment. The distribution includes requirements.txt and environment.yml that both contain names and version numbers of LAM dependencies. The environment.yml is used to create an Anaconda virtual environment in anaconda prompt:

- conda env create -n <yourenvname> -f <path\to\environment.yml>
- conda activate <yourenvname>

while replacing the text between "<>" changed to desired environment name and the path to the file. Alternatively, the two required packages can be installed directly to Anaconda base environment (see box below).

When creating a python virtual environment, the needed packages can be directly installed by using *pip* with the requirements.txt in the terminal:

1. python -m venv <yourenvname>
   - Linux:
     source <yourenvname>/bin/activate
   - Windows:
     <yourenvname>\Scripts\activate.bat
2. pip install -r <path-to-requirements.txt>

Installation to Anaconda base environment

1. Install Anaconda3 distribution (https://www.anaconda.com/distribution/)

2. Add Shapely-package:

   Windows:
   Get Shapely wheel from https://www.lfd.uci.edu/~gohlke/pythonlibs/#shapely
   Then write following command(s) in Anaconda prompt:
   (0.) pip install wheel (should be included in Anaconda)
   1. pip install <path-to-the-downloaded-whl-file>

   OS X & Linux:
   Open Anaconda prompt and write following command:
   pip install shapely

3. Add pycg3d-package:
   Open Anaconda Prompt and write command:
   pip install pycg3d

Dependencies: matplotlib (3.1.1), numpy (1.16.5), pandas (0.25.1), pathlib2 (2.3.5), pycg3d (0.0.1), scipy (1.3.1), seaborn (0.9.0), shapely (1.6.4), scikit-image (0.15.0), statsmodels (0.9.0)

3 Usage

LAM is used by executing run.py in terminal or an integrated development environment (IDE). The execution by default opens the graphical user interface (GUI) which allows the use of all functionalities of LAM. The workflow is as follows:

0. (Feature detection from microscopy images) (user)
1. Organizing files for input (user)
2. Creation of sample-specific vectors (automated or user)
3. Gathering of data and projection onto vector
4. Calculation of cell numbers and additional data
5. Finding nearest cells and clusters
6. Calculation of statistics
7. Plotting

The analysis requires a very specific folder and file hierarchy, all of which need to be named respecting a convention in order to provide information for the program (3.2.2). The sample vectors that are used to reduce the dimensionality of data can be either made by the user or alternatively by LAM (see 3.4). The rest of LAM's functionalities are performed automatically in accordance to user-defined settings.

3.1. Image pre-processing

To assure the best functioning of LAM, certain pre-processing steps are required before feature detection and subsequent data feeding to LAM. Of these, the masking and orienting of the microscopy images is paramount. The vector creation also functions best with straightened midguts; inordinate curvature may cause premature end of vectorization. The samples are to be masked so that no excess features are present, i.e. the image contains only intensities from the midgut. The samples should also be oriented so that either end of the midgut has the smallest X-coordinate value. The orientation has to be the same between samples of an experiment, e.g. all samples have the anterior end at the lowest X-axis values. The orientation is necessary in order to create proper vectors and to provide comparable quantification. LAM also includes a script to rotate feature coordinate csv's around the origin, if necessary (Companions/rotator.py).

Originally, LAM was designed to be an extension to Imaris–analyses and consequently exported data files from Imaris' feature detection and other tools are in a format that is immediately usable by LAM. However, LAM only requires that the data is numerical, in csv–file format, and with certain column labels.

3.2. Input

The main functionality of LAM, i.e. the locational quantification of cells, requires only X- and Y-coordinates of cells within a csv-file with all variables on columns and each cell on separate row, i.e. in wide-format. For finding nearest cells and clusters, the Z-coordinate of the cells is also required. Any additional data that is to be analyzed must be defined in the settings and can either be located within the same csv-file with the positional data or separately. All samples are not expected to have data on all channels; LAM only adds the data to analysis if it is found.

The vectors are created based on the positional data of the 'vector channel', defined by the given value in *'Channel'*-setting (*'vectChannel'* in settings.py). The creation begins from cells with the lowest X-coordinates, and consequently all samples are expected to be oriented the same in the coordinate system. In case of wrongly oriented samples, a coordinate system-rotation script (rotator.py) can be found in the 'Companions'-folder of LAM.

3.2.1. Sample anchoring (MP)

On some experiments the size proportions of different regions may alter, e.g. when comparing starved and fully fed midguts. In these cases, results that are more accurate can be obtained by dividing the image/data into multiple analyses. A typical way to do this is to run separate analyses for R1-2, R3, and R4-5. Alternatively, a user-defined coordinate (MP = measurement point) at a distinguishable point can be used to anchor the individual samples for comparison. For example, MP at R2/3-border of each sample causes them to be lined at a specific bin, with each sample having variable numbers of bins on either side (Fig. 1). The proportional variation however likely leads to a compounding error as distance from the MP grows. When MP is not used, the samples are lined at bin 0 and compared bin-by-bin. The MP-input is given similarly to channel data, i.e. as a separate directory that contains position.csv for a single coordinate, the MP.

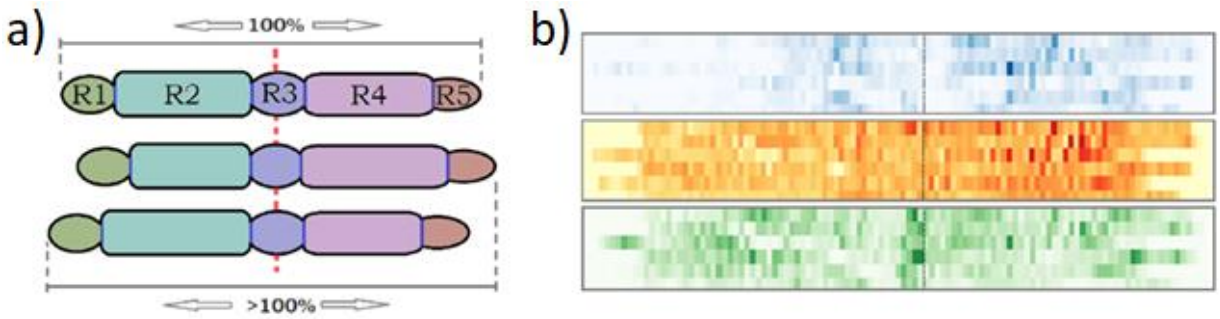

Fig. 1. Sample anchoring using Measurement Points. *Due to possible proportional variation between sample groups, LAM has the option to anchor samples based on user-given coordinates. The anchoring coordinate is called a measurement point, or MP. When using MPs, the samples are lined so that every MP is located at same bin index. As a result, the samples are in 'focus' at the given positions and comparison between samples and sample groups gives biologically relevant data.* ***a)*** *MP marked by the red dashed line. Each sample has the same number of bins, i.e. 100% of user-defined bins, but after sample anchoring the total length of the dataset can be greater than 100% due to proportional variation.* ***b)*** *Heat maps of real data, with each row representing one whole-midgut sample. MP marked by the black dashed line. Shared characteristics between the samples are most clearly seen near the MP-bin.*

3.2.2. File organization and naming

LAM expects to find all input data in a certain hierarchy of directories and with specific filenames that contain necessary information (Fig. 2). As LAM scans the directory for files and folders without user-defined samples, sample groups, or channels, the recommendation is not to have any unrelated files or folders within the input data-directory, as this can break the analysis. LAM automatically gathers the names of the samples, groups, and channel-names from the paths of the files.

The full naming convention for the paths in the analysis is as follows:

<group>_<descriptor>_<sample>_<channel>_<xyz>,

where group denotes the sample group, descriptor and sample are used to identify individual samples (e.g. date of imaging and sample name), and channel is an identification for a data folder. The "xyz" at the end can be any string of text and is not used by LAM; it is present for convenience, as e.g. Imaris-exported data includes "_Statistics" at the end.

Within the analysis directory, the files and folders for the analysis must be organized in the manner shown by Fig. 2. All samples should be located at the root of the analysis directory in separate folders named as <group>_<descriptor>_<sample>, e.g. Control_09-06-2019_sample1. Within these sample folders, each sample should have a separate channel folder for each channel it has data for, named with the before mentioned name extended with <channel>_xyz, e.g. Control_09-06-2019_sample1_DAPI_Stats. Each data file that relates to the specific channel and sample must be found within these channel folders.

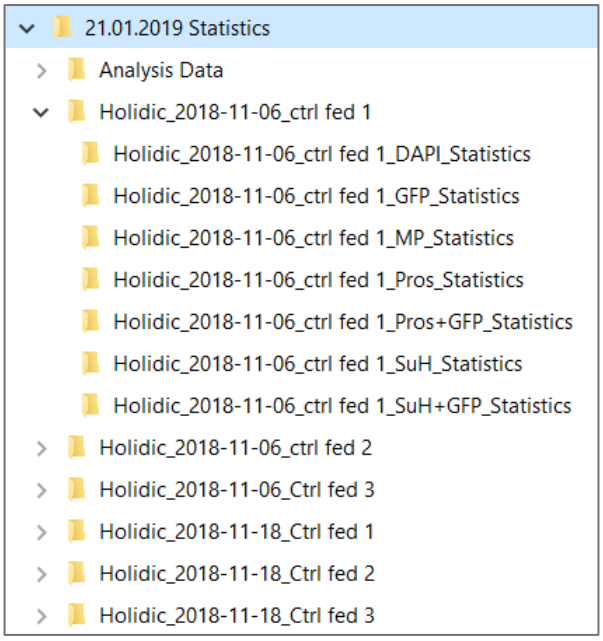

Fig. 2. Analysis directory hierarchy. *All sample folders are to be located in the root of the analysis directory. Within the sample folders, each channel must have its own folder that contains all channel-specific data files. Additionally, the folders have to be named consistently and according to a convention.*

Underscore '_' is used as a delimiter for information, and is consequently reserved for LAM. Any use of underscore by the user will likely interfere with the analysis. Naming should be restricted to letters [A-Z], numbers [0-9], and plus and minus signs [+ -].

3.2.3. Data file column labels

Within the data files, specific column headers are required. In the Position.csv, the coordinates of cells should be marked with column labels 'Position X', 'Position Y', and 'Position Z'. Additionally, each cell should have its own unique identification in a column labeled as 'ID'. This cell ID should be unique within the sample and shared between all data files to correctly associate data. If the data files contain for example metadata-headers, the *'header_row'*-setting can be changed to point at the correct row with column labels, with row numbering starting from zero. LAM expects all user input data files to have the same header row index number.

All data specific to a sample must be located in its sample-specific folder at the root of the analysis directory. Similarly, all data related to one channel must be in the specific channel-folder of the respective sample folder.

Position X,Position Y,Position Z,Unit,Category,Collection,Time,ID,		
4735.07,390.268,0.7,um,Spot,Position,1,0,		
4741.71,397.542,0.7,um,Spot,Position,1,1,		
4726.84,399.752,0.7,um,Spot,Position,1,2,		
4705.06,405.75,0.7,um,Spot,Position,1,3,		
4698.85,411.068,0.7,um,Spot,Position,1,4,		

Fig. 3. Example of Positions.csv column labels and data in Imaris–format. *LAM requires the positional coordinates in columns labeled with 'Position X', 'Position Y', and 'Position Z'. Additionally, the file must contain 'ID'–column where each cell of the channel has a unique identifier that is used for merging data. Of these column labels, the unit, category, and collection are not used by LAM.*

3.2.4. Additional data

LAM considers all data in addition to the positional coordinates and ID as 'additional data', e.g. area, volume, and intensities. These data files must be defined within the *'AddData'*–dictionary variable in *settings.py*, or alternatively in the *'Other'*-window if using the GUI. Additional data can be in the Position.csv, but its location must be given in the aforementioned variable.

In the case of additional data, its key (in bold below) in the 'AddData' dictionary defines the required column label for the data:

AddData = {"**Area**": ["Area.csv", "Area, $\u03BCm^2$"]}.

The values given to the key, in order, are the name of the data-containing file, and the unit of the variable. Any special characters should be given in Unicode and between '$'-signs. The unit is only used for plots.

When inputting a data type that has multiple values, e.g. intensities from each microscopy channel, LAM expects the column labels to have an identifier separated by underscore, e.g. "Intensity Median_Ch=1" or "Intensity Median_ch2". The identifiers do not need to be defined for LAM as they are only used to separate the data from each other, but they can be defined in order to change them to a form that is more informative, e.g. from Ch=4 to DAPI. The replacement can be done with *'replace file ID'*–option in GUI's Other–window (*replaceID* and *channelID* in settings.py).

It should be noted that internally LAM uses '-' as separator instead of '_' when handling identifiers of additional data with multiple column labels, such as intensities from various microscopy channels.

3.3. Primary functionalities

LAM has several primary functionalities:

- Process
- Count
- Distance
- Plots
- Stats

The '**process**' setting creates the vectors, while '**Count**' projects cells from all found channels and counts their numbers. The output files from both are required for plots and statistical calculations, which are performed via the '**Plots**' and '**Stats**' settings. The '**Distance**' setting is used for the calculation of distances

to nearest cells and for finding clusters of cells. In the GUI, clicking any of these primary settings will enable or disable input for the related settings.

NOTE: using *'Count'* clears the *'Analysis Data'*–folder that holds many LAM–created data files. Move any earlier data you may want to keep.

3.3.1. Default values

The default values for the GUI (and the analysis) are stored in settings.py and can be altered at will. Any changes to the settings within the GUI will revert to the default values when the program is run again.

3.4. Vector creation

The vector creation of LAM is limited to planar geometries by the *Shapely* package, i.e. only XY-coordinates are used. The limitation should not cause variation in the counting of cells as the midgut is pseudostratified and there is little Z-coordinate difference between cells, assuming the sample is flat when imaged.

The vectors for the samples can be created in two ways: by a running median or by skeletonization. Both methods have their own benefits and drawbacks (3.4.1). The whole analysis can be performed in one run; however, the best and most accurate results can typically be obtained by subjecting the samples to multiple rounds of vector creation and then passing only the well-fitting vectors for further analysis.

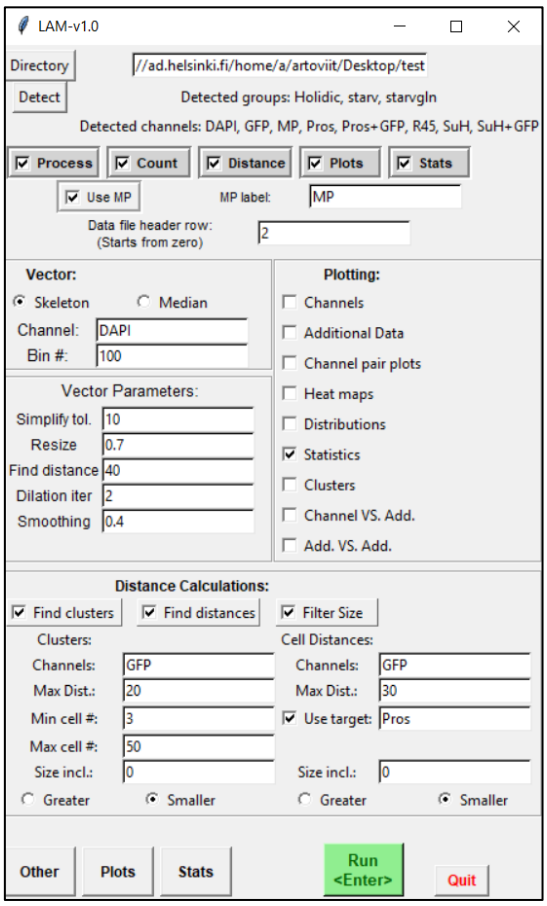

Fig. 4. LAM graphical user interface. *The five primary functionalities of LAM (dark boxes) control which settings are enabled. Use of 'Count' requires that either 'Process' has been run or user has provided self-made vector.csv's. The three other functionalities require that 'Count' has been performed.*

A typical workflow starts by selecting only the 'Process'-setting to create vectors for all of the samples. After each round of vectorization, the quality of each sample's vector can be verified from the vector plots located at *'./Analysis Data/Samples'*, and when deemed fit, the respective LAM-created sample folders containing the vector.csv-files can be collected to be used later. Next, vector creation can be performed with other settings, again collecting acceptable vectors. Any unfit vectors can also be directly modified by changing any offending coordinates or by adding new ones to the vector.csv. Alternatively, the user can generate the vector themselves and place it in vector.csv (3.4.2).

Once all vectors are created, the collected sample folders can be transferred back to *'./Analysis Data/Samples'*, overwriting any non-usable data.

3.4.1. Vector types

Both vector creation types share two settings: the vector channel and simplify tolerance (Fig. 5). The vector channel designates the data channel that is used for creating the vector. Best approximation of the sample is achieved by using a channel that is densely populated by cells. Consequently, the recommendation is to use data files containing nuclear label data or similarly dense features.

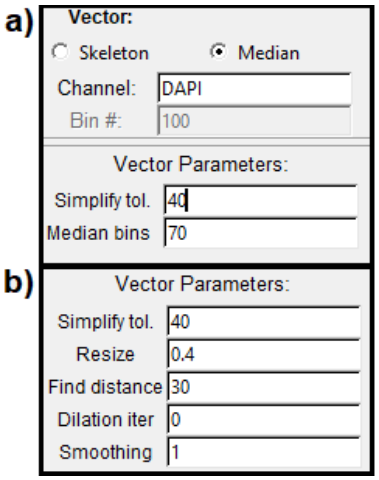

Fig. 5. Vector creation settings. a) In the upper part are the general settings for vector creation and in the lower part the median vector-related settings. b) All settings related to skeleton vector creation.

Different methods of vector creation can lead to different kinds of artifacts in the approximation. The artifacts can often be removed with vector simplification, i.e. the straightening of the vector by moving its points. The *'Simplify tol.'*–setting gives the maximal distance of the movement.

Median vector

The median vector–creation performs ideally with fully straightened samples. However, steep curvature in the sample or any backwards movement on X-axis will cause problems for this vector type. The vector creation is simple; it is based on calculating the middle Y-axis coordinate between the top-most and bottom-most cells in each 'median bin'. The number of the bins is defined by the *'Median bins'*-setting in vector parameters (Fig. 5a). Equidistant points between the lowest and highest X-coordinates of the data set's cells define the edges of the median bins. The left-most bin is assumed to be the first bin, meaning that the vectors are created from left to right (smallest X-value to largest), and the samples should be oriented to account for this.

To assure the best quality of sample approximation for the vector, the number of median bins should be set high. The median bin number is ultimately limited by the density of cells on the channel that is used for vector creation; too many bins might lead to bins with no cells, in which case the previous bin's value is copied. The empty bins can consequently lead to stair-like increases in the median, but this can be remedied with vector simplification, set by *'Simplify tol.'*.

Skeleton vector

The skeleton-method offers better approximation for highly curved samples, especially if the sample has any back and forth movement on the X-axis. The downside is that some settings may sample-specifically cause branching in the skeleton, resulting in a broken vector. Recommendation is to ascertain the quality from the vector plots; there is no guaranteed warning when a vector is faulty. When using skeleton vectors, LAM produces an additional plot of the original binary image and the resulting skeleton to each sample's folder. This skeleton plot can be useful in determining how the creation settings should be changed to get a fit vector.

The creation settings for the skeleton vector are *'resize', 'find distance', 'dilation iter.',* and *'smoothing'* (Fig. 5b). Modifying these settings is encouraged; optimal settings are highly dependent on the shape of each sample. Most important of these are *'resize'* and *'smoothing'* (Fig. 6). Both of these settings remove

roughness from the edges of the sample and consequently reduce branching of the skeleton. However, in samples where loops in the midgut bring the intestinal walls close together, these settings can cause the different segments to merge. In these cases, it is necessary to have a smaller size decrease and less smoothing. When any resizing would lead to merging of midgut segments, increasing the iteration number of binary dilation (*'dilation iter.')* may help in making the midgut uniform in the transformed binary image and consequently reduce skeleton branching.

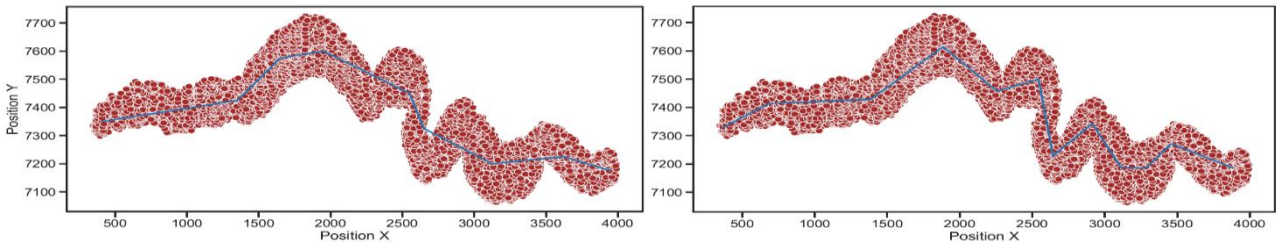

Fig. 6. Effects of resizing and smoothing on skeleton vector creation. *On the left values of resizing and smoothing are 0.4 and 1, respectively. The settings cause segments of the latter half of midgut to merge, subsequently leading to a mistaken approximation and false peaks in cell counts. On the right, using a lesser reduction in size (0.7) and only a little smoothing (0.3) leads to a much-improved vector.*

The transformation of the skeleton into a vector is performed by following pixels beginning from the smallest X-coordinate. From the start point, the creation algorithm adds new points to the vector by scoring pixels based on distance and the direction of the existing vector. Due to the scoring method, the vector creation can at times follow the wrong pixels if there is any branching in a steep curve. In these cases, it is necessary to reduce the branching with for example increased smoothing in order to get a fit vector.

3.4.2. User-generated vectors

When opting to use self-created vectors, it should be noted that the first coordinate in the file is the beginning of the first bin (i.e. bin zero) of the vector. Consequently, all vectors must be given in the same orientation, e.g. from the anterior to the posterior end of the sample. The number of the coordinates that define the vector is arbitrary, and the binning of the vector for projection of cells is done relative to the length of the vector. A convenient method for self-drawn vectors is to use one of Fiji ImageJ's line tools on the microscopy image and then saving as XY coordinates. The user-generated vectors have to be placed similarly to LAM-created vectors, in the vector.csv's located at respective sample-folder in *'./Analysis Data/Samples'.*

3.5. Cell projection and counting

After vector creation, all channels found in the sample folders located at the root of the analysis directory are projected onto the vector if the *'Count'* setting is selected. For counting, the most critical value is *'Bin #'* located within the *vector*-frame in the GUI (*projBins* in settings.py). The bin number determines the length of LAM–created data arrays and consequently affects plotting and statistical analysis. All vectors are divided into the same number of bins. The length of the bins is uniform within a sample but can be variable between samples due to length differences of the vectors. Consequently, the binning functions well between sample groups that have similar proportions of different parts of the midgut. If proportional

variance exists, the samples should be cropped so that the binning corresponds to the biological proportions.

If analyzing samples that have been cropped into multiple parts, e.g. R1-2, R3, and R4-5, 'Count' can be run once with arbitrary number of bins to collect the length of created vectors into *'Data Files/length.csv'*. From these lengths, the needed fraction of the total bin number can be determined for each cropped section of the sample in order to preserve a more standardized biological length of each bin.

When anchoring the samples at some specific point along the midgut, e.g. R3-region, the *'use MP'*-option can be selected and input given to *'MP label'* to collect the coordinates of the point for each sample. The coordinate must be located in each sample's folder in a channel folder named after the *MP label*. When '*use MP'* is not selected, the samples are anchored bin to bin.

The projection is done by minimal distance estimation and consequently the cells are counted as belonging to the bin sector that is nearest to them. Data on the projection can be found within csv's named by respective channels at *'./Analysis Data/Samples/'*. The new data columns created by the projection are *'VectPoint'*, *'NormDist'*, and *'DistBin'*, which are the XY-coordinates of the point of projection along the vector, the normalized distance (0–1) along the vector, and the projection's bin number, respectively.

3.6. Distance calculations

The two functionalities controlled by the *'Distance'* setting, i.e. the computation of nearest cells and the detection of clusters, both function based on a similar mechanism. For each cell, first its nearest neighbors are found, distances to them are calculated, and if the distances are in acceptable limits they are passed forward. Only the smallest distance is collected when computing nearest cells, but when analyzing clusters, the nearby cells are grouped to form 'seeds' that are later merged based on shared cell ID's in order to find the final clusters.

Both of the functionalities are controlled individually, but they share similar settings (Fig. 7). Both can be performed on multiple channels within the same analysis, set by *'Channels'*. The *'Max Dist.'* defines how distant cells are considered to be neighbors. If the data contains 'Volume' label, the cells can also be filtered by their size, including either smaller or larger cells than the value set in *'Size incl.'*. In case of cell distance calculation, the distance can be set to be calculated against a specific target channel, defined by *'Use target'*. With this setting, the user can for example find nearest Pros-positive cells for each GFP-positive cell. When finding clusters, the user can define cell number limits for what is considered a cluster by modifying the *'Min cell #'* and *'Max cell #'*–settings.

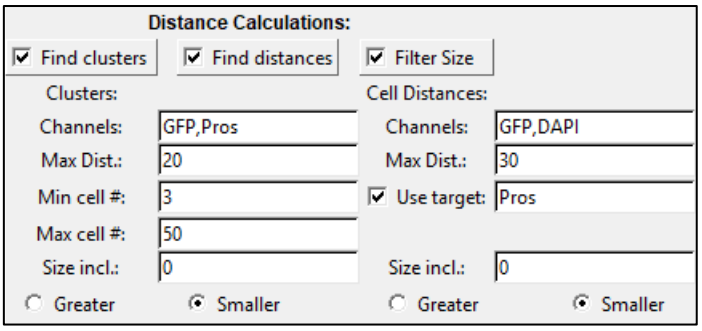

Fig. 7. The settings for distance calculations. *The distance calculation performs two separate functionalities, i.e. the detection of clusters and finding of nearest cells.*

Both functionalities create new data columns to the channel csv's in the *'./Analysis Data/Samples/'*–folders. The 'Find distances' creates three columns, labeled '*Nearest_XYZ'*, *'Nearest_Dist'*, and *'Nearest_ID'*, which are the coordinates of the nearest cell, the distance to it, and the ID of the nearest cell, respectively. The 'Find clusters' creates only one column labeled 'ClusterID', which is a unique identifier that is shared between members of same cluster.

3.7. Statistics

LAM computes two different kinds of statistics, i.e. total statistics and versus statistics. The total statistics create comparisons of sample group data regarding total cell numbers on each channel and averages of additional data in each bin of sample groups. The versus statistics are bin-to-bin comparisons of channel data between the control group and a test group. The versus statistics are channel-specifically calculated for both cell counts and additional data, e.g. for GFP cell counts and for the areas of the GFP cells. Additionally, the statistical analyses are performed for any LAM-calculated cluster and cell-to-cell distance data. For statistical plots to be created, both *'Stats'* and *'Plots'*–primary settings must be selected. The statistical data can be found in *'./Analysis Data/Statistics'*-folder.

The versus statistics are calculated with Mann-Whitney-Wilcoxon U-test with corrections for continuity and multiple testing. The total statistics are done similarly, but without multiple test correction. The projection bin comparing versus statistics can be performed either bin-by-bin, or with a sliding window. On channels with low count numbers, increasing the size of the window leads to greater number of non-zero observations per test, consequently increasing its power at the cost of resolution.

3.8. Plotting

Plots are drawn by selecting the *'Plots'*–setting and subsequently selecting all wanted plots from *'Plotting'*. All plotting excluding statistical plots can be performed in a separate run from the other functionalities of LAM; the plots are created from the data files that are found in the analysis directory. To obtain statistical plots, the *'Stats'*–setting has to be also selected.

The LAM-included plotting options are:

Channels – Bin-by-bin box plots of counts for each channel, with bins of the total, anchored matrix on X-axis and cell counts on Y-axis. The whiskers of the boxplots extend to 1.5 times interquartile range and scatters mark data points beyond this.

Additional Data – Bin-by-bin line plots of additional data, including LAM-created cell-to-cell distances. The line is formed from each sample's averages at the respective bin marked by X-axis. The band around the line indicates the standard deviation of the averages.

Channel Pair plots – A grid of channel count data against channel count data, with channel count distribution on the diagonal axis. Each scatter marker on the grid represents the average cell counts of the samples in one bin on the respective channels. The bands around the regression lines indicate standard deviation.

Heatmaps – Heat maps of bin-by-bin cell count averages of sample groups (*'All Channels Heatmaps'*) and cell counts of samples on all channels (*'All Samples Channel Heatmaps'*).

Distributions – Probability density distributions of bin values of channel counts and additional data, including LAM-created cell-to-cell distances. Output to *'Plots/Distributions'.*

Statistics – Two types of plots: total and versus plots. Total statistics creates violin plots with each sample group for every channel and additional data type. The versus statistics are plotted as bin-by-bin box plots of two sample groups, with P-value significance marked either with stars or negative log2 line plot, and with a possibility for coloring of significant bins. Additionally, the 'observations'-setting in the Plots-window can be selected to plot individual observations on top of the box plot. The Y-limit of the negative log2 line can be changed in the GUI's Plots-window or by changing *'ylim'* in settings.py. The stars from one to three indicate significances of 0.05, 0.01, and 0.001, respectively. Color fill of significant regions is based on the same significances but begins from the P-value defined by *'alpha'* (GUI's Stats window). The whiskers of the boxplots extend to 1.5 times interquartile range and scatters mark data points beyond this. Output to *'Plots/Stat Plots'.*

Clusters – Creates positional scatter plots and boxplots of clustered cells, and bin-by-bin heat maps of average numbers of clustered cells for each sample group. On the positional scatter plots, the tan-colored markers are all cells on the channel, and each found cluster has an individual randomized color. Requires data from *'Distances / Find clusters'.* Positional plot output to *'Plots/Clusters'.*

Channel VS. Add. – A bivariate density estimate of the channel counts and additional data, using averages of each bin of a sample group. Plots created for all possible combinations and may consequently create many plots depending on settings and data. The plotting combinations can be restricted by inputting only wanted data variables in GUI's Plots–window under the *'versus plots'*-section (*'vs_chans'* and *'vs_adds'* in settings.py), e.g. to DAPI and Intensity Mean-Ch=3. The plots represent density estimation of one sample groups values of two variables. The marginal plots are distributions of the variable on the opposing axis line. Output to *'Plots/Chan VS AddData'.*

Add. VS. Add. – As in *'Channel VS. Add.*' but with additional data type against another type.

4 Output Files

LAM creates myriad of output files depending on used settings. All created files can be found in *'./Analysis Data'* and its subfolders; the root of the analysis directory is read-only, with the exception of log files.

Analysis-related data is always stored as csv's, but the type of plot-files can be changed within the *'Plots'*-settings in the GUI or by *'saveformat'* in settings.py.

Collections of the used channels, samples, and sample groups can be found at the root of the *'Analysis Data'*–folder. The folder also contains the following subfolders for storing various data:

Data Files – For storing collections of data from samples. As a baseline, the files have either samples or sample groups in the columns and the projection bins in order as rows. The exception to this are 'Total Counts.csv' with the total cell numbers for each channel on each row, the 'MPs.csv' with the bin number of each sample's MP on the row, and 'Length.csv' with the length of each samples vector on the row.

The rows of files with the 'All_'-prefix contain cell counts for a bin of each sample. The data with 'Norm_'-prefix is the same data as with 'All_', but with MP anchoring. The rows of 'Avg_'–prefixed files contain the average values of the variable for all cells that fall in to the bin. Similarly, the rows of files with 'ChanAvg_'–prefix contain the average channel cell counts of the sample groups for each bin. The 'Clusters' and 'ClNorm' contain the numbers of clustered cells without and with anchoring, respectively.

Plots – Contains all plots created by LAM, with the exception of vector-related plots that are in 'Samples'-subfolder.

Samples – Stores LAM–created and –edited sample-specific data. Within each sample's folder can be found the vector.csv and channel-specific csv's. The channel csv's contain all collected input data for the sample, in addition to LAM–created columns. The created columns are:

- Projection – *'VectPoint'*, *'NormDist'*, and *'DistBin'* are the XY-coordinates of the point of projection along the vector, the normalized distance (0–1) along the vector, and the projection's bin number, respectively.
- Clustering – *'ClusterID'* indicates the given identification number to the cluster that the respective cell belongs to.
- Distance – *'Nearest_XYZ'*, *'Nearest_Dist'*, and *'Nearest_ID'* are the coordinates of the nearest cell, the distance to it, and its ID, respectively.

The root of the *'Samples'*–folder also contains the plots of each sample's vector, and within the subfolders can be found the skeleton plots if using the respective vector creation.

Statistics – Contains all files with statistical data. Each file contains the data of bin-to-bin or windowed tests of the control group against another group. The test variable is indicated after the '='–sign in the file name, i.e. *"Stats_<test groups> = <channel> (<additional data>)".*

Column labels:
- U Score – The MWW U–score of the compared populations
- Corr. <Greater/Lesser> – Corrected P-value that control group is <Greater/Lesser>
- P <Greater/Lesser> – Non-corrected P-value that control group is <Greater/Lesser>
- Labels with 'Two-sided' indicate the probability that there is a significance in either way

Additionally, the Statistics–folder contains the 'Total Count Stats.csv' with the two-way statistics of each channel for the test groups.

5 Definitions

The following definitions are named after the settings in the GUI. The corresponding variables in settings.py are marked with brackets.

5.1. Top Frame

GUI	*Settings.py*	Description
Count	*process_counts*	Projection, anchoring, and counting of data on all channels
Data file header row	*header_row*	The expected row of column labels in user given data files
Distance	*process_dists*	Calculation of cell-to-cell distances and detection of clusters
Directory	*workdir*	Path to analysis directory where sample data is located
MP label	*MPname*	The name of the csv data files containing anchoring point locations
Plots	*Create_Plots*	Creation of any/all plots
Process	*process_samples*	Creation of vectors for samples
Stats	*statistics*	Calculation of statistics
Use MP	*useMP*	Whether to use user given anchoring points

5.2. Vector Frame

GUI	*Settings.py*	Description
Bin #	*projBins*	Number of bins onto which data is projected on all vectors
Channel	*vectChannel*	The channel on which vector creation is based on
Skeleton / Median	*SkeletonVector*	The type of vector to be created

5.3. Vector Parameters Frame

GUI	*Settings.py*	Description
Dilation iter.	*BDiter*	Iterations of binary dilation (2x2) on resized and image-transformed position coordinates

Find distance	*find_dist*	The maximum coordinate distance of next pixel (coordinate) of vector when creating from skeleton
Median bins	*medianBins*	The number of medians that are calculated for the sample when creating a median vector
Resize	*SkeletonResize*	Binary image resizing factor when creating skeleton vector
Simplify tol.	*simplifyTol*	Tolerance of coordinate adjustment for vector simplification
Smoothing	*SigmaGauss*	Sigma for Gaussian smoothing of binary image

5.4. Plotting Frame

GUI	*Settings.py*	Description
Add. VS. Add.	*Create_AddVSAdd_Plots*	Plot additional data against additional data
Additional Data	*Create_AddData_Plots*	Plot additional data averages per bin
Channel pair plots	*Create_Channel_PairPlots*	Create plot matrix of all channels against all channels
Channel VS. Add	*Create_ChanVSAdd_Plots*	Plot channels versus additional data
Channels	*Create_Channel_Plots*	Plot box plots of channel data
Clusters	*Create_Cluster_Plots*	Create box plots and heat maps of clustered cells
Distributions	*Create_Distribution_Plots*	Plot distribution densities for all channel and additional data
Heat maps	*Create_Heatmaps*	Plot sample and sample group heatmaps of channel counts
Statistics	*Create_Statistics_Plots*	Plot statistical box plots of all data

5.5. Distance Calculations Frame

Compared to cell-to-cell distances, the variables specific to clustering algorithm have the prefix 'Cl_' in settings.py.

GUI	*Settings.py*	Description
Channels	*Distance_Channels, Cluster_Channels*	The channels to which distances are calculated
Filter size	*N/A*	Whether to filter cells based on volume
Find clusters	*Find_Clusters*	Whether to perform clustering of cells

Find distances	*Find_Distances*	Whether to find cell-to-cell distances
Greater / Smaller	*incl_type, Cl_incl_type*	The direction of size filtering, includes cells of either smaller or greater size
Max cell #	*Cl_max*	Maximum cell number that is considered to be a cluster
Max Dist.	*maxDist, Cl_maxDist*	Maximum distance between cells to be considered neighbors. In clustering Max Dist is a hard distance limit for inclusion into a cluster.
Min cell #	*Cl_min*	Minimum cell number of a cluster
Size incl.	*Vol_inclusion, Cl_Vol_inclusion*	The volume limit of filtering
Use target	*use_target + target_chan*	Calculate cell-to-cell distances from one channel to the target channel

5.6. Other Window

GUI	*Settings.py*	Description
Column label / csv-file / Unit	*AddData*	Gathering of additional data. The label of the column where data is located / the name of the file where data is / unit of the data for plotting
File descriptor / Change to	*channelID*	Replace these ID's found in file names / the proper names of the ID's
Replace file ID	*replaceID*	Whether to replace channel ID's found in file names

5.7. Plots Window

General settings:

GUI	*Settings.py*	Description
Drop outliers	*Drop_Outliers*	Whether to drop data point outliers based on **Std. dev**
Pairplot jitter	*plot_jitter*	Create jitter for pair plot scatters for easier visualization of discretized data
Plotted add. data	*vs_adds*	Define additional data to be plotted in versus plots
Plotted channels	*vs_channels*	Define channel data to be plotted in versus plots
Save format	*saveformat*	The saving format of all plots
Std dev.	*dropSTD*	The standard deviation limit for **Drop outliers**

Statistical Plotting:

GUI	Settings.py	Description
Sign. color	*fill*	Do color fill for statistically significant regions
Sign. stars	*stars*	Include significance stars to statistical plots
Neg. log2	*negLog2*	Create negative log2 significance line for statistical plots
y-limit	*ylim*	Y-axis value limit for **Neg. log2** in plot
Observations	*observations*	Plot individual observations

5.8. Stats Window

GUI	Settings.py	Description
Alpha	*alpha*	The P-value limit for the rejection of null hypothesis
Control Group	*cntrlGroup*	Name of the control group
Group vs. Group	*stat_versus*	Create control group versus sample group statistics
Total Statistics	*stat_total*	Create statistics of total cell counts
Trailing / Leading window	*trail / lead*	Number of included bins in front and behind current index position when using **windowed statistics**
Windowed statistics	*windowed*	Statistics done in a sliding window defined by **trail** and **lead**

6 Troubleshoot

- *The resulting cell counts have peaks at the ends of the bin range*

The vectors are not optimally created. Make sure that both ends of the vector extend approximately to the ends of the sample. For example, when creating vectors through skeletonization, resizing of the transformed binary image may lead to incorrect approximation of the sample. As a result, the skeletonization produces a truncated vector and causes overrepresentation of cells in the last bins.

- *The vector ends abruptly when created through skeletonization*

Certain patterns in the sample may cause branching in the skeletonization. As the algorithm follows the skeletonized pixels to create the vector, sometimes these branches can confuse the algorithm into a dead-end. Increasing *'find distance'* can allow the vectorization to jump back to the correct skeleton. Alternatively, trying different resizing or greater smoothing can also solve the problem. The best solution can be determined by looking at the skeleton plot in *'./Analysis Data/Samples/'*.

- *The vector jumps in a unexpected manner when created through skeletonization*

Sometimes the algorithm that finds the next pixels of the binary image is fooled by branching of the skeleton. If the vector jumps into a branch that has already been passed, reducing *'Find distance'* can solve the problem.

- *LAM will not run because system cannot find path at start up*

Modify *'workdir'* variable in settings.py to point at the analysis directory. Double check that the path is written correctly and between pl.Path(r'<PATH>'), e.g. pl.Path(r'C:\\experimentDirectory').