**EyeHex: a toolbox for ommatidia segmentation**

By Huy Tran and Ariane Ramaekers

Flyteam, Laboratory of Nuclear Dynamics, UMR3664, Institut Curie

Corresponding email: [huy.tran@curie.fr](mailto:huy.tran@curie.fr)

# Introduction

FlyHex is a MATLAB toolbox for ommatidia segmentation from 2D images of Drosophila eyes. The toolbox contains multiple Graphical User Interfaces allowing users to (1) perform manual ommatidia segmentation (for generation of training data for the external machine learning module), (2) mapping ommatidia to hexagonal grid, and (3) manual verification/correction of auto-segmented ommatidia.

The toolbox is to be used in combination with the machine learning module from WEKA-trainable segmentation plug-in [1] (included in Fiji toolbox [2]) to preprocess input 2D images.

# Software requirements

MATLAB (tested with version 2016b or later)

Fiji toolbox (download from https://fiji.sc/)

# Installation

1. Download and extract zip file or clone from github.

2. Place all the input images (e.g. img\*.tif) of the fly eyes into Script/data/original/ folder.

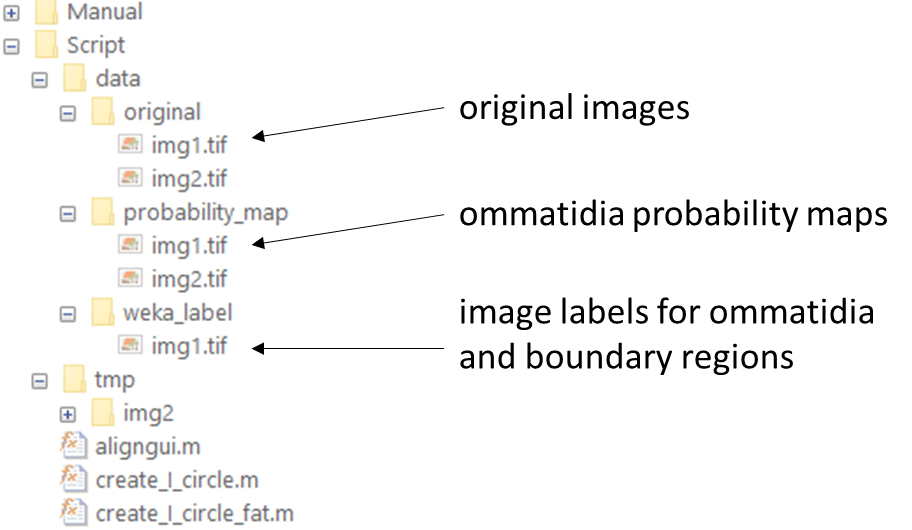


Figure 1. Examples of the Script/data/ folder structure, with two original images. Image img1.tif is used here to generate training data. The ommatidia segmentation is then applied to img2.tif file.

# Preparing training data

This step will help users generate training data for the machine learning module. It needs to done once whenever a new type of image (from e.g. electron microscope, brightfield microscope…) or a drastic change in microscopy settings is introduced.

1. Browse to Script/ folder via MATLAB.

2. Type MAIN\_manual\_segmentation(input\_file) in MATLAB command window to run the manual segmentation GUI. The input\_file is the full name of the input image (e.g. ‘img1.tif’) in string format.

3. To add ommatidia, press **A** and manually clicking on the center of a few adjacent ommatidia. Press **Enter** to register the input. From this group of ommatidia (called a “patch”), a hexagonal grid of ommatidia will be automatically spawned and visualized on the GUI.

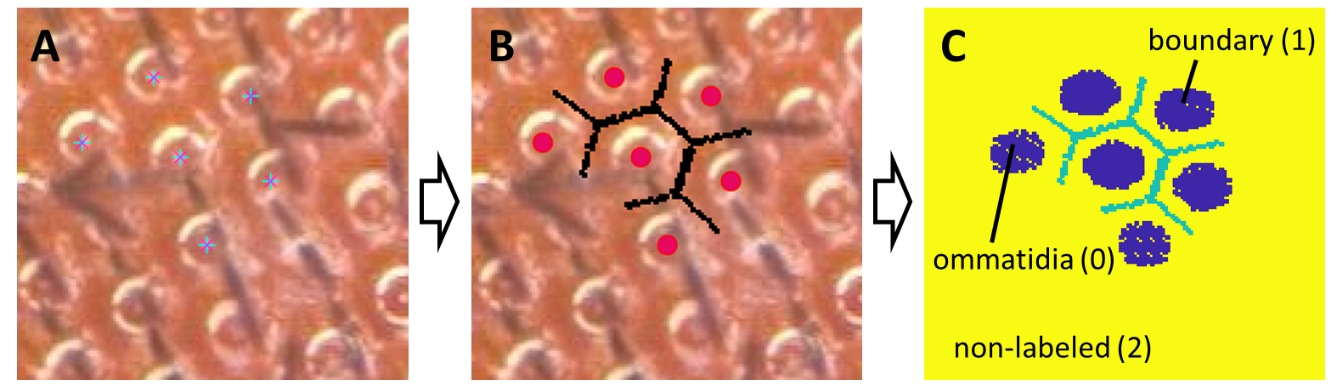


Figure 2. Manual segmentation for the training data. From a few manually labeled ommatidia (A), a hexagonal patch is spawned, with the ommatidia centers (red solid circles) and boundaries (black) visualized in panel B. A labeled image (C) is then exported, containing information on the ommatidia region (pixel value 0), the boundary regions (pixel value 1) and non-labeled region (pixel value 2). The example image (img1.tif) is taken from a brightfield microscope.

4. In case of bad grid spawning, press **R** and remove the patch with the freehand tool.

5. Press **Ctrl + E** to export the ommatidia and boundaries labeled image to Script/data/weka\_label/ folder. The labeled image will have pixel value of 0 (ommatidia region), 1 (boundary region) and 2 (unlabeled region)

Note:

-Press **F1** to access all the hotkeys (zoom in/out, save/load progress)

-Use right-click to zoom in and navigate around different image region.

-When adding a new ommatidia patch, please ensure the first 3 facets are adjacent on the hexagonal grid.

-Try to add patches both at the middle and near the edges of the eye to best

-No need to segment all ommatidia. The machine learning module should work even with less than 50 ommatidia segmented.

# Automated preprocessing with machine learning

This step allows to generate the probability image of ommatidia region, in contrast to the boundary region, based on the trained classifier using the data from the previous step. Here, a macro for Fiji is provided to easy load the training data and apply the apply the classifier to the all eye images.

1. Place the original manually segmented images (e.g. ‘img1.tif’ found in Script/data/original/ folder) to an empty folder called ‘original’.

2. Place the exported labeled images (e.g. ‘img1.tif’ found in Script/data/weka\_label/ folder) to an empty folder called ‘label’.

3. Start Fiji and run the macro TrainClassifier\_gui.bsh in WekaMacro/ folder (by dragging the file to Fiji interface and press F5).

4. A user interface will appear, prompting you to enter the path to directories of:

The original training images (in ‘original’ folder)

The labeled training images (in ‘label’ folder)

The input images to be classified

and output directory to store the ommatidia probability maps.

Select OK.

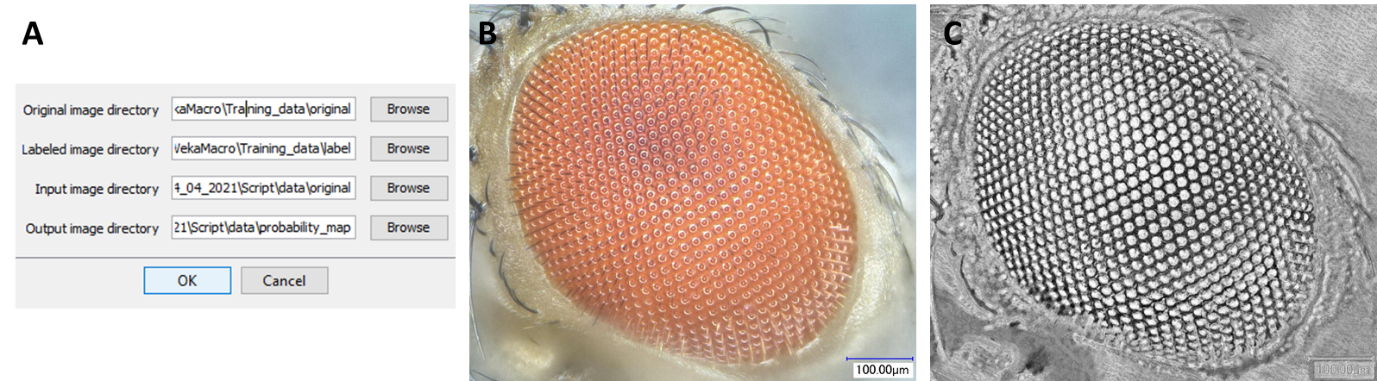


Figure 3. Automated preprocessing with WEKA-Fiji tool: (A) The GUI for selecting directories of training dataset and testing dataset, triggered by running TrainClassifier\_gui.bsh. The trained classifier will convert the original image (B) in the input folder to the ommatidia probability map (C) and save the result in the output folder.

The WEKA plug-in in Fiji will train the classifier with the training data and apply the classifier to all images in input image folder. For each image, a tif file containing the probability map of the ommatidia region is created and saved to output folder.

Note:

-For convenience, you can set the input folder to Script/data/original/ and output folder to Script/data/probability\_map/. However, the classifier will be applied to all images in the input folder, even to ones that have been analyzed.

-The machine learning model can be changed by modifying the TrainClassifier\_gui.bsh file. The default model is set to Fast Random Forest, with 100 trees. However, changing the model (e.g. normal Random Forest, Deep Neural) or increasing the number of trees are found to have little effects on the classifier accuracy.

-The training and classifying process, depending on your computer, takes 2-5 minutes per image.

-From the probability map, you can see which region of the eye is badly segmented, usually due to the lack of training data. Go back to the Manual Segmentation step to focus more on these regions.

# Hexagonal grid expansion

This step generates a hexagonal grid of ommatidia from the learned ommatidia probability map. This grid is spawned from the first 3 user-prompted adjacent ommatidia, which forms the origin and the axes for the grid. This grid will attempt to expand from this origin to detect as much ommatidia as possible (up to 1200).

1. If you haven’t already, copy the original input images to Script/data/original/ and the probability maps to Script/data/probability\_map/.

2. Browse to Script/ folder via MATLAB.

3. Type MAIN\_hexagon\_expand(input\_file) in MATLAB command window. The input\_file is the full name of the input image (e.g. ‘img2.tif’).

4. The ommatidia probability map of the input image will be displaced. Click on the center of three adjacent ommatidia, preferable in the middle of the eye. Press **Enter**.

The program will spawn the hexagonal grid from the three ommatidia. The spawning process will be displayed in real time in the same figure panel and also saved into Script/data/tmp/ folder.

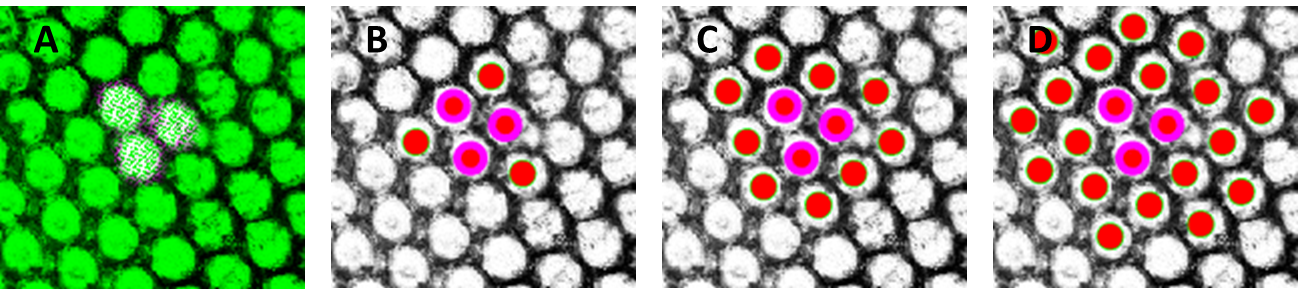


Figure 4. Expansion of hexagonal grid from (A) three manually added adjacent ommatidia. The grid is then expanded automatically layer by layer of ommatidia (B-D) and eventually covers the whole eye.

# Manual correction

As ommatidia at the eye’s edges are heavily tilted, it is difficult for the machine learning module to recognize them properly. Also, non-eye region is not defined, leading to over-spawning of ommatidia during the automatic hexagonal grid expansion. Therefore, a final manual correction is required.

1. Browse to Script/ folder via MATLAB

2. Type MAIN\_manual\_correction(input\_file) in MATLAB command window to run the manual correction GUI. The input\_file is the full name of the input image (e.g. ‘img2.tif’)

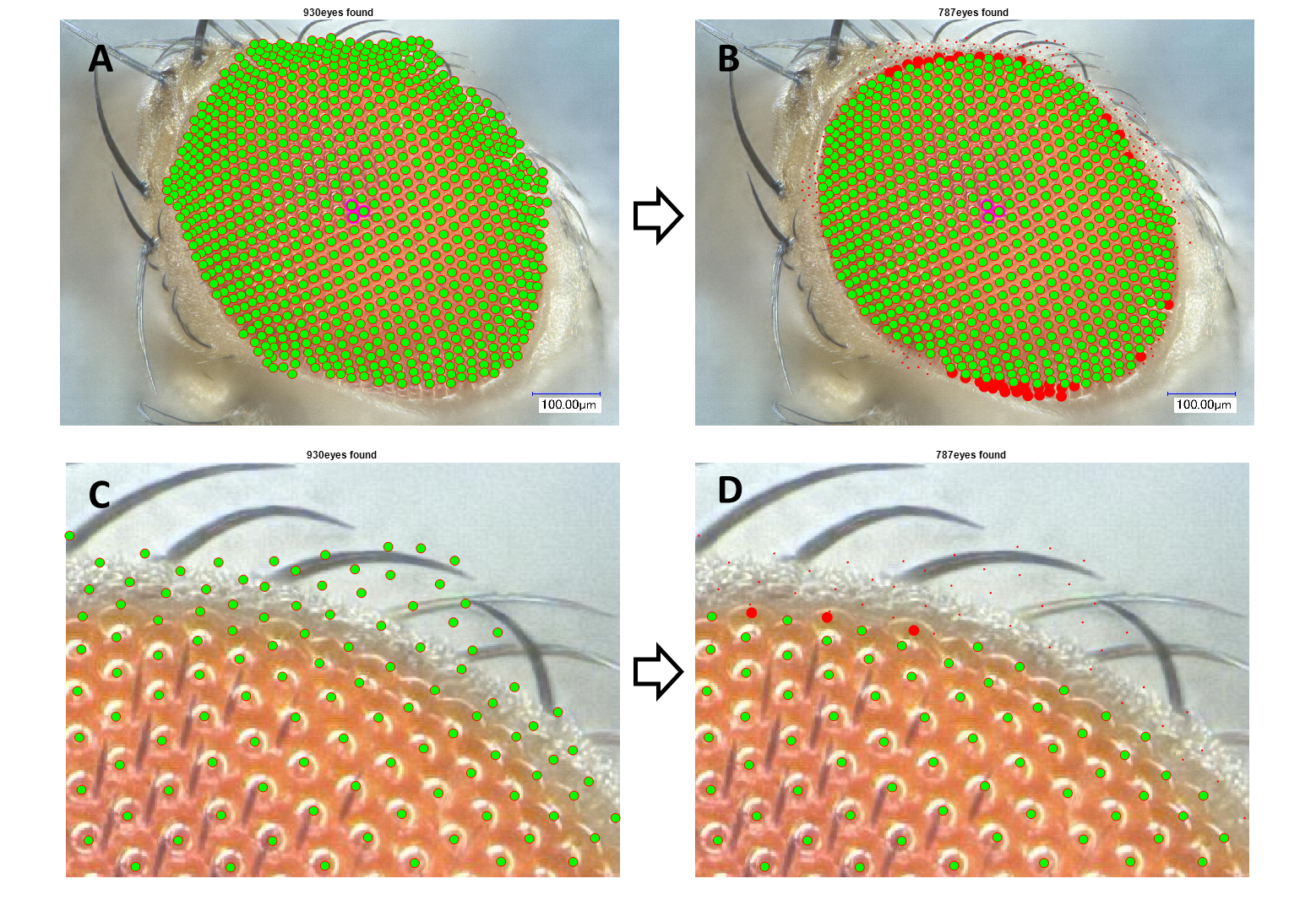


Figure 5. Manual correction after automatic ommatidia detection: (A,C) original image overlaid with automatic ommatidia segmentation. (B-D) manually corrected ommatidia segmentation. Panel C-D is a zoom-in region of panel A-B. In (A-D), automatically detected ommatidia are shown as green circles, manually added ommatidia are shown as red circles and manually removed ommatidia are shown as red dots. The example image (img2.tif) is taken from a brightfield microscope.

The program will load the largest hexagonal grid spawned in the previous hexagonal grid expansion process. You can select a grid of specific size by pressing Ctrl + Left/Right arrow keys.

3. Add (press **A**) or remove (press **R**) ommatidia. You will focus mostly on the edges of the eye where errors might appear. You can save and load the correction progress by pressing **Ctrl + H** (to save progress) and **Ctrl + L** (to load progress). The ommatidia count will always be displayed on top of the image.

4. Press **Ctrl + E** to export the image label to Script/data/weka\_label/ folder.

Note:

-Press F1 to access all the hotkeys (zoom in/out, save/load progress).

-Use right-click to zoom in and navigate around different image region.

# Label alignment

As the hexagonal grid expansion is performed based on the probability map, rather than the original image, there might be some small misalignments between the exported label image and the original image. If you want to know the exact ommatidia position, to extract features from individual ommatidia or to create new training data, you can manually realign the label image to match the original image.

1. In the manual correction GUI, after the ommatidia has been manually segmented, press Ctrl + I to enter the alignment interface. The original image overlaid with the label will be shown.

2. Select a few (~10) anchor points at the center and at the edges of the eye.

3. Drag and drop the anchor points so that the boundary labels match with the boundary in the original image.

4. Press **Enter**. The program will automatically generate the aligned label image and overwrite the current one in Script/data/weka\_label/ folder.

Note:

-The exported label image has the same format as the label image for training, and thus can be used as training data for subsequent image classification.

# License:

Copyright (C) 2021 Huy Tran

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>

# Citing EyeHex:

# References:

[1] Arganda-Carreras, I.; Kaynig, V. & Rueden, C. et al. (2017), "Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification.", Bioinformatics (Oxford Univ Press) 33 (15).

[2] Schindelin, J.; Arganda-Carreras, I. & Frise, E. et al. (2012), "Fiji: an open-source platform for biological-image analysis", Nature methods 9(7): 676-682.