Flynotyper 2.0: A tool for rapid quantitative assessment of *Drosophila* eye phenotypes

Johnathan Ray[1], Deepro Banerjee[1,2], Qingyu Wang[1,2], and Santhosh Girirajan[1,2,3]

1. Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, PA
2. Bioinformatics and Genomics Graduate Program, Huck Institutes of Life Sciences, Pennsylvania State University, University Park, PA
3. Department of Anthropology, Huck Institutes of Life Sciences, Pennsylvania State University, University Park, PA

Abstract

About two-thirds of the genes in the *Drosophila melanogaster* genome is also involved in its eye development, making the *Drosophila* eye an ideal tool for genetic studies. We developed Flynotyper that uses image processing operations to identify and quantify the degree of roughness by measuring disorderliness of ommatidial arrangement in the fly eye. This software has enabled researchers to quantify morphological defects of thousands of eye images caused by genetic perturbations. Here, we updated Flynotyper's computer vision library, improved its performance, and streamlined the analysis for high-throughput analysis of multiple eye images. We also tested several batches of *Drosophila* eye images to ensure robustness and reproducibility of the updated Flynotyper software.

Availability and implementation: The source code for Flynotyper 2.0 can be downloaded and installed from https://github.com/girirajanlab/flynotyper-desktop-application.

Introduction

Drosophila melanogaster remains a robust model organism for genetic studies, offering valuable insights into fundamental biological processes (Reiter *et al.* 2001; Chow and Reiter 2017; Sun *et al.* 2024). The *Drosophila* eye stands out as an exquisite system for genetic screening and molecular studies due to its non-essential nature, ease of phenotyping, and involvement of a majority of essential genes for its development (Thaker and Kankel 1992; Thomas and Wassarman 1999; Kumar 2018). We developed Flynotyper to automatically detect and accurately quantify morphological changes observed in the rough eyes of *Drosophila melanogaster,* thus facilitating phenotypic assessment of the individual units or ommatidia of the fly eye following genetic perturbation. Given a bright-field or scanning electron microscopy derived image of the fly eye, Flynotyper uses computer vision to analyze and calculate "phenotypic scores" related to the eye's ommatidial disorderliness. We previously demonstrated its efficacy by analyzing morphological defects resulting from the knockdown of *Drosophila* orthologs of 12 genes linked to neurodevelopmental disorders and validated through examination of eye images from six independent studies, including screens for modifiers of neurotoxicity and interactors of various genes. Its quantitative analysis accurately classified genetic modifiers of *sine oculis* obtained from genome-wide screens, demonstrating its effectiveness in assessing diverse genetic influences on eye phenotypes (Iyer *et al.* 2016). The current release of the software uses an outdated version of its computer vision tool, introducing bugs that make it difficult to use. Additionally, it can only analyze a single image per execution and provides solely a command line interface (CLI).

Our manuscript introduces an updated version of Flynotyper that provides bug fixes by utilizing a computer vision tool that is supported by computational researchers and developers. Additionally, it provides quality of life improvements by allowing multiple images to be analyzed at once while optimizing time spent performing such computations. Researchers can also analyze their images using a graphical-user interface (GUI). By updating Flynotyper, we hope that it can serve as either an introduction to those who want to facilitate their fly ommatidia analysis or as an upgrade to those who are already familiar with it.

Implementation

The original version of Flynotyper utilizes OpenCV, a computer vision library. With it, the program highlights the fly eye by applying morphological transformation to the original image and drawing a circle over it, isolates each ommatidium with a single circle, and removes any noise caused by the borders between each ommatidium to ensure the most accuracy (**Figure 1**). Once all of this is done, it calculates five different values based on the distance and angle between each of the ommatidium: the total distance ommatidial disorderliness index of all stable ommatidia (OMI_D), the total angle ommatidial disorderliness index of all stable ommatidia (OMI_A), the total ommatidial disorderliness index of all stable ommatidia (OMI), the number of detected ommatidia (Z), and the phenotypic score (P). Previously, OpenCV-2 was the most used version of the library, and we used it when creating Flynotyper. However, developers now use a different version of this library: OpenCV-4. This discrepancy in versions caused issues when installing Flynotyper. Its scripts specifically reference the OpenCV-2 library and search for binaries associated with it. If one were to install OpenCV-4, Flynotyper would not search for those binaries and assume that the researcher does not have OpenCV installed. The updated version of Flynotyper uses OpenCV-4, providing an up-to-date source code for it to use. As a result, the software has fewer bugs and researchers can now install the latest version of OpenCV without having to spend time fixing issues.

In addition to analyzing images with an updated library, Flynotyper 2.0 allows researchers to run calculations on multiple images. Previously, one could only provide a single image as a command line argument for the software to analyze. This allowed results for that image to generate quickly, but at the cost of having to spend extra time rerunning the command for each individual image. Changing the source code to allow Flynotyper to accept multiple images eliminates this. However, the software would run analysis on the images iteratively. This meant that a given image would have to wait until all the images before it to finish their calculations, resulting in more time being spent. To save time, the updated version of Flynotyper utilizes parallel computing with the help of the C++ library OpenMP. Parallel computing allows

for multiple images to be analyzed at the same time on separate cores. This cuts down the time needed to run the analysis.

Usage

Flynotyper utilizes wxWidgets, a C++ library used for creating desktop applications. With it, we created a GUI that is easy for researchers to use and understand. It provides the same functionality as its CLI counterpart while also providing some unique features of its own. Rather than passing a single image to the application, one can pass in multiple in PNG, JPG, JPEG, BMP, TIF, or TIFF formats.

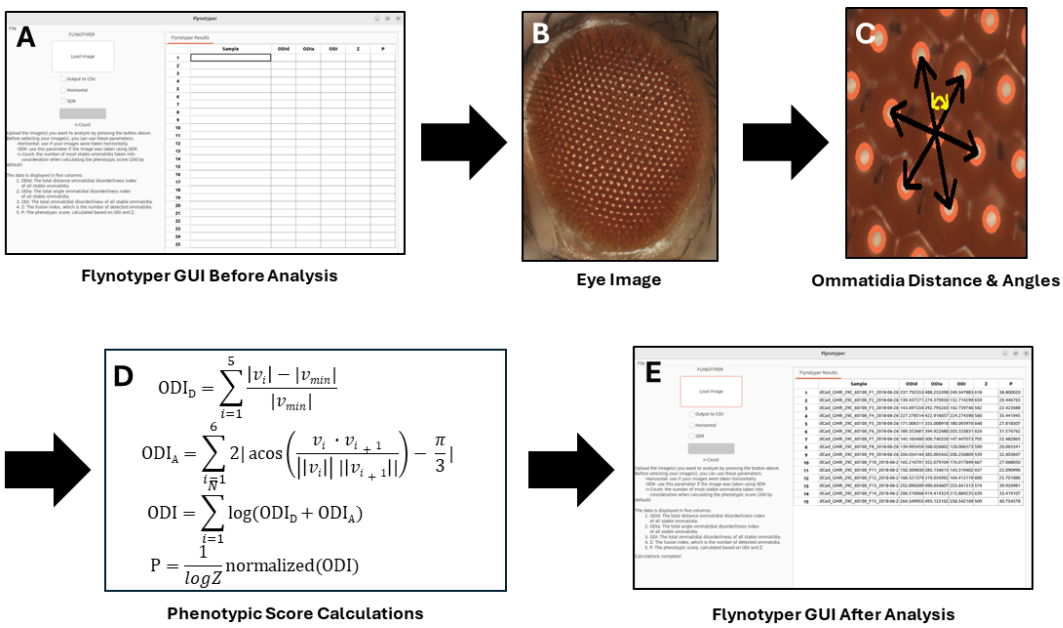


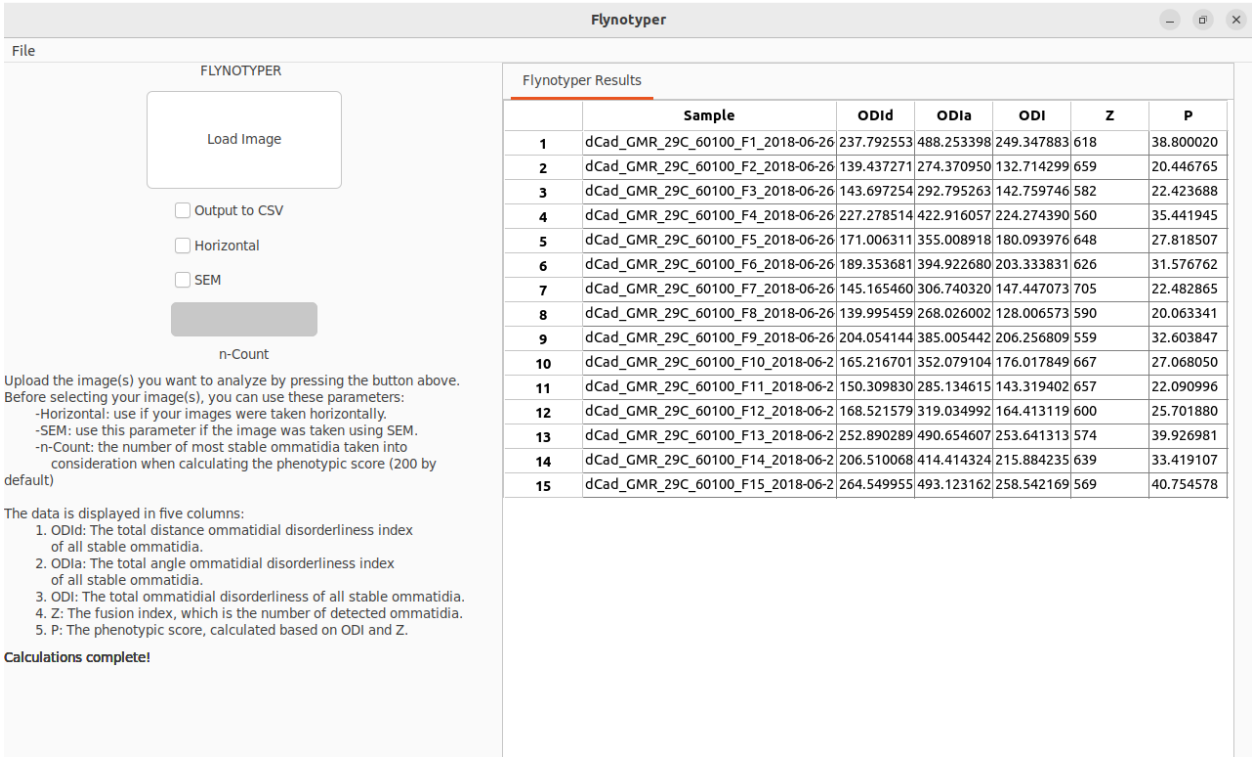
Figure 1: A flowchart representing the process Flynotyper goes through to calculate its phenotypic scores and output them to the GUI. Before analysis begins, researchers are presented with a prompt to submit their images (**A**). For each image they submit (**B**), OpenCV detects and draws a circle on top of each ommatidium. Flynotyper then calculates the distance from and angles between each of them (represented with the vectors found in image (**C**). For each ommatidium, ommatidial disorderliness indices are calculated using the lengths of each of the vectors: the smallest vector, v_{min}, and the five local vectors associated with it, v_i ($i = 1\ldots5$). From there, the phenotypic score is calculated (**D**). Once this is done for all the ommatidia, the results are brought to the GUI and shown in a table (**E**).

Both the GUI and the CLI use a make file that researcher should run first to create the executables needed to run the software. Once this is done, they can run the executable and pass in their images. They can also use three optional flags: the horizontal flag, the SEM flag, and the count flag. The horizontal flag (-h) is used for images that were taken horizontally. The SEM flag (-sem) is used for images that were taken with a bright-field or scanning electron microscope. The count flag (-n) updates the number of stable ommatidia taken into consideration when calculating the phenotypic score. This flag is set to 200 by default.

The source code for the updated version of Flynotyper can be installed on Mac and Linux for free here: https://github.com/girirajanlab/flynotyper-desktop-application.

Discussion

The GUI for Flynotyper (**Figure 2**) has a similar workflow to its CLI counterpart with a few additions. Prior to uploading their images, researchers can check boxes to indicate if they would like to use the horizontal or SEM flags, and they can enter a value to update the count flag to what they would like. There is also an "Output to CSV" flag. This exports Flynotyper's output to a CSV file format should a researcher want a record of their results. Once their images are uploaded (either as a file or a batch of them), the software conducts its calculations (**Figure 1**). For each ommatidium in the image, Flynotyper calculates the distance between each adjacent ommatidium and angles between each of them. These values are used to calculate the phenotypic scores (as seen in image D of **Figure 1**). With the help of the OpenMP library, this takes minimal time. Once Flynotyper is finished, the results are shown in a table. Each row contains five values: the name of the file and the five phenotypic values (OMI_D, OMI_A, OMI, Z, and P).



	Sample	ODId	ODIa	ODI	Z	P
1	dCad_GMR_29C_60100_F1_2018-06-26	237.792553	488.253398	249.347883	618	38.800020
2	dCad_GMR_29C_60100_F2_2018-06-26	139.437271	274.370950	132.714299	659	20.446765
3	dCad_GMR_29C_60100_F3_2018-06-26	143.697254	292.795263	142.759746	582	22.423688
4	dCad_GMR_29C_60100_F4_2018-06-26	227.278514	422.916057	224.274390	560	35.441945
5	dCad_GMR_29C_60100_F5_2018-06-26	171.006311	355.008918	180.093976	648	27.818507
6	dCad_GMR_29C_60100_F6_2018-06-26	189.353681	394.922680	203.333831	626	31.576762
7	dCad_GMR_29C_60100_F7_2018-06-26	145.165460	306.740320	147.447073	705	22.482865
8	dCad_GMR_29C_60100_F8_2018-06-26	139.995459	268.026002	128.006573	590	20.063341
9	dCad_GMR_29C_60100_F9_2018-06-26	204.054144	385.005442	206.256809	559	32.603847
10	dCad_GMR_29C_60100_F10_2018-06-2	165.216701	352.079104	176.017849	667	27.068050
11	dCad_GMR_29C_60100_F11_2018-06-2	150.309830	285.134615	143.319402	657	22.090996
12	dCad_GMR_29C_60100_F12_2018-06-2	168.521579	319.034992	164.413119	600	25.701880
13	dCad_GMR_29C_60100_F13_2018-06-2	252.890289	490.654607	253.641313	574	39.926981
14	dCad_GMR_29C_60100_F14_2018-06-2	206.510068	414.414324	215.884235	639	33.419107
15	dCad_GMR_29C_60100_F15_2018-06-2	264.549955	493.123162	258.542169	569	40.754578

Figure 2*: A visual of what the Flynotyper desktop application looks like. Researchers can customize the flags they want to use prior to submitting files and select any number of images that they need analyzed. Once analysis is done, the results are displayed in a table on the right side, showing a different set of values for each image. If the "Output to CSV" flag is selected, the results are exported to a csv file format that can be found in the directory that Flynotyper was made in.*

Example

We tested the effectiveness of the Flynotyper GUI by running analyses on four sets of images used in a separate study. The study used the original version of Flynotyper to analyze *Drosophila* eyes and identify developmental, cellular, and neuronal phenotypes in genes related to the 3q29 deletion region (Singh *et al.* 2020). We found that the updated version of Flynotyper calculated values (i.e. ommatidial disorderliness indices, phenotypic scores) whose averages matched those found in the study.

Additionally, to test how time efficient Flynotyper is at analyzing multiple images using parallel computing, we ran the software on the datasets twice: once without OpenMP and once

with OpenMP. The analyses that did not use OpenMP and instead went through all the images iteratively took around forty seconds on average. Meanwhile, the analyses that used OpenMP took around twenty-two seconds on average, nearly cutting the calculation time in half (**Figure 3**).

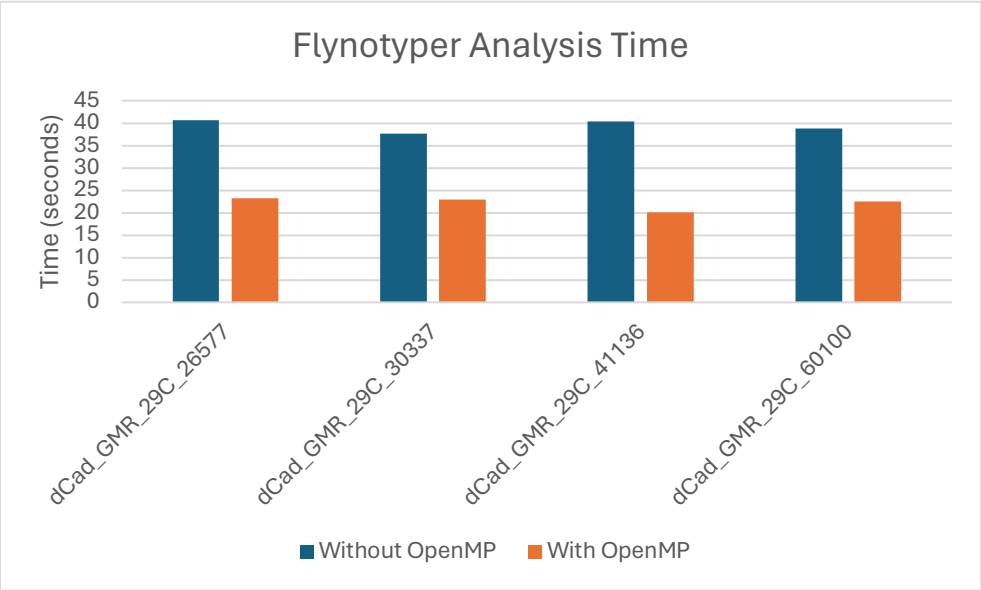


Figure 3*: A graph showing the time it takes for different batches of image data to be analyzed in Flynotyper. On average, the time it takes to analyze the images using a parallel approach is nearly half as much as doing so using an iterative approach.*

References

Chow CY, Reiter LT. Etiology of Human Genetic Disease on the Fly. *Trends Genet* 2017;**33**:391–8.

Iyer J, Wang Q, Le T *et al.* Quantitative assessment of eye phenotypes for functional genetic studies using Drosophila melanogaster. *G3: Genes, Genomes, Genetics* 2016;**6**:1427–37.

Kumar JP. The fly eye: Through the looking glass. *Dev Dyn* 2018;**247**:111–23.

Reiter LT, Potocki L, Chien S *et al.* A systematic analysis of human disease-associated gene sequences in Drosophila melanogaster. *Genome Res* 2001;**11**:1114–25.

Singh MD, Jensen M, Lasser M *et al.* NCBP2 modulates neurodevelopmental defects of the 3q29 deletion in Drosophila and Xenopus laevis models. *PLoS Genetics* 2020;**16**:e1008590.

Sun J, Noss S, Banerjee D *et al.* Strategies for dissecting the complexity of neurodevelopmental disorders. *Trends in Genetics* 2024;**40**:187–202.

Thaker HM, Kankel DR. Mosaic analysis gives an estimate of the extent of genomic involvement in the development of the visual system in Drosophila melanogaster. *Genetics* 1992;**131**:883–94.

Thomas BJ, Wassarman DA. A fly's eye view of biology. *Trends in Genetics* 1999;**15**:184–90.