Project APC 524

Guanhua He (Molbio), Ping Wu (Molbio), Zongjun Tan (ORFE)

November 27, 2018

1 Task description

This project is about to improve some image analyzing codes and their performance related to a study on drosophila embryo conducted by one of our group members Ping Wu. She uses some fluorescent antibodies to stain for endogenous proteins to get spatial gene expression profiles at certain stage of embryonic development. Then, she will infer how the perturbation exerted on the embryos affect the transcription network and downstream gene expression base on the profile. Before making any biological analysis, she requires to extract fluorescent intensities from each cell along the whole embryo, Given the above background information, we would like to achieve two objectives in this project. The first one is to create an interface adapted to the existing codes in order to facilitate the future development in her own research. The second objective tends to improve automation. The initial image processing is still done manually currently, which is time consuming. The initial step requires recognition the morphology of the embryo and distinguish head and tail, dorsal and ventral side, and then rotate and center the object. Some noise removal steps are also required. If we have spare time, we could adapt this code to a series of time lapse images, with cell tracking function and parallel computing function for large scale image processing.

?

2 Organization

The codes will be written in Python. They are basically built on two classes, one for data storage and another one for structure identification and intensity analysis.

2.1 Class for data storage

There are at least 4 type of genes being tracked in this project. For each of them, a photograph of the whole embryo is taken in which we can observe their spatial positions along with the fluorescent intensities.

- **embryo**: an abstract class. Each instance of class embryo represents a sample embryo for study. Two child classes will be derive from it: **gene, topology**. This class also contains a member of helper class: **filter**

abstract methods:

  - apply a filter to a gene object, returns a topology object.
  - apply a filter to a topology object, returns a topology object.
  - study the effect of a topology on a gene object, returns a topology object. This method will be the core for the upcoming biological analysis. In this situation, the class member "filter" in the topology object will be invoked.

- **gene**: this class represents a particular status of some kind of gene. It contains one image of the spatial location of this gene at a certain time. For serial data, it can also contains consecutive images that represent the spatial evolution of this gene along time.

- **topology**: this class represents the spatial structure of a typical fluorescent gene in an embryo. It can contain one or multiple structures of this typical gene in a sample embryo.

  It can have child classes, for example **"boundary"** and **"intensity"**, which are designed for storing special topology information.

- **filter**: this class represents the information of filters that we would like to apply on an image. The image can be a typical gene's spatial locations (store in the "**gene**" object), or a certain topology image of an embryo (store in the **"topology"** object).

2.2 Class for intensity analysis

There are three classes there: **preprocessing**, **analysis**, **drawing** These three classes are wrappers of the "embryo" class.

- **preprocessing**: this class is dedicate for any pre-treatments on an embryo's photograph. It contains a class member of "topology" type.

  Methods:

  - boundary detection, return type: void.
  - rotation/centralization, output: embryo object.

- **analysis**: this class collects all methods for studying fluorescent intensity. It contains a class member of "topology" type.

  Methods:

  - get intensity, return type:void.
  - normalization, return type:void.

- **drawing**: this class provides some methods for viewing graphically the data. Methods have no returns type.

3 Schedule

In total 8 weeks.

- Nov 18 (Sun) – Nov 24 (Sat) (Thanksgiving week):

  Discuss the project with Gabe. Question: what should be included in the prototype? To which extend should we do parallelization? How to represent profiling checking?

  Read reference papers of Ping's project.

  Understand properly the existence code structure. (potentially 1 extra meeting) Clarification of all relationships between existing functions and their input/output.

  List one or two potential requires in the future development.(Ping)

  First version of the prototype. (Graphical + coding)

- Nov 25 (Sun) – Dec 1 (Sat) :

  Second version of the prototype.

  Familiar with parallel programming.

  Familiar with location detection techniques in the existing codes.(To be done)

- Dec 2 (Sun) – Dec 8 (Sat):

  prepare for presenting the prototype in course. (third version)

  alpha version of parallelization.

- Dec 9 (Sat) – Dec 15 (Sat):

  Improve parallelization.

- Dec 16 (Sun) – Dec 22 (Sat) [Winter Break]:
- Dec 23 (Sun) – Dec 29 (Sat) [Winter Break]:
- Dec 30 (Sun) – Jan 5 (Sat) [Winter Break]:
- Jan 6 (Sun) – Jan 12 (Sat):

References