# GSoC 2019 Project
# Towards Better Images.jl Ecosystem

Jiuning Chen
Github: Johnnychen94
Mentors: Zygmunt L. Szpak, Júlio Hoffimann, Tim Holy

March 24, 2019

**Abstract**

This project aims to achieve a better ecosystem for Images.jl, an image-processing toolbox in Julia. Main contributions consist of user-friendly documentation of Images.jl ecosystem, developer manual, and more consistent, robust, and extensible APIs. This project also serves as a subproject to Images.jl v1.0 milestone.

The author is currently a third-year graduate student and Ph.D candidate in School of Mathematical Sciences, East China Normal University, Shanghai. His current research interests are image processing and computer vision, convex optimization, and machine learning. More information about him is listed in section 3.

## 1 Project Description

Images.jl is a Julia image-processing toolbox, including several packages such as: ImageCore.jl, ImageTransformations.jl, ImageAxes.jl. It provides a collection of out-of-box functions[1] to do image processing tasks just like scikit-image and MATLAB Image Processing Toolbox do.

However, despite of the performance, this toolbox at present is still not friendly to both users and developers; unlike other mature julia packages such as JuMP.jl and GPUArrays.jl, Images.jl requires potential users to understand the very details of its mechanism and architecture, and this becomes even

[1]An overview of currently implemented image-processing functionalities is shown at api comparison.

harder for them without comprehensive documentation on it. Under this circumstance, most image-processing researchers are still using Python and MATLAB for their daily work.

Some apparent causes for its poor usability are:

- there're few demos or recipes in Images.jl for new users to start with;
- the APIs lack of consistence and don't match the julian style well;
- there's no image-processing-specific style guide on naming and programming, except the Julia style guide;
- there're too many temporary helper functions defined everywhere;
- Images.jl is an ecosystem but it lacks of a comprehensive illustration of its packages;
- coverage of trait functions are not fully tested.

Fundamentally this is because that it is still in the progress of finding the most suitable programming style to process images using Julia.

Fortunately the problem is well-concerned in the community. Issues such as

- JuliaImages/ImageCore.jl#63 and JuliaMath/FixedPointNumbers.jl#41 – how to deal with overflow behavior of default `N0f8` type?
- JuliaImages/Images.jl#766 – Use `channelview` as possible as we can?
- JuliaImages/Images.jl#767 – Towards consistent style, part 1: a naming guide
- JuliaImages/Images.jl#772 – Revisiting the Images API
- zygmuntszpak/ImageBinarization.jl#23 – What's the appropriate argument order?
- zygmuntszpak/ImageBinarization.jl#24 – Export limited number of symbols?

discuss the coding styles and programming practice in the most generic way. Packages such as HistogramThresholding.jl and ImageBinarization.jl are examples that validate the effectiveness and usefulness of style consensus reached in those issues. For instance, in ImageBinarization.jl, one could binarize an image using any implemented methods[2] with one unified API:

[2]At the time of writing, there're 12 methods implemented.

```
binarize(::BinarizationAlgorithm, ::AbstractArray{T,2}) where {T}
```

With these existing work, it's in the right time to revisit the whole Images.jl ecosystem and head towards a more easy-to-use Images.jl package. This project aims to solve this problem by:

- providing more comprehensive and integrated documentation on both style guide and ecosystem illustration,
- pruning codebase of the ecosystem according to the provided documentation

Writing demos of Images.jl is not included in this project since it belongs to a totally different project. Trait functions will be examined carefully to support high-level API design.

Basically, this is a project on documentation and code refactoring, and it is also a sub-project to Images.jl v1.0 milestone. Potentially involved packages are:

- **user entrance:** Images.jl
- **core packages:** ImageCore.jl, ImageAxes.jl and ImageMetadata.jl
- **application packages:** ImageMorphology.jl, ImageTransformations.jl, ImageDistances.jl and ImageFiltering.jl
- **new packages**[3]**:** ImageBinarization.jl, HistogramThresholding.jl, ImageInpainting.jl

High-level packages (e.g., ImageTracking.jl) and ploting packages (e.g., ImageView.jl) are not under consideration of this project.

This project will have many side effects:

- partially rewritting of user documentation in a more meaningful way;
- potential bug reports and patches to all related Julia repositories;
- introduction of a new image processing packages, ImageEdge.jl, to place legacy methods in Images.jl
- introduction of a new image denoising package, ImageNoise.jl, as a concept-validation experimental field.[4]

[3]These packages are not yet imported by Images.jl.
[4]This is the author's research field.

## 2 Delivery Schedule

As described in the end of section 1, the project will be delivered in two stages: **documenting** and **pruning**.

Documenting and recording in the first stage as a preperation, and cleaning the codebase in the second stage to make real changes to Images.jl ecosystem. With regard to GSoC timeline, Phase 1 evaluates the documentation work, and Phase 2 and Final evaluations focus on the pruning stage.

### 2.1 Documenting

#### Stage Expectations

The main purpose of this stage is to provide trackable records for the next stage's pruning work. There'll be three types of records: **ecosystem documentation**, **developer manual**, and **RFCs** (Request For Comments).

Ecosystem documentation illustrates the scope of image ecosystem and relationships between different relevant packages, it helps users and developers to understand what package belongs to Images.jl and what package doesn't. Developer manual consists of style guide and best practice as well as other related community-operating rules, it gives a documented reference to developers to solve potential conflicts. RFCs with detailed list of API changes and porting operation will be proposed as trackable records for the pruning work in next stage.

#### Stage Workflow

From April 22 to June 24[5], this stage will continue for ten weeks, which will be divided into two periods: **discussion period** and **RFC drafting period**. Ideally, this stage ends after the Phase 1 Evaluation with regard to GSoC timeline. However, since a lot of repositories will be involved in this project, which makes the timeline hard to be sticked to, the timeline serves in a flexible way.

The discussion period begins from April 22 to June 9 (weeks 1 to 7). In this period the community will share ideas and thoughts on the future of APIs and on best practices. Documenting work will be included in this period as well. The RFC drafting period begins from May 27 to June 23 (weeks 6 to 9), one or more RFCs will be drafted and discussed in this period. Basically,

[5]Although the coding officially begins from May 27, the author will start this project when he's available.

the content of RFCs come from previous discussions. The last week of this stage is used for evaluation, merge and announcement.

With a quick discussion among the community, an ecosystem documentation will be add to juliaimages.github.io as soon as possible to reach a consensus on the future of Images.jl, this consensus shall be the fundamental principle to all future discussion. Ideally, the current Images.jl maintainer, i.e., Tim Holy, is supposed to participate[6].

From weeks 1 to 7, many discussions will happen simultaneously in the following way:

1. **Code Review:** dig into source codes of repositories of images ecosystem to find anything that's likely in need of changing. Other mature Julia packages, and image-processing libraries in other languages such as scikit-image and MATLAB Image Processing Toolbox are references.

2. **Issue Open:** open an issue for anything that is worth a discussion, e.g., legacy codes, misplaced codes, codes with bad practice, and undocumented practices and decisions.

3. **Decision Make:** the purpose of discussion is to make decision on API and practice. The conventional principles are taken: a decision is made when consensus is reached, otherwise the current maintainer of Images.jl make the decision. If a decision can't be made before June 16 (Week 8), it'll be dropped as future work.

4. **Record:** all approved, rejected and future-work proposals will be documented in a temporary repository - GSoC2019_Document. Developer manual will be drafted to juliaimages.github.io when there're enough decisions made.

From weeks 6 to 9, RFC drafting[7] will happen simultaneously in the following way:

1. **Code Review:** for each approved proposal, find all involved code pieces, and give a solution to it according to developer's manual. The principle of code review is to rigorously sticking to decisions made in the discussion period – either there's one principle or no principle.

2. **RFC Post:** post the draft-version of RFC in GSoC2019_Document.

[6]In case of maintainer being busy on other work, the author will draft a document based on his understanding and post it to the maintainer to get a feedback.

[7]A RFC Template is available in the Tensorflow community, and 20180827-api-names.md is a good API-renaming RFC example.

3. **Discuss:** if there's any issue with any item in the proposed RFC, suspend the related items and go back to the discussion workflow until a decision is made.

4. **Merge and Announcement:** RFCs will be merged to a new folder in juliaimages.github.io and be announced to the community via slack and discourse. RFC merge and announcement will only happen in last two weeks in case there're more items to be added.

RFC details on how codebase is pruned will be in section 2.2.

### Stage Evaluation

Four items are evaluated during this stage, i.e., Phase 1 Evaluation:

- 2/10: activity on issues and discussions
- 2/10: ecosystem documentation
- 3/10: developer manual
- 3/10: RFCs

A score of 6/10 stands for Evaluation Pass.

## 2.2 Pruning Codebase

After the RFCs being merged and announced to the community, the pruning stage begins.

### Stage Expectations

The pruning stage is to clean the codebase according to the RFC operation guide. There'll be three types of pruning work:

- symbol renaming, move, and removal
- API changes
- API enhancement

For the ease of tracking the progress, a milestone will be set in Images.jl to track the progress, each pruning PR/issue will be assigned a tag.

Challenges during this stage are: backward incompatibility, and complex package dependencies. The next section will be strategies to this challenges.

### Stage Workflow

One principle of the pruning work is to start from packages with the least dependencies to that with the most dependencies. In this project, we do from core packages (e.g., ImageCore.jl), to application packages (e.g., ImageTransformations.jl), and finally to the user-entrance package, i.e., Images.jl.

Question: Should all the pruning work happen in a seperate branch other than master branch?

Porting methods from package package A to package B takes the following routine:

1. implement new methods and unit tests in package B, bump a patch[8] version to package B,
2. in package A, move methods to a seperate `deprecated.jl` file and deprecate them, bump a patch version to package A,
3. wait for one month for exported symbols to give downstream packages time to replace with new symbols,
4. in package A, delete the deprecated files, bump a minor version to package A

Steps 1-2 will be done as a part of this project, and steps 3-4 will be future work.

### Stage Evaluation

With regard to GSoC timeline, this stage includes the Phase 2 and Phase final evaluations. The Phase 2 evaluation shall focus on checking if the pruning work begins as expected, and the Phase final evaluation shall focus on the milestone progress.

## 3 About the Author

My name is Jiuning Chen. I'm currently doing research related to image processing, computer vision, convex optimization and machine learning.

[8]According to Semantic Versioning, minor version should be backward compatible. However, in this project, it's not suitable to bump major version until all the pruning work is done, hence here we bump minor and patch version.

## 3.1 Programming Background

I started to use MATLAB to do research on image processing in the end of 2016, met and immediately fell in love with Julia at Aug 9, 2018[9], and learned Python during the Spring Festival of 2019.

Although my programming career is only about three years, however, I think I'm qualified to achieve the project milestones for the contributions I've done to the Julia community:

- PR: JuliaImages/ImageTransformations.jl#58 reviewed by Christof Stocker and Tim Holy;
- PR: JuliaImages/ImageTransformations.jl#59 reviewed by Christof Stocker and Tim Holy;
- PR: JuliaImages/ImageDistances.jl#8 reviewed by Júlio Hoffimann;
- PR: JuliaImages/TestImages.jl#35 reviewed by Tim Holy;
- PR: JuliaLang/julia#29626 reviewed by Matt Bauman;
- PR: FluxML/Flux.jl#372 reviewed by Mike J Innes;
- PR: FluxML/Flux.jl#371 reviewed by Mike J Innes;

to members in the lab of my supervisor:

- Set up the whole self-hosted research platform independently from scratch for my supervisor's laboratory[10];
- *De facto* maintainer of the deep learning servers of the School of Mathematical Sciences, and that of a laboratory in Computer Sciences Department;
- Proudly create and maintain the homepage of my supervisor, prof. Fang Li;

and to undergraduate students in the university:

- Head teaching assistant of courses of "Deep Learning and Action (Fall 2018)" and "Digital Image Processing (Spring 2019)".

[9] It's the day after the historic announcement of Julia v1.0.

[10] The platform includes but not limited to homepage, documents for users and administrators, server monitor, gitlab, jupyterhub, sharelatex, DNS servers, and VPN servers. I'd like to show you how this looks like but it's all built in a LAN environment.

- Unofficially mentor talented students with all the best programming practices I learned from the open-source community and from the English world[11].

The following is an informal self-evaluation to let you have a more structural overview of my skill:

- **Mathematics & Image Processing (8/10)**: my current research is on image denoising based on hybrid method of variational model and deep learning;
- **Linux (7/10)**: heavy usage of docker, bash, git and vim in my daily work to maintain the servers;
- **Matlab (8/10)**: the only programming language used throughout my early-stage of research;
- **Julia (6/10)**: fully understand and stick to the philosophy of Julia, but lack of real project experience;
- **Related packages (6/10)**: familiar with other open-source image-processing packages but haven't dig into them yet;

## 3.2 Education Background

**2016-Present (Postgraduate)** [12] Study on image processing and computer vision in School of Mathematical Sciences, East China Normal University, and supervised by Prof. Fang Li.

**2013-2016 (Undergraduate)** Bachelor of philosophy, Department of Philosophy, Shanghai University.

**2011-2013 (Undergraduate)** Study on metal material in School of Material Sciences, Shanghai University.

[11] Most Chinese students are afraid of reading English since it's not their native language, however, almost all best materials are in forms of it. My role here is to learn and to preach.

[12] The author just passed the Ph.D qualification examination and will be a Ph.D candidate in September 2019.