Live 2D Classification from Warp

The Live2D tool is a lightweight python-based application to do semi-automated 2D classification of cryoEM particles simultaneously with data collection. It operates using a simple web interface for the frontend and uses Warp and cisTEM 2 on the backend.

Author: **Benjamin Barad** *benjamin.barad@gmail.com/ baradb@gene.com*
Date Modified: *2019-10-11*

Getting Started

These instructions will get Live2D functioning on your local machine. It is recommended to install one copy per microscope onto separate workstations - it is a CPU-heavy multi-process application and is currently tested for linux only (but should in theory work for windows). Performance scales well up to 32 cores tested.

This application cannot currently be installed on a cluster, but we are interested in extending it to do so.

Prerequisites

- cisTEM >= 2.0 - cisTEM version must be using STAR files instead of PAR files.
- Warp on a separate workstation (same computer behavior is untested, but warp is GPU-limited while Live2D is CPU-limited, so they may play nicely together).
  - The application requires that warp has been set up with `Pick Particles` and `Export Particle Coordinates` setting checked.
- Python >= 3.7

Installation

Edit your `.bashrc` or `.zshrc` to ensure that the default shell has `refine2d` and `merge2d` in the `$PATH`.

```
git clone https://github.com/bbarad/Live2D
cd Live2D
pip install .
```

Building documentation (optional):

```
cd docs
pip install mock
make html
```

Configuration

Most server-level configuration happens in `$HOME/.live2d/server_settings.conf` - this file is generated the first time `live2d` is run, and contains a set of variables that are loaded into the app on launch, and which can be changed at launch by command line flags. Minimally, users need to set `warp_prefix` and `live2d_prefix`, which are the parent paths for individual warp directories and live2d directories, respectively, in the user's workflow. The individual folder name for the warp folder will be copies as the folder name for the live2d folder, but in the different specified location. Additionally, `warp_suffix` and `live2d_suffix` can be used if it is desirable to use subfolders of the project-level unique-named folder, such as in cases where one folder structure houses raw data and processing output. It may also be convenient to change the port to `8080`, which will allow viewing of the site at `http://$HOSTNAME` without supplying a port. You may also want to modify `process_pool_size` - this is the number of processors that will be used for `refine2d` jobs, and should never be greater than the number of logical cores available to the workstation. By default,

`process_pool_size` is 32, but increasing it will dramatically improve performance if there are more than 32 logical cores available in the workstation.

Running the server

The application runs via a Tornado server in python. By default, the application runs on port `8181`. This behavior is user configurable - see the Configuration section for more details. The server must be run by a user with read and write permissions to the folder that Warp is working in. At launch time, configurations can be overridden by command line flags.

```
live2d
```

or

```
live2d --port=$LIVE2D_PORT
```

This application currently does not do any user authentication. Ensure that the port you choose is only accessible on a secure network. For additional security, make sure that the port is not exposed at all, and that the website can only be viewed from the local machine.

This will launch the Tornado server. After this, the website can be viewed in any modern browser (any with support for websockets) at `http://$HOSTNAME:$LIVE2D_PORT` (or `http://$HOSTNAME` if you used port `8080`, or `http://localhost:$LIVE2D_PORT` if you are viewing on the same machine the server is running on). It is recommended to run the server in a detachable session (`screen` or `tmux` can help with this) in order to allow it run to for long periods of time - under ideal circumstances, the server should be able to do many live 2d classifications between re-initializations.

Using the Server

The website is organized with user controls in a panel on the left and results in a larger panel on the right. When the application is first opened, no results are available, and none will be visible. The server has global state - that is, individual users for the most part are looking at the same information at all times. If you open a new page, on your phone, the same computer, or a different computer, it should present the same information to all of those pages. Within a page, you can stage settings changes (which get sent to the server and all other open clients on job submission) and browse through previous results.

Setting up your job

The user's first action should be to change the directory by clicking the **Update Warp Directory** button. This will open a modal with a dialog asking for a new warp folder. Enter the directory name for the folder where warp is reading movies and writing out its results (you set that directory up in the Warp gui, but it may be a slightly different path depending on whether you use a `Samba` fileshare or other way to make files accessible to windows) and click **OK**. If Warp has run or is running in that folder, the server will read the `previous.settings` file output by warp in that folder and set up a new `classification` subfolder in the same directory that will house results from 2d classification.

One a warp folder is successfully opened, most 2d classification-related settings are automatically pulled from warp, including guessing that the mask radius is likely to be the same as the particle size used to pick particles in Warp. Remaining settings are laid out in the **User Settings** and **Expert Settings** tabs, which can be collapsed or expanded by clicking on them.

Starting a job

Three buttons are available: **Start Job Now**, **Start Automatic Jobs**, and **Stop All Jobs**.

- **Start Job Now** sends user-chosen settings to the server and then triggers an immediate 2D classification job, which iteratively runs `refine2d` at different resolution cutoffs and subsets of particles, mimicking the behavior of the 2D classification job in the cisTEM GUI. When this kind of job finishes, automatic jobs are automatically started.
- **Start Automatic Jobs** sends user-chosen settings to the server and then tells the server to begin watching for new particles being picked by Warp. The server will trigger jobs automatically at regular intervals (which can be changed in the **Expert Settings** tab).
- **Stop All Jobs** immediately stops the system from watching for new particles from Warp, and will end any currently running jobs after the next iteration of `refine2d`. This can take a long time when many particles have been picked by warp, but in order to avoid unexpected behaviors, there is no current way to kill a job in the middle of a `refine2d` iteration. Killing and restarting the server will stop a job mid-run, but do this at your own risk! There is a chance damage the configuration settings file for the server, requiring manual fixes

Clicking either of the **Start** buttons will send the settings chosen by user, and are the only time user settings get sent to the server.

Settings Overview

User Settings

Very good results can be achieved by only interacting with these two settings. Start with just these settings, and then move to **Expert Settings** to improve performance only when necessary.

Mask Radius
The radius of the circular mask applied to the class averages before new rounds of classification. This value should be a little bit larger than your particle. Live2D makes a conservative (on the large size) initial guess of a good mask radius based on the particle size used in Warp.

Classification Type
This setting switches between 3 macro programs for 2D classification.

- *Ab Initio* - Perform a classification from random seed classes, starting at low resolution (40Å by default) and iteratively increasing the resolution cutoff with a small set of the particles, followed by several rounds of classification with 100% of particles at high resolution (8Å by default). When no previous classes have been calculated for the current set of warp-generated particles, this is the required option. Before classes are clearly resolvable as different views of the target, this is a good default behavior. When an *Ab Initio* job finishes successfully, the setting for the next automated run is changed to *Seeded Startup*.
- *Seeded Startup* - Perform a series of classifications that start at low resolution and iteratively improves, using the most recently generated classes as the starting point for classification, then perform several rounds of classification at the high resolution cutoff. Uses all particles from the beginning, unlike *Ab Initio*. This is the setting that should be used for the majority of the data collection, and is the default after the first job if automatic jobs are run from the beginning. Should not be used until enough particles have been collected to get clearly distinguishable class averages - generally around 50k particles is plenty, and sometimes this setting can be changed sooner than that if the classes look good. When hundreds of thousands to millions of particles have been collected, this setting can begin to take 10+ hours, and if the classes are already very good, new particles can instead be more efficiently incorporated with the *Refinement* setting.
- *Refinement* - Perform a series of classifications with the most recent set of classes as a starting point, with all classifications at the high resolution cutoff and using all particles. High chance of overfitting to local minima - only switch to this when the existing classes are very good and enough particles have been collected that seeded startups are prohibitively slow.

Expert Settings

Initial High Res Limit
This is the high resolution limit used at the beginning of *Ab Initio* and *Seeded Startup* runs. Generally, leaving this at 40Å is best, but when a decent number of particles are picked and classes are good but not good enough yet for *Refinement* raising this limit to 20Å or better and decreasing the corresponding number of **Startup Cycles** can improve speed with minimal impact on quality.

Final High Res Limit
This is the final high resolution limit for *Ab Initio* and *Seeded Startup* runs, which will iteratively increase resolution from the **Initial High Res Limit** to this number, and the resolution limit used for all *Refinement* classification jobs. Leaving it at 8Å is safe generally, but cisTEM lets you change it, so I thought I should too.

Startup Cycles
The number of cycles in *Ab Initio* and *Seeded Startup* jobs to move from the **Initial High Res Limit** to the **Final High Res Limit**. Resolution will increase linearly over this number of cycles (so 40 to 8 in 15 cycles means the resolution will improve by ~2.3Å per cycle). This number is ignored in *Refinement* runs. For *Seeded Startup* runs, you can decrease this number along with raising the **Initial High Res Limit** to get faster classification and corresponding faster incorporation of new particles, at the cost of some risk of overfitting to local minima when the starting classes are poor.

Refinement Cycles
The number of cycles to perform with all particles at the maximum resolution. *Refinement* runs only use this number of jobs. For *Refinement* jobs when hundreds of thousands of particles have been collected, decreasing this number so that new particles are incorporated more often is advised - I often drop it to 2 cycles in *Refinement* mode after a million particles have been picked by Warp. It is a good idea to leave this number at 3 or above for *Ab Initio* and *Seeded Startup* runs, as the classes will often improve a lot in the first few high resolution classifications after starting from a low resolution seed.

Particle Count Before Starting
This is the threshold number used by the automated job starter to trigger the first classification job for a warp run - when no classes exist at all. If you change your warp settings and have to do a new *Ab Initio* run, this is not the setting that will be used in that case - this only gets used for the very first class. Despite this setting existing, the authors usually trigger the first job manually once sufficient particles (at least 15k, but the default is 50k) particles are picked.

Additional Particle Count Before Update
Number of additional particles that need to be picked by warp before triggering a new job. Often late in runs more particles than this will be picked before a *Seeded Startup* job completes, unless you change settings above. Early on, dropping this to 50k can give faster feedback on whether a good variety of views are getting picked by warp.

Number of Classes
The number of classes to be generated during an *Ab Initio* job. 50 is generally a good number, but this can be guided by previous cisTEM classifications of this or similar proteins. Computation time scales linearly with number of classes, so responsiveness may be impacted when this value is increased.

Initial Particles Per Class
The number of particles per class to be used for the startup runs in *Ab Initio* jobs, when a subset of particles are used. 300 is the cisTEM default. In sessions with significant preferred orientation problems, I have found increasing this number to 1000 can help with getting good results (at the cost of some computation time).

Automask
This flag tells cisTEM to try to automatically mask class averages. This can result in overfitting, so leave this off before the classes are good.

Autocenter
This flag tells cisTEM to automatically center class averages to their center of mass. This is on by default, but can result in bad classes looking worse.

Results Panel

The results panel is organized into two subpanels: **Classes** and **Job Output**.

Classes
This is the primary output subpanel for users to be focused on. When first loaded, it displays the latest set of class averages from live processing. At the top of the panel are two different ways of selecting previous classes, which is useful for following the trajectory of classification and seeing how addition of new particles has impacted classification (which may be useful for deciding when to stop). Below that is a small amount of metadata about the current classification displayed. Finally, all of the class averages are displayed in columns resized to ~180 pixels wide and tall. Clicking on a class average brings up a modal with the full size class average (up to the size of the results panel) as well as the number of particles in the class. In this view, the left and right arrow keys can switch between class averages. This view is recommended for determining the detail level available in the images.

Job Output
This subpanel has the last 1000 lines of the processing log available. It can be used to get information about processing, and generally should have errors logged (See When Something Goes Terribly Wrong).

Migrating to offline processing.

Live 2D iteratively generates a combined particle stack, named `combined_stack.mrcs`, and a combined cisTEM-style star file named `combined_stack.star`, which can be used (in addition to the `allparticles` starfile output by Warp) to import particles for further classification and refinement using standard 3DEM software such as relion, cryosparc, and cisTEM. This is often more convenient than combining the particle stacks output by Warp manually.

Additionally, more comprehensive Live 2D (+ Warp) results can be imported for off-line processing using `cisTEM 2`. `cisTEM 2` (`>=r1125`) includes a command line program called `warp_to_cistem` that will generate a new `cisTEM` project and import results from Warp, and if desired, results from Live2D. The program allows for import of as much or as little information as desired, from movies alone all the way through importing classification results. The results are imported to allow for complete cisTEM functionality, including resizing boxes, choosing particle subsets from 2d classes, adjusting particle picks, refining CTF, etc. **NB: Most files are not copied to the cisTEM project folder and instead are referenced in their current location, so if your pipeline involves moving warp and/or live2d files to an archival location, ensure that `warp_to_cistem` is run after that move to ensure the continued functionality of the `cisTEM` project.**

When something goes terribly wrong

Sometimes something goes wrong - the filesystem breaks down for a moment, a websocket sends something really uninterpretable by the server, or someone updates cisTEM and suddenly all the paths are different!

When this happens, the easiest thing to do is generally to kill the server process with a `SIGINT` message (`ctrl+C` on linux/mac) and restart it. Generally, you will reinitialize with the same state as you were in before the crash (no need to repeat the setup), and can continue without problem. Occasionally, the program will not relaunch (or will soon crash again) for one of a few reasons.

1. Check that the warp folder is accessible and that the server account has write access - if either changes, the program will not work and will be very unhappy.
2. Check that the `$HOME/.live2d/latest_run.json` file in the folder the server runs from (usually the git repository folder) is correctly formed - occasionally an error can occur during writing and it will end up malformed. If this happens, copy the `latest_run.json` file from the `$LIVE2D_FOLDER` to `$HOME/.live2d/latest_run.json`. You might lose the most recent class (if it was the one that the write failed on) but that should be the maximum amount of data you lose.

3. Check that the server account's path has `refine2d` and `merge2d` from cisTEM in it - generally, if `cisTEM` is in your path, they will be too.
4. Try pulling the latest version of the github repository - occasionally, something changes in `refine2d` or `merge2d`, and the inputs sent by live2d need to be changed accordingly. Generally we will stay on top of this, and you can get the fix quickly.
5. It is possible for the particle stack to become irreversibly corrupted. Fix this by setting `next_run_new_particles` to `true` in `$HOME/.live2d/latest_run.json` and restarting the server.
6. Open an issue on Roche Stash or Github. You may have found a novel bug.

This PDF was automatically generated from a readme markdown file. To regenerate it after making edits, run:

```
module load apps/pandoc
module load apps/texlive
pandoc -V geometry:"margin=1in" --highlight-style=pygments -f gfm readme.md -o Live2D_User_Guide.pdf
```