RCT in EMAN2 Tutorial

Getting Started:

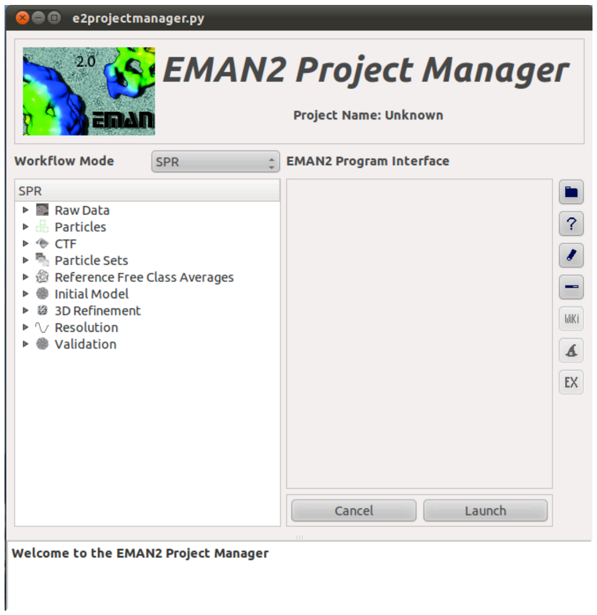
Make sure you have the latest version of EMAN2. For now, you will need either the nightly build or EMAN 2.05, available for download at http://blake.bcm.edu/emanwiki/EMAN2. In the future, you should use EMAN2.1. You also need to download the test data associated with this tutorial, and then untar it to get the simulated Ip3R dataset. For this tutorial, you will work with Ip3R1, a 1.3MDa membrane protein. This data consists or a stack on simulated untilted image, and a stack of simulated 60 degree tilted images.

Introduction:

EMAN2 is an image processing software package directed towards electron microscopy data. It is composed of a C++ core and a suite of python programs that implement higher-level data processing functions. On top of this infrastructure, there is a GUI based data workflow framework, which will be used and introduced in this tutorial.
This tutorial aims to guide the user through steps required to generate a 3D reconstruction using RCT techniques. This tutorial uses simulated data to illustrate a clean example, in the real world, life is usually not so easy. This tutorial focuses just on the steps required to make an RCT recon starting from untilted and tilted data stacks. If you want guidance on picking untilted-tilted particle pair stacks, see the ***tilt-validation tutorial***.

Initializing a new project using e2projectmanger.py:

To begin a new project, move to your desired project directory and launch *e2projectmanger.py*, the new EMAN2 GUI. A window should pop up as shown below:



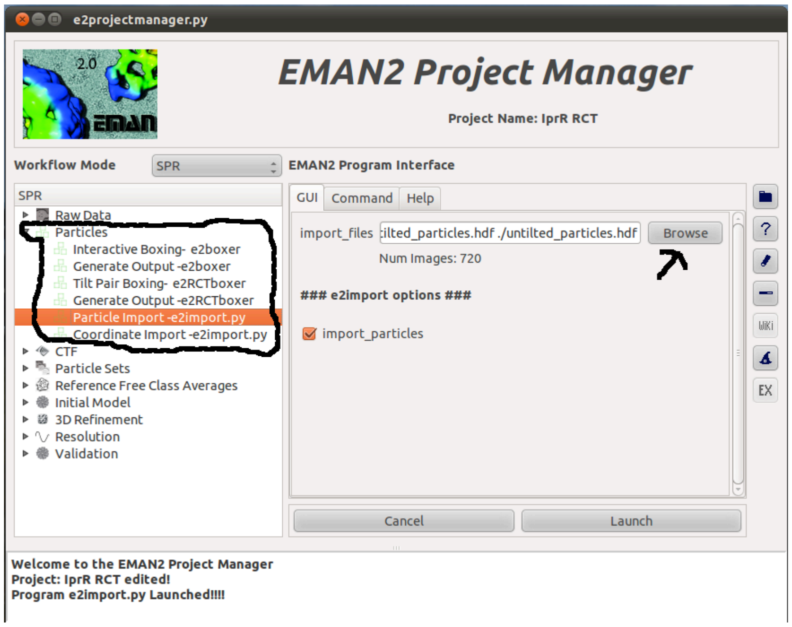
First you need to set project parameters, so in the menu bar (the menu bar location is platform dependent) click: *Project->Edit*. A dialog box will pop up. Here set the project parameters (then click **OK**).

	Ip3R
Project Name	Ip3R RCT
Project Icon	Use default
Mass	1,300
CS	2.0
Voltage	200
Apix	1.88

Importing the simulated data:

Since we already have a stack of untilted images and a stack of tilted images, all we need to do is import the particles. In the projectmanager workflow tree, click on the *Particles* tab. Next click on *Particle Import*. This will load a GUI interface to e2import.py, which allows importing of particle stacks into the current project. You want to find the particle stacks to import, so click on the browse button in the GUI interface widget. This will launch a browser dialog box allowing you to search for files. You should select *tilt_particles.hdf* and *untilted_particles.hdf* for import using multi select (ctrl + click), then click **OK**. The files you want to import will now be listed in the option field *import_files*, so click **Launch** to run the program. As a side note, you can click on the **Command** tab (to the right of the GUI tab) to see the command that will be executed. Advanced users can edit this line to customize behavior.

BTW: If you want to learn how to actually pick untilt-tilted particle pair stacks, see the ***tilt-validation*** tutorial.



import_files ilted_particles.hdf ./untilted_particles.hdf

Num Images: 720

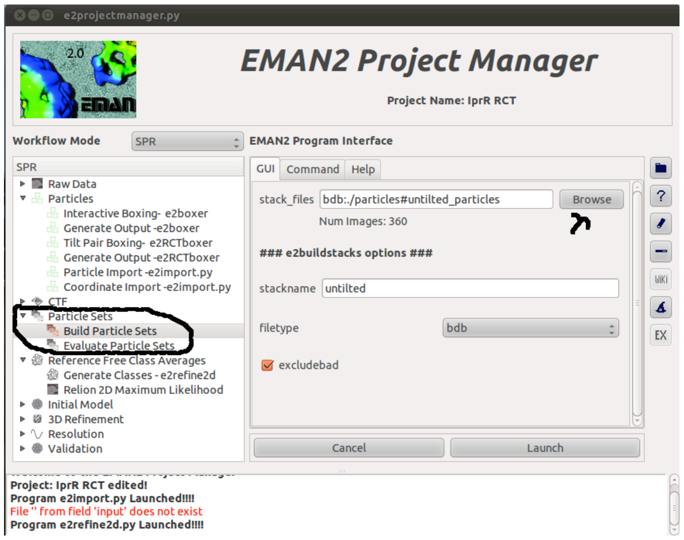
e2import options

☑ import_particles

Welcome to the EMAN2 Project Manager
Project: IprR RCT edited!
Program e2import.py Launched!!!!

Building particle stacks:

After particle import, we want to build particle stacks. Click the *Particle Sets Tab*, listing options to build particle stacks for processing. Then click *Build Particle Sets*, loading the GUI to build stacks for processing.
In the GUI, click browse to search for stacks to build. The file browser will launch. Select all the stacks corresponding to the untilted images. In this case there is only one stack, untilted_particles, but in most projects there will be multiple stacks to process. Choose a name for the stack and write this name in the option field 'stackname'. Click **Launch** to run the program, e2buildstacks.py. Do the same for the tilted images, naming the stack *tilted*.

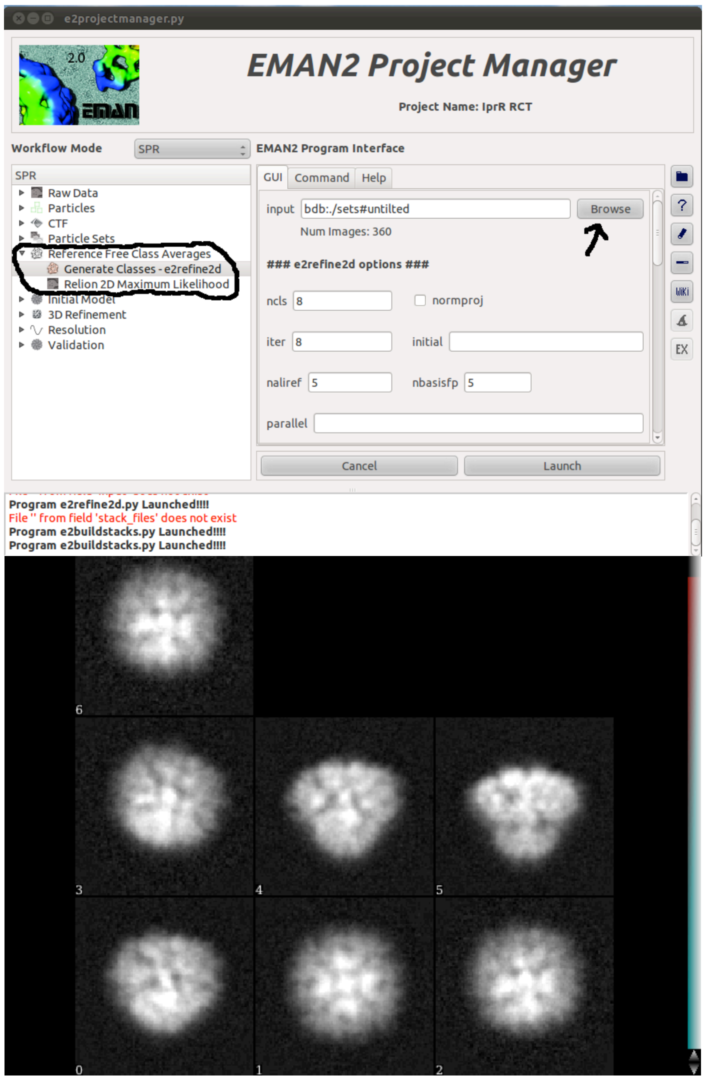


stack_files bdb:./particles#untilted_particles
Num Images: 360
e2buildstacks options
stackname untilted
filetype bdb
[✓] excludebad

```
Project: IprR RCT edited!
Program e2import.py Launched!!!!
File '' from field 'input' does not exist
Program e2refine2d.py Launched!!!!
```

Generating reference free class averages using e2refine2d.py:

The first step towards actually making RCT reconstructions is to sort the data into self-similar classes. For this step we use multiple statistical analysis techniques to produce reference free class averages. e2refine2d.py implements this technology. To run e2refine2d.py in the context of e2projectmanager.py, click the tab, *Reference Free Class Averages* listing options for reference free class averaging. In this tab click *Generate Classes – e2refine2d* to load the e2refine2d.py GUI. There are many options for e2refine, but we only need to alter a few from their default values. First click browse to and select the untilted stack using the file browser. Click **OK** in the browser. Second set the option *ncls*, the number of classes to 8. For RCT we want to have a large number of particles per class. Reducing the number of classes achieves this, but at the cost of the particles in each class not being as 'self-similar'. Last, set the option *parallel* to thread:X where X is the number of cores on your machine. Lick **Launch** to run e2refine.py.
After the program has finished running (you can check the status by clicking on the task manager button, 4th tool button from top), examine the class averages by clicking on *Reference Free Class Averages* node. Do not click on the triangle shaped icon as that controls child expansion, as you have previously done in this tutorial. Clicking in the text area of the node opens a table with e2refine2d results. Open r2d_01 and find classes_08. Clicking on this file reveals several buttons at the bottom of the browser. Click **showstack** to view your class averages. You should get something looking like the image montage below. Normally your data is not this good, so when using real data, you'll need to note any 'bad' classes to exclude from RCT processing. Bad classes are classes that have not black CTF hallow(in real data) surrounding them or have no structure (look like a blobby disk).



Project Name: IprR RCT

input bdb:./sets#untilted

Num Images: 360

e2refine2d options

ncls 8 normproj [ ]

iter 8 initial

naliref 5 nbasisfp 5

parallel

```
Program e2refine2d.py Launched!!!!
File '' from field 'stack_files' does not exist
Program e2buildstacks.py Launched!!!!
Program e2buildstacks.py Launched!!!!
```

Computing RCT recons for each class average using e2rct.py:

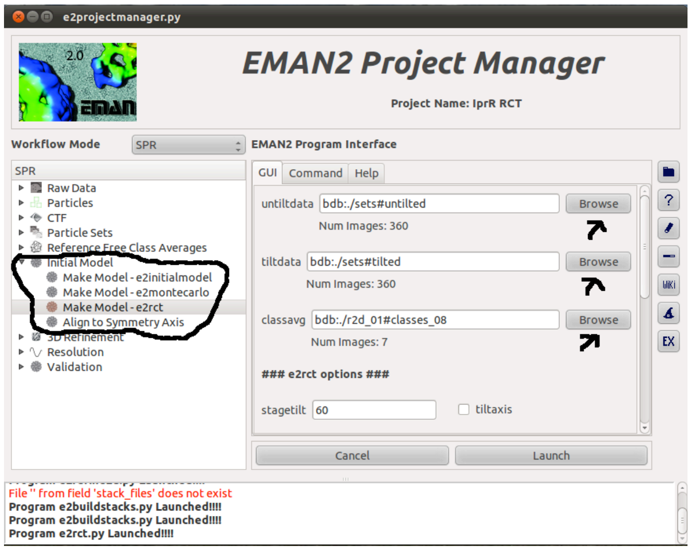
Finally, we create a RCT reconstruction for each class average using e2rct.py. Click on the *Initial Model* tab listing options for initial model building. We want to do RCT, so click on *Make Model - e2rct* to load the GUI interface to e2rct.py. Load the untilted stack, as before, by clicking on browse, then selecting the untilted particles stack. Click **OK** to import into the GUI. Do the same for the tilted images, and the *classavg* field, except you will need to browse to the r2d_01 directory first. Choose the best-looking class averages stack, which will almost always be the class averages from the last iteration.

Next we need to set the options for e2rct.py.

- *stagetilt*, set this value to 60 for Ip3R simulated data. This is the amount of simulated tilting. In real data the value will be equal to the amount you tilted the stage during data collection.
- *titlaxis*, you can check this box to do a titlaxis correction, this only works if you picked particle pairs using e2RCTboxer.py. If you imported them from another program, or you are using simulated data as we are now, leave this box unchecked, as the simulated data all have the same tilt axis.
- *careject*, list bad class averages you don't want RCT reconstructed. Since we are using simulated data, we use all class averages, so leave this line blank. In real data list class averages to reject in comma delimited form.

- *minproj*, set the minimum number of particles required to compute a RCT recon. Obviously 1 or 2 particles will not produce a decent 3D volume, so I recommend setting this value to 10
- *align*, check this box if your tilted particles need to be centered. Centering works by computing an average, translationally aligning to this average, and then iterating the process. This may not work well for highly elongated particles. For this dataset check this box.
- *maxshift*, set the maximum translational shift allowed during tilted particle centering. For the Ip3R simulated dataset set this to 6 pixels.
- *avgrcts*, after creating an RCT for each class average, the RCT recons are aligned and averaged to produce an optimal RCT recon, averaging away the missing cone. For this tutorial, check this box.
- *reference*, without a reference with RCTs are aligned to each other, however, if you align to a reference model, you can get improved results. For this tutorial we do not need a reference because our data is unrealistically good.
- *sym*, specify the symmetry of your structure. You should only use this option if you provide a reference AND this reference is aligned to the symmetry axis. If you don't have a reference, you can run e2rct.py with no reference setting *sym* to 'c1'. Then you can align your RCT recon to its symmetry axis using e2symsearch.py. Use this model, aligned to its symmetry axis, as your reference and set *sym* to the correct symmetry. For this data, since we do not have a reference leave the default set to 'c1'. You are welcome to try the above suggestion, but this tutorial will not walk you through this.
- *aligngran*, the fineness of 3D alignment, usually 10 degrees is fine.
- *weightrecon*, check this box to perform a weighted average using particle population as the weights. Leave this unchecked for the tutorial.

When you have set all the options, click **Launch**.



Project Name: IprR RCT

Workflow Mode: SPR

untiltdata: bdb:./sets#untilted
Num Images: 360

tiltdata: bdb:./sets#tilted
Num Images: 360

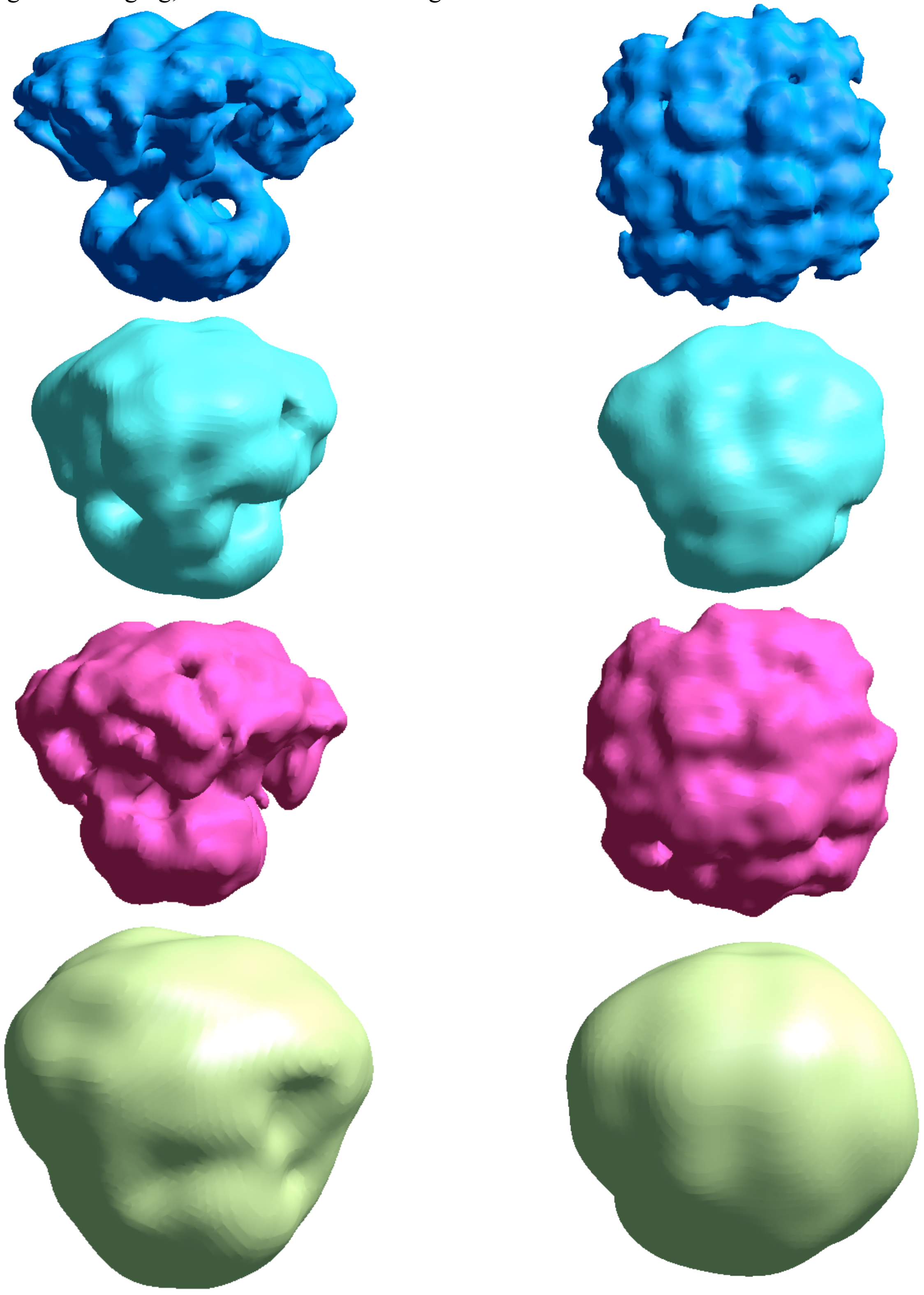
classavg: bdb:./r2d_01#classes_08
Num Images: 7

e2rct options

stagetilt: 60 ☐ tiltaxis

```
File '' from field 'stack_files' does not exist
Program e2buildstacks.py Launched!!!!
Program e2buildstacks.py Launched!!!!
Program e2rct.py Launched!!!!
```

After e2rct.py finishes, open the file browser, by clicking on the folder icon in the projectmanager toolbox. Browse to the rct folder and click on files named ‘rctrecon_??’ These are your RCT reconstructions, one for each class average and rctrecon_00 corresponds to classaverage number zero in the above class averages figure. Shown below is the Ip3R structure and Ip3R RCT reconstructions. Top row, Ip3R structure; second row, RCT recon from class average number zero; third row, RCT recon from class average number two; bottom row, the average RCT reconstruction generated by aligning and averaging each RCT reconstruction. As you can see rctrecon_02 is much better quality than rctrecon_00, and even better quality than the rctaverage. The low rctaverage quality is caused by low quality RCT recons. Hence it is very important to remove any low quality rct recons from the average, but this can be a bit problematic if you are doing a de novo reconstruction where you don't know what the structure looks like in the first place! To attempt a better RCT average, you can export the rctrecons to a 3D stack using e2proc3d.py(see wiki page) and then use e2classaverage3d.py, which is used for subtomogram averaging, to make the RCT average.



That's it. I hope you have enjoyed this tutorial. Email: sludtke@bcm.edu for further information/questions.