Intro

fast_rsm is a feature-rich piece of software that is designed to map raw detector images into reciprocal space. Scattering ***Q***-vectors are individually calculated per pixel per detector image; their corresponding intensities are individually corrected for Lorentz, polarisation, solid angle and transmission errors.

fast_rsm takes, as an input, detector data (images typically in the .tiff or .hdf5 formats) and metadata (containing e.g. motor positions during a scan, which is normally stored in .nxs and/or .dat files).

Setting up SSH connection

Prior to loading and using the fast_rsm package, you will need to setup your SSH connection to the wilson server which is used to submit jobs to the computer cluster.

This is done with the following steps, note that commands to enter are the text after the $ symbol

- log into a workstation with your fedID
- Create a fresh ssh key with the commands:
  - use command $ ssh-keygen
  - will ask to provide a name for the key pair, go with default option id_rsa $ /home/<*fed12345*>/.ssh/id_rsa
  - enter a new passphrase twice, which can be something really simple, you will need to remember it to use it in a few steps time. Note the command line remains empty as you type the password.
- authorise the use of the key by adding it to their trusted keys $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
- alter SSH settings to read-only with the following settings
  - $ chmod 700 ~/.ssh
  - $ chmod 600 ~/.ssh/*
- Connect to wilson $ ssh wilson
- enter ssh passphrase created in earlier step, note this is NOT your fedID password
- if all has gone well then you should see this screen :
-
- Check you can connect without needing password - type $ exit
- Type $ ssh wilson
- This should go straight to the welcome text above, without the need to input a passphrase
- Once this is successfully connecting to the wilson using SSH you are ready to follow the steps in the next 'Using fast_rsm' section
-

Using *fast_rsm*

step-by-step instructions on how to use *fast_rsm* at i07 are detailed below. If you are entirely unfamiliar with linux/computers and you can't follow along, ask a beamline member of staff or email philip.mousley@diamond.ac.uk

Using the command line interface :

1. Login to a diamond linux machine. If you're on the beamline, the beamline scientists will show you which machine uses linux. If you're at home, use nomachine [https://www.diamond.ac.uk/Users/Experiment-at-Diamond/IT-User-Guide/Not-at-DLS/Nomachine.html] to access a linux workstation with access to the diamond servers.
2. type "module load fast_rsm/testing" into the terminal. ***note that currently testing branch has this setup, will upload to main branch once testing example datasets ***
3. type "activate" into the terminal to activate the fast_rsm conda environment
4. make a mask by typing "makemask -dir path/to/experiment/directory -s ##scan##number#" to open up the mask GUI. Save the created mask and note down the file path ###/###/#####.edf
5. make an experiment setup file by typing "makesetup" to open up a template experimental file, edit with your experimental information. this will include

    - *edfmaskfile* = path to the mask you just created.
    - *local_output_path*
    - *local_data_path*
    - *beam_centre*
    - *detector_distance*
    - *process_outputs = specify what data outputs you are wanting to be calculated and saved*

6. Then save this exp_setup.py file noting the path
7. run the processing by typing

"process_scans -exp 'path/to/exp_setup.py' -s scan-numbers-to-be-mapped

e.g.

process_scans -exp **/home/rpy65944/fast_rsm/example_exp_setup.py** -s **441187 441188**

8. alternatively use the -sr option to define an evenly spaced range of scans using the format [start,stop,stepsize]

process_scans -exp **/home/rpy65944/fast_rsm/example_exp_setup.py** -sr **[41187, 441189,1]**

9. use a list of lists to define several sets of evenly spaced scans using the format [[start1,stop1,stepsize1],[start2,stop2,stepsize2]], where the ranges are inclusive i.e. the stop value is the final scan in the range which you want analysed

process_scans -exp **/home/rpy65944/fast_rsm/example_exp_setup.py** -sr **[[41187, 441189,1] ,[41192, 441195,1]]**

Submitted batch job ###### *--> this line is printed if job successfully submitted to cluster*

Job submitted, waiting for SLURM output. Timer=## *--> this line checks for a new SLURM output file every 5 seconds, current time limit of 250 seconds*

Slurm output file: /path/to/home/fast_rsm//slurm-#####.out *--> if new SLURM output file is found within timer limit, output file path*

****
STARTING TO MONITOR TAIL END OF FILE, TO EXIT THIS VIEW PRESS ANY LETTER FOLLOWED BY ENTER* --> monitoring of tail end of slurm out file to monitor progress of calculation
****

Any errors encountered during the processing will saved to the slurmout file and show-up in the terminal where the tail monitoring is taking place

If the tail monitoring shows the line :

PROCESSING FINISHED.

Then the processing has been completed successfully.