Reciprocal space mapping with fast_rsm

Current issues with code - table to keep track of bugs with the fast_rsm code

Example datasets for testing and benchmarking fast_rsm mapper

version history

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 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).

Note that while *fast_rsm* aims to only natively support direct mappings into 3D spaces, it is the aim of *fast_rsm* to support every physically relevant three-dimensional coordinate system.

Q: Why wouldn't you just map directly into Q_{xy}-Q_z / $Q_{||}Q$ / I(*Q*) / I(2) / *I(L)* / insert favourite coordinate system here ?!

A: This would involve significantly expanding the scope of *fast_rsm* – allowing the core of the code to map to lower dimensions would introduce a significant headache with almost no reward. Instead, *fast_rsm* provides some convenience functions that can be used to downsample your 3D volumes into the above-mentioned lower dimensional coordinate systems.

Q: Does this introduce an error?

A: Yes. This error will be larger for smaller reciprocal space volumes. If this error is problematic, consider using the map_per_image option (discussed below) to compute exact reciprocal space maps.

Steps to use 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

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 :

```
[rpy65944@cs03r-sc-serv-46 ~]$ ssh wilson

 _______ __       ______        __   __ _______   _____
|       \ |  \     /      \      |  \ |  \|       \ /     \
| $$$$$$$\| $$     |  $$$$$$\     | $$ | $$| $$$$$$$\|  $$$$$$\
| $$  | $$| $$     | $$___\$$     | $$_| $$| $$__/ $$| $$   \$$
| $$  | $$| $$      \$$    \      | $$    $$| $$    $$| $$
| $$  | $$| $$      _\$$$$$$\     | $$$$$$$$| $$$$$$$ | $$   __
| $$__/ $$| $$_____|  \__| $$     | $$  | $$| $$      | $$__/  \
| $$    $$| $$     \\$$    $$     | $$  | $$| $$       \$$    $$
 \$$$$$$$  \$$$$$$$$ \$$$$$$       \$$   \$$ \$$        \$$$$$$

Welcome To Wilson - The DLS Slurm Cluster
For Help or Support - Visit: https://schelpdesk.diamond.ac.uk

Please Refrain From Running Any Tasks Directly On This Node - It Is A Submission Node Only

For jobs needing to access the GPFS02 filesystem please use the cs04r partition, for the GPFS03 filesy
stem please use the cs05r partition status

To get an interactive session on a GPFS02 node enter "iact02", for a GPFS03 node enter "iact03"
Last login: Tue Jun 27 13:17:12 2023 from i07-ws004.diamond.ac.uk
[rpy65944@cs04r-sc-vserv-300 ~]$
```

-
- 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

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" into the terminal
3. If this is your first time using the fast_rsm mapper, type "firstrun" to setup a folder in your home directory
4. type "activate" into the terminal to activate the fast_rsm conda environment

1. if you need a mask file for your data, 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
2. if you do not have an experiment file, make an experiment setup file by typing "makesetup" to open up a template experimental file, edit with your experimental information. this will include
    a. *edfmaskfile* = path to the mask you just created.
    b. *local_output_path*
    c. *local_data_path*
    d. *beam_centre*
    e. *detector_distance*
    f. *ensure your process_outputs are correct - explained lower down*
3. There is a very important switch that changes the operating mode of fast_rsm: map_per_image. map_per_image defaults to False, in which case all of the scans will be combined into a single volume in reciprocal space. If map_per_image is set to true, then **a very large amount of information will be saved per image**. Generally speaking, if map_per_image is set to true and you are mapping 1GB of images.

   If map_per_image = True, then, for each image in the scans selected, the following will be saved:

   - A binned reciprocal space map, of exactly the same form as those generated when map_per_image is false. Each binned reciprocal space map will have the size set by
   - ***Every q-vector for every pixel in the image.***
   - Every intensity obtained for every pixel in the image, **without** polarisation correction being applied.
   - Every intensity obtained for every pixel in the image, **with** polarisation correction applied.

   Because map_per_image stores every q-vector for every pixel in the image, the complete scattering information is saved. From this, exact maps to e.g. I(Q) and Qxy-Qz can be computed.
4. ## Process_outputs explained

   'full_reciprocal_map'

   calculates a full reciprocal space map combining all scans listed into a single volume. Use this option for scan such as crystal truncation rod scans, fractional order rod scans, or in-plane HK scans,

   'pyfai_qmap'

   calculates 2d q_parallel Vs q_perpendicular plots using pyFAI. Use this options for GIWAXS measurements either with a static detector or a moving detector.

   'pyfai_ivsq'

   calculates 1d Intensity Vs Q using pyFAI. Use this options for GIWAXS measurements either with a static detector or a moving detector.

   DEPRECATED OPTIONS

'pyfai_2dqmap_IvsQ' - use parallel multiprocessing to calculate both 2d Qpara Vs Qperp map, as well as 1d Intensity Vs Q integration - both using pyFAI package

'large_moving_det'

utilise MultiGeometry option in pyFAI for scan with a moving detector and a large number of images (~1000s), outputs: I, Q, two theta, caked image,Q_para Vs Q_perp

'curved_projection_2D' *(**use 'large_moving_det' option instead**)*

this projects a series of detector images into a single 2D image, treating the images as if there were all from a curved detector. This will give a projected 2d image. This should only be used when a detector has been moved on the diffractometer arm during a scan, and the images need to be combined together. *NOTE: this will not work on a continuous scans with ~1000s of images - for these scan types use the 'large_moving_det' option.*

'pyfai_1D' - **(use 'pyfai_2dqmap_IvsQ' option instead)**

Does an azimuthal integration on an image using PONI and MASK settings described in corresponding files. This can handle a short series of images (~50) for individual integrations. If used in combination with 'curved_projection_2D', this will simply integrate the large projected image, and not the series of small images.

'qperp_qpara_map' - **(use 'pyfai_2dqmap_IvsQ' option instead)**

projects GIWAXS image into q_para,q_perp plot. NOTE: Similar to 'curved_projection_2D', this will not work with 1000's of images - for these scans use the 'large_moving_det' option.

Then save this exp_setup.py file noting the path

5. Usually the default calc_setup.py file that contains the calculation settings will be suitable, however if a bespoke calc_setup.py is needed, copy over the calc_setup.py file in the fast_rsm/CLI/i07 folder and edit accordingly. contact beamline staff or i07 data analysis scientist for guidance on this.

1. run the processing by typing "process_scans -exp 'path/to/exp_setup.py' -s scan-numbers-to-be-mapped
    a. e.g. process_scans -exp **/home/rpy65944/fast_rsm/example_exp_setup.py** -s **441187 441188**
2. alternatively use the -sr option to define an evenly spaced range of scans using the format [start,stop,stepsize]
    a. -sr **[41187, 441189,1]**
3. 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
    a. -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

****

get_setups - use this to get setup information used to calculate a hdf5 file. if hdf5 has the configuration saved to config, and it contains the joblines from the analysis job sent to the cluster - then you can type

get_setups -hf path/to/hdf5file.hdf5 -outdir path/to/out/directory

This will open up the hdf5 file, locate the joblines information, and then it will save a copy if this information and open exp_setup.py and calc_setup.py files with the information that was used to send off the analysis job.

1. If you can't get nomachine working, contact the beamline staff/diamond IT support.
2. If module load fast_rsm doesn't work:
    a. If you get "command not found: module", you're probably on your own computer, not the diamond servers!
    b. If you get "Unable to locate a modulefile for 'fast_rsm', there is a serious problem. Contact beamline staff immediately.

3. The slurmout file should contain error messages with information about what error was encountered. If these do not provide an easy solution, forward this onto the i07's data analysis scientist for support

Q: I finished processing, but the output looks ridiculous. What do I do?

There are many reasons why this could happen, but first make sure that you remembered to correctly set your central pixel (this is the easiest thing to forget). Otherwise, either some other piece of information has been entered incorrectly, or your data file is corrupt in some way. In case of the latter, contact the i07's data analysis scientist to investigate. It is usually possible, with some work, to recover your data.
`open up /dls_sw/i07/scripts/gda-zocalo/test_zocalo_functions.py`

this will allow you to access the functions mapstart, currentScan, mapend

To create a map to be sent for processing automatically, the format is as follows

NOTE: the output directory needs to be the processed folder in your experiment directory

```
#define exp_setup and calc_setup files
#enter latest version of example_calc_setup.py - do module load fast_rsm on a new terminal, and see what
version fast_rsm v#### number is loaded
calc_file="/dls_sw/apps/fast_rsm/v0.0.7/fast_rsm/CLI/i07/example_calc_setup.py"

exp_file="path/to/exp_setup.py"


#start of new map
scanlist,ps=mapstart()

#collect scans that make up the map, using append(currentScan) to add scans to list
for i in range (1,4,1):
    scan testMotor1 1 1 1
    scanlist.append(currentscan())

#use mapend function to send off scans. Format   mapend(scanlist,ps,setup_files):
mapend(scanlist,ps,[exp_file,calc_file])
```

You can then collect and process a second map by repeating the code from scanlist,ps=mapstart() onwards.

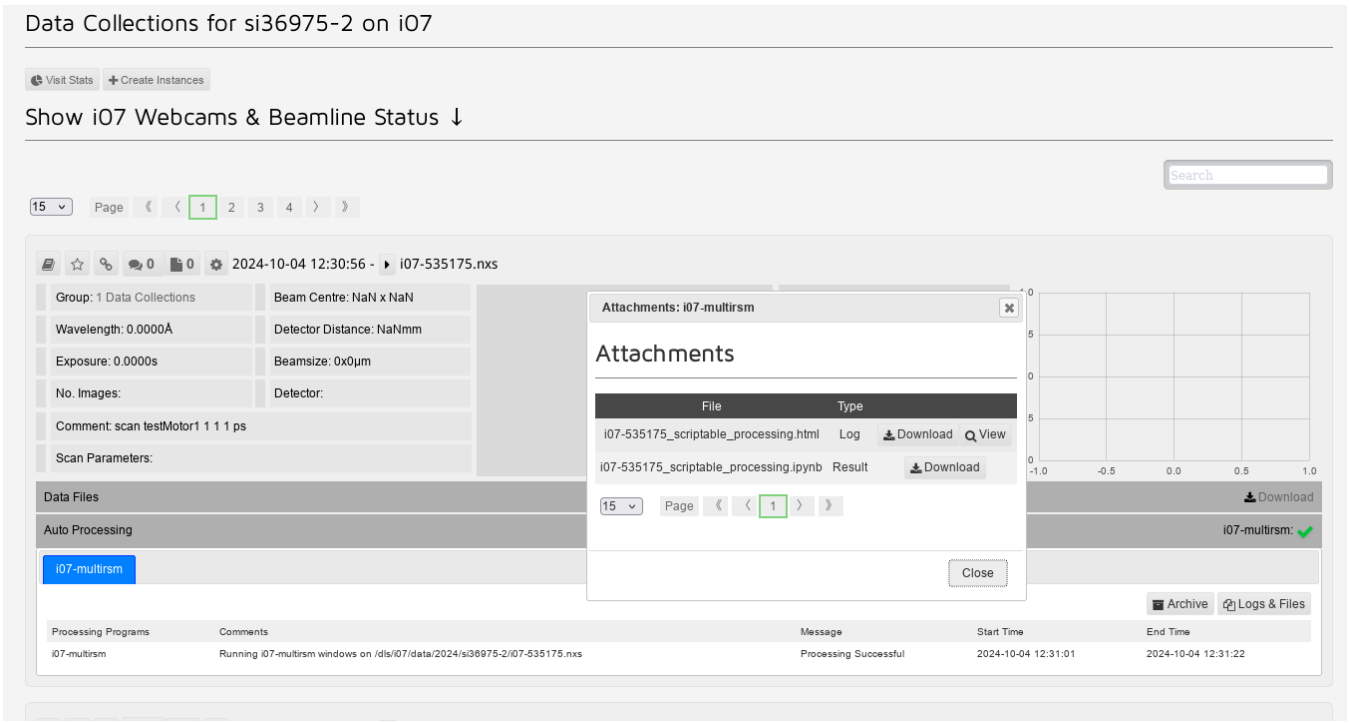
Additionally you can change the exp_file after the first map has finished if a different type of processing is required for the second map

An example macro script is located here /dls_sw/i07/scripts/gda-zocalo/example_gzc_script.py

to monitor progress go to https://ispyb.diamond.ac.uk/dc/visit/si###### using your experiment number

if processing has gone correctly there will be an entry on the webpage with a green tick.

clicking the auto processing line for the entry gives options for 'Logs & Files'. Selecting this opens a window show the extra attachments you can view.



from this you can select the 'View' option on the top line, which will open a scrollable window where you can see the code that was run for the processing, as well as any error messages that were output