X-ray volume scan and centring

Table of Contents

Introduction

The VolumeScan class was generated to support X-ray centring, but (with a few tweaks) the same framework could be adapted for use with Tomogaraphy, optical centring, or even three-click centring if so desired.

Contents

There are two new classes:

- PointCloud. A Datatype containing

- volumeType (str) = “pointCloud”, to distinguish from possible alternative implementations, like triangle mesh
- points: List[Tuple[float, float, float]] a list of 3D points

- VolumeScan(Job): A Volume scan (e.g. X-ray centring) experiment.

Attributes are:

- experimentType (str) E.g. ‘Xray.centring’, ‘Xray.recentring’, ‘Xray.exploration’
  String describing the type and function of the experiment – allowing you to select appropriate defaults. NB, there might at some point be other types than mesh scan, e.g. tomography,, and maybe optics(?)
- rotationAngles (List[float]) A list of axis (omega) values to use for volume scan, in degrees
- searchVolume (PointCloud) The (input) volume to search, in goniostat coordinate system coordinates.
- boundingBoxSize. The size of the bounding box to use, in microscope coordinate system coordinates (horizontal, vertical, beam)
- boundingBoxShape (str) Shape of bounding box, either “Box” or “Ellipsoid”

- resultVolume (PointCloud) The volume of interest found (output), in goniostat coordinate system coordinates. Could contain points for multiple disjoint objects
- subVolumes (List[PointCloud]) List of individual compact subvolumes selected from resultVolume

In addition the VolumeScan would have a single templateData object, a CollectionSweep containing acquisition parameters to use, most importantly the starting motor positions, axisPositionsStart

Doing Mesh Scans

Before we get to use cases we need to look at how mesh scans are specified and executed. The relevant parameters are all in CollectionSweep. There are all the standard ones, (energy, transmission image width, etc.) but I concentrate on those directly relevant for the mesh scan:

- sweepType 'mesh' (or 'line' determines that this is a scan)
- numberImages and numberLines determine the grid pattern
- The motor positions set in axisPositionsStart specify the orientation, and the position at the centre of the mesh (which is the one on beam at these positions).
- meshRange determines the horizontal and vertical extent of the mesh in mm
- scanAxis is set to either 'horizontal' or 'vertical' to determine which dimension is scanned fastest and how the mech is oriented.
- If necessary we could add an extra attribute to determine the zigzag scanning pattern, but that can wait till it is needed.

Setting up a mesh scan the requires setting these parameters in a Collection Sweep, and executing it with an MxExperiment job that specifies the mesh scan.

Scanning and its use cases

The VolumeScan job will work by setting up a series of Mesh scans (there may be other variants later, like tomography?) but for now we only look at this). The inputs come from the VolumeScan attributes and the template CollectionSweep attached to the VolumeScan. I can see three operation modes. For all of them the number of meshes and their orientation is determined, either from the VolumeScan.rotationAngles attribute, of by defaults selected using to the VolumeScan.experimentType.

Starting from a bounding box.

If you simply have a rough loop orientation rather than a proper point cloud, you would specify only a bounding box. This is done by first setting the axisPositionsStart so that the desired volume is oriented along the horizontal/vertical/beam axes and centred on the beam position. The boundingBoxSize and bondingBoxShape then determine the relevant volume, which can be used to define the axisPositionsStart, meshRanges and other mesh control parameters that specify the individual mesh scan for each rotationAngle. Note that not all box-shaped volumes can be aligned so as to have the axes parallel with the horizontal/vertical/beam axes. If this is your case you must specify your input as a point cloud, or first convert the actual volume of interest to a larger box that can be aligned appropriately.

Starting from a point cloud

If you have your volume of interest as a point cloud, you pass that as input to the VolumeScan as the searchVolume. This can then be used to calculate the axisPositionStart and a boundingBox (in the appropriate coordinate system). Alternatively you can skip this step and have your algorithm determine the individual mesh scans directly from the point cloud.

Starting from a point – recentring

If you are simply doing 'touch-up' recentring all you want to do is to pass in the desired orientation and calculated motor positions for the centring. For that you need only to set the axisPostionsStart values to the orientation and centring you want, set experimentType to 'Xray.recentring' and then execute the VolumeScan. The bounding box will then be set to a suitable set of default parameters, and the mesh scans calculated from there.

Output

Output comes as a resultVolume, which is a (probably disjoint) pointCloud with all measured points, and a list of subVolumes, which is a filtered list of (presumably compact) objects found. For any practical use you would presumably need to work off the subVolumes, so you need a routine to select them, filter them, and a way to decide which one(s) to use if there is more than one. For (re)centring, particularly, you need to decide which point you are recentring, and you need to convert from the point cloud of that subvolume , first to get a single centroid point, and then to convert the goniostat coordinate system coordinates of that point (together with the axisPositionsStart of the experiment) to a set of motor positions to use.