Background:

STXM (Scanning transmission X-ray microscopy) is an X-ray microscopy technique, where a monochromatic X-ray beam is focused onto a spot of a sample, and the transmitted X-ray intensity is recorded with a suitable detector.
By raster scanning the sample and recording the transmitted intensity at each point of the scan, a STXM image is then formed. Although the transmission can be measured this way with a scalar detector (giving a single value at each position of the sample), it is more common to measure the distribution of the transmitted beam with an area (2D) detector.

Scenario:

A new STXM experimental endstation has been installed at one of the DESY beamlines.
The endstation is equipped with an area (2D) detector. The sample is mounted on a high-precision XY piezoelectric stage. The programmable controller of the stage is capable of producing triggers based on the motor position to trigger the detector.

This setup allows the user to set up and run very fast 2D scans (fly-scans), with synchronization between motor positions and detector readouts being handled by hardware.

At the beginning of each scan, the experiment control software emits scan metadata via 0MQ. This includes start and stop positions and step size for each of the X and Y motors, as well as exposure time at each position and detector frame shape.
The detector, in addition to file-writing, is also streaming recorded data via 0MQ.

The scientists would like to have a live view of the detector frames as well as a live STXM image, with STXM signal intensity at each given X/Y position being a sum of intensities of all pixels of the detector frame.

What to do:

Using the supplied python script that simulates (random) data and metadata stream from the endstation, build a python-based visualization application with two views: one for frame data and one for STXM data.
At minimum, the end user should be able to run the application and start seeing the data live as soon as the scan starts.
If you are up for a bit of a challenge – feel free to add in more functionality, that you may see useful.

Consider the facts, that:
- detector frames could be as large as 4000 x 4000 pixels;
- data stream rate could be as high as hundreds of frames-per-second;
- scientists may want to run a visualization app on a low-performance office-grade laptop.

You are free to use any tools you wish. If you aren't sure where to start, we would recommend using *pyqtgraph* – it comes with a variety of examples.

If you have any questions about the assignment, please feel free to reach out to our Software Engineer Lisa Dorofeeva at elizaveta.dorofeeva@desy.de