




CrystalGrowthTracker: A Python package for calculating the face advancement rates of crystals from videos taken using synchrotron radiation

Joanna Leng^{*13}, Jonathan H. Pickering^{†23}, Sven L. M. Schroeder⁴⁵⁶⁷, and Gunjen Das⁴⁵⁶⁷

¹ EPSRC Research Fellow in Software Engineering ² Research Fellow in Software Engineering ³ School of Computer Science, University of Leeds, Leeds, LS2 9JT, UK ⁴ School of Chemical and Process Engineering, University of Leeds, LS2 9JT, UK ⁵ Diamond Light Source Ltd, Oxfordshire, OX11 0DE, UK ⁶ Research Complex at Harwell, Oxfordshire, OX11 0FA, UK ⁷ EPSRC Centre for Innovative Manufacturing in Continuous Manufacturing and Advanced Crystallisation, University of Strathclyde, G1 1RD, UK

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Lucy Whalley](#) 

Submitted: 28 January 2022

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

X-ray synchrotron radiation allows the investigation of many physical processes in unprecedented ways, one application is early stages of crystallization important to the world's fine chemicals industry. To aid chemical engineers working in this area we developed the CrystalGrowthTracker package. It allows crystals to be found in videos and their growth rates measured, we hope that this work can provide the basis for further fully automated systems.

Statement of Need

Much of the output of global fine chemicals industries consists of crystalline powders produced by precipitation from solution. Since the geometry of a crystal is the result of differential growth rates on different crystal planes, there is great commercial interest in studying the growth rates of crystal faces in the early stages of crystallization. One approach is to use the X-rays of synchrotron radiation ([Baruchel et al., 2013](#)) to produce shadow graphs of crystals precipitating onto a substrate. Videos of the shadow graphs first need enhancement using the packages such as Euler's Magnifier ([Wu et al., 2012](#)) and some statistical analysis of the raw video is desirable to find regions of interest. **CrystalGrowthTracker** has been developed to assist in the analysis of videos of the raw and enhanced videos.

Design

Although the obvious approach would be to use image analysis and machine learning, the relatively noisy data, the limited number of data sets and the need for verifiable results lead to a manual approach. User analysis of video required the package to be based on a graphical user interface (GUI). Finally the users would need to be able to download and run the package on any machine, so the package was developed using Python and PyQt5 ([riverbank?](#)) withing the conda package management system. The use of Qt naturally leads to an object orientated architecture.

*co-first author

†co-first author

36 The package must allow the user to load two videos, the raw and the enhanced and had to
37 provide the following functions.

- 38 1. Analyse and display intensity statistics of the raw video, to assist selection of times of
39 interest.
- 40 2. Play enhanced video and allow the selection of regions of interest.
- 41 3. In regions of interest and allow features to be marked up on a given frame. The markers
42 being, lines for crystal faces, or points for other features.
- 43 4. Change the frame of a region of interest and move the markers to follow the underlying
44 feature.
- 45 5. Use time and space calibration data to calculate the true speed of the marker motions.
- 46 6. Store the above data in an open human readable format and present a report in HTML.

47 Functionalities

48 A model-view-controller (MVC) software architecture was used, in which the view and the
49 controller were merged into the main widget. The video was accessed via a VideoSource
50 object, which was held by the main window, together with project data, and results objects.
51 These objects constitute the MVC architecture model and were made available to the widgets
52 carrying out user functions by getter methods. These widgets were themselves held by the
53 main window in a tab widget.

54 The VideoSource object accessed the video via the ffmpeg package, which was encapsulated
55 in a using the subprocess module. The frame access had to be in terms of time rather than
56 frame number as when accessing a specified frame number ffmpeg scans the entire video to
57 count frames from the start. The widgets requiring video stored their own time in the video
58 and when playing called for the frame at the next time step. No effort was made to play at
59 the correct number of frames per second.

60 Project data and results are saved in comma separated value (CSV) files handled by an input-
61 output module using the Python csv module. The results of a project can be saved in a HTML
62 report, which can be viewed using the report tab widget. The HTML was written by a report
63 writer module using offscreen rendering for the associated graphics.

64 The video is displayed using a QGraphicsView with an associated QGraphicsScene, which
65 together provide a scene-graph and view. Video is displayed as a QPixmap at depth zero in the
66 scene, and the user markings are QGraphicsItems at depth one (above the pixmap). A zoom
67 feature is provided by scaling the view matrix in the QGraphicsView.

68 The user was able to add markers to image features in a single frame of the video, then advance
69 to a new frame and drag clones of the markers to the features new locations. The markers
70 were graphics items augmented to hold their frame numbers and the identity of their parent.
71 Calculating the speeds of motion consisted of ordering the chains of cloned graphics items by
72 frame number, finding the displacements in pixel coordinates, converting the pixel distances to
73 real distances using the scaling factor. The pixels are assumed to be square as individual x-ray
74 sensors in the array were square. Frame number intervals are converted to times and speeds
75 calculated.

76 Testing, Documentation and Linting

77 A test suite is provided, that utilizes the Python unittest module. The Qt QTest object is
78 used to simulate IO events, and the capture the resultant signals. The tests can be run using
79 unittest from the command line or all tests can execute using the run_test tool, which can
80 also save the results to a CSV file. A separate test was developed for the VideoSource as calls
81 to subprocess cannot be tested in unittest.

82 Doxygen was used to generate documentation from source code comments. The code was
83 developed using the Pylint static code analysis tool, for which a runner script was developed.
84 The script runs Pylint on all files, with the output displayed in the shell tool window or saved
85 to CSV file.

86 Availability

87 The software can be obtained from [GitHub](#) under an Apache License.

88 Acknowledgement

89 This work was funded by Joanna Leng's EPSRC funded RSE Fellowship (EP/R025819/1).

90 Baruchel, J., Di Michiel, M., Lafford, T., Lhuissier, P., Meyssonier, J., Nguyen-Thi, H., Philip,
91 A., Pernot, P., Salvo, L., & Scheel, M. (2013). Synchrotron x-ray imaging for crystal
92 growth studies. *Comptes Rendus Physique*, 14, 208–220. [https://doi.org/10.1016/j.crhy.](https://doi.org/10.1016/j.crhy.2012.10.010)
93 [2012.10.010](https://doi.org/10.1016/j.crhy.2012.10.010)

94 Wu, H.-Y., Rubinstein, M., Shih, E., Gutttag, J., Durand, F., & Freeman, W. T. (2012).
95 Eulerian video magnification for revealing subtle changes in the world. *ACM Transactions*
96 *on Graphics (Proc. SIGGRAPH 2012)*, 31(4).

DRAFT