EPICS Collaboration Mtg.
APS, Argonne/IL
June. 15, 2018

On-axis 3D Microscope for X-ray Beamlines at NSLS-II

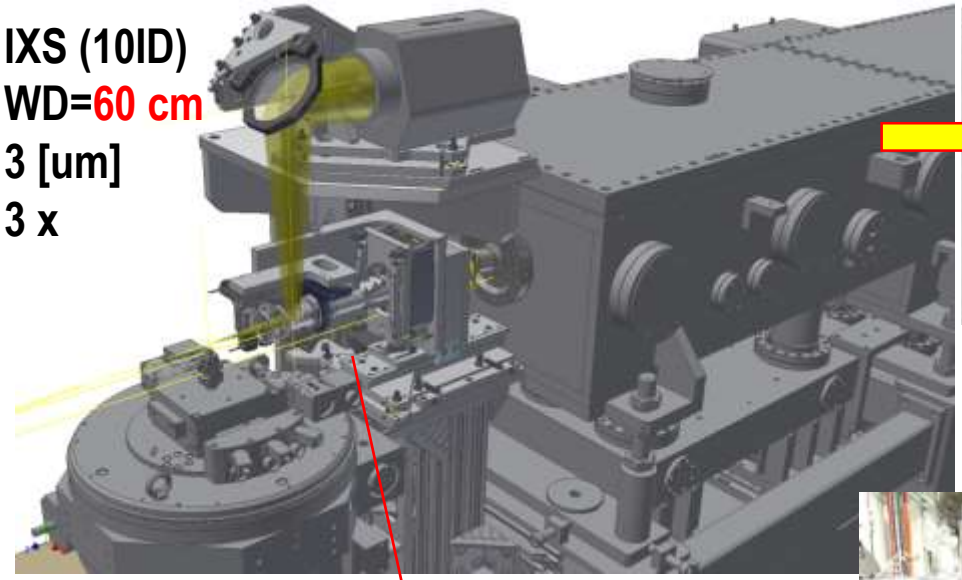
- ***Kazimierz J. Gofron*** *(kgofron@bnl.gov), NSLS-II*

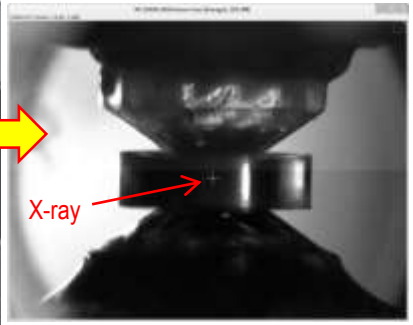
Outline

- Overview –NSLS2
  - Beamlines (26 operational + 3 under constr.)
  - End stations
  - Beam size(s) @ sample 1 [um], 4 [um], 4x6[um]
  - Sample/features: 1 [um]
- On-axis microscope(s): Non-dispersive (mirror) optics
- Computer Vision
  - Goniostat centering (17-ID, 19-ID)
  - X-ray beam detection, stability studies
  - Real time image processing
  - Focus stacking
  - 3D microscope
  - AD barcode plugin
- Credits

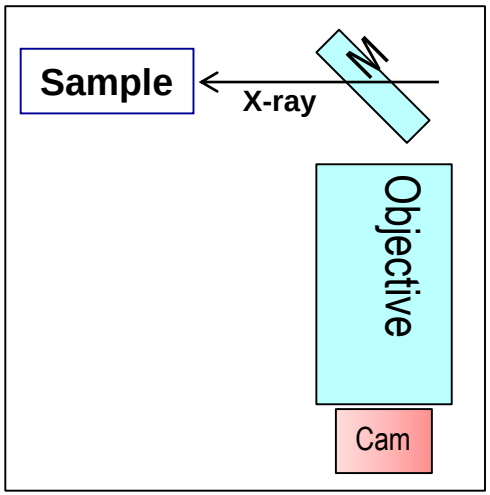
On-axis beamline microscopes

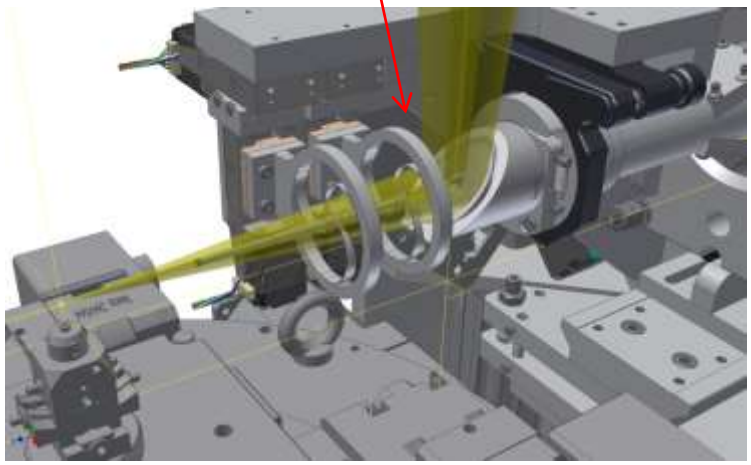
IXS (10ID)
WD=60 cm
3 [um]
3 x





Diamond cell (IXS)

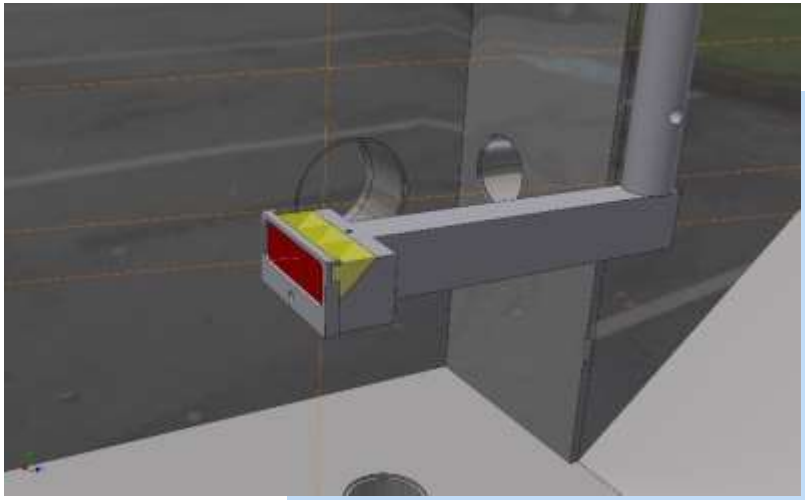




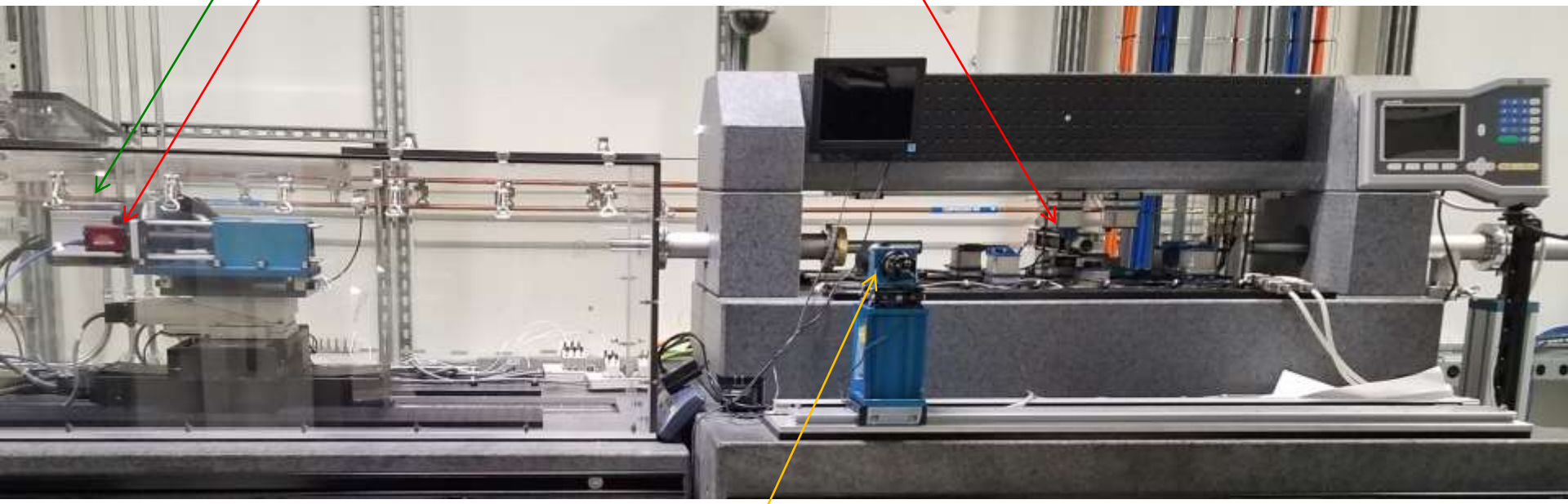
K.J. Gofron, et. al., NIMA, Vol. 649, Issue 1, 1 September 2011, Pages 109-111.

AMX/FMX
(17ID)
30x,5x
1 [um]

Endstation microscopes (IXS)



Endstation microscopes (TXM)

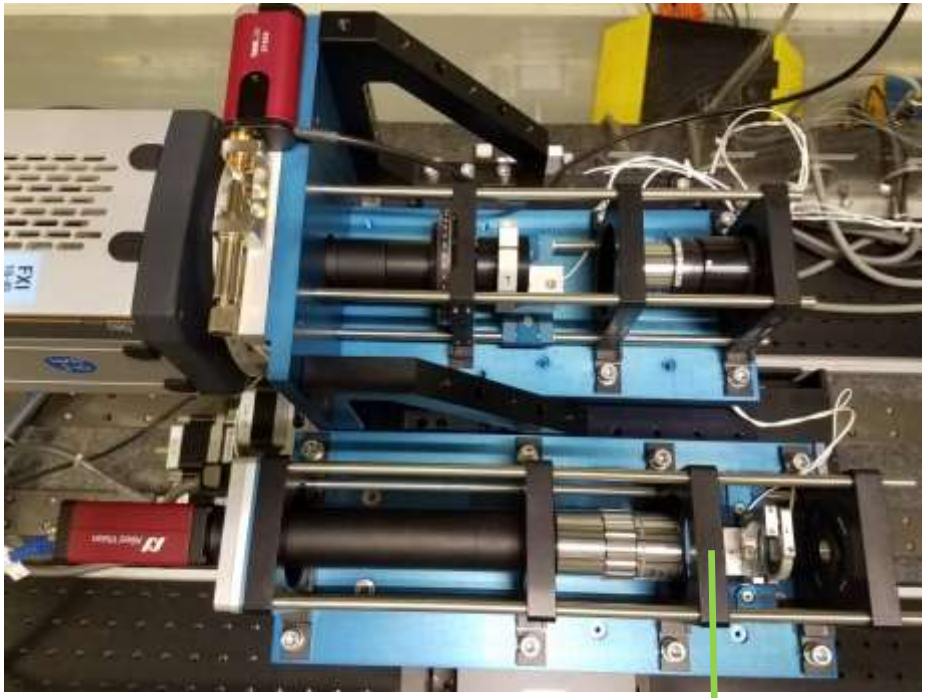


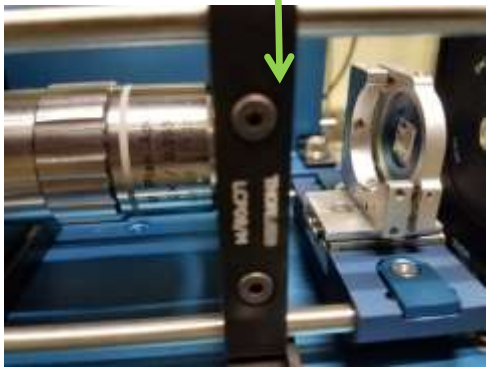
2 X-ray detecting microscopes

2 optical alignment microscopes

All are home assembled microscopes

Endstation microscopes (FXI)

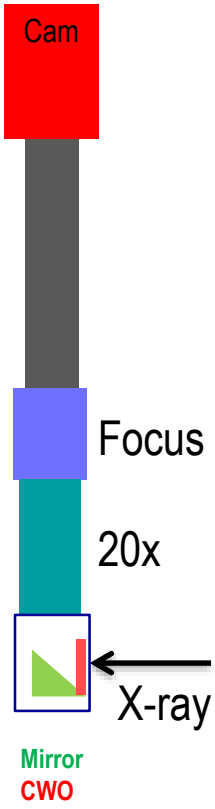


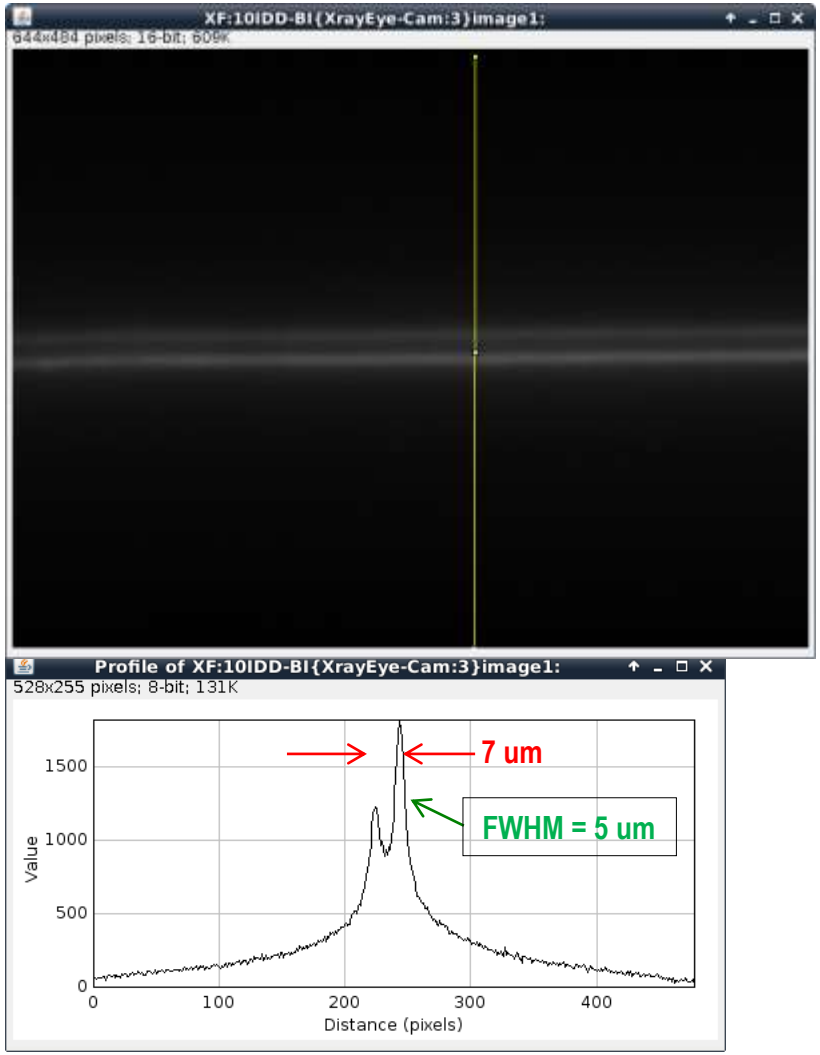


X-ray imaging , TXM

Sample imaging

X-ray eye: 10ID (KB mirror - VFM)





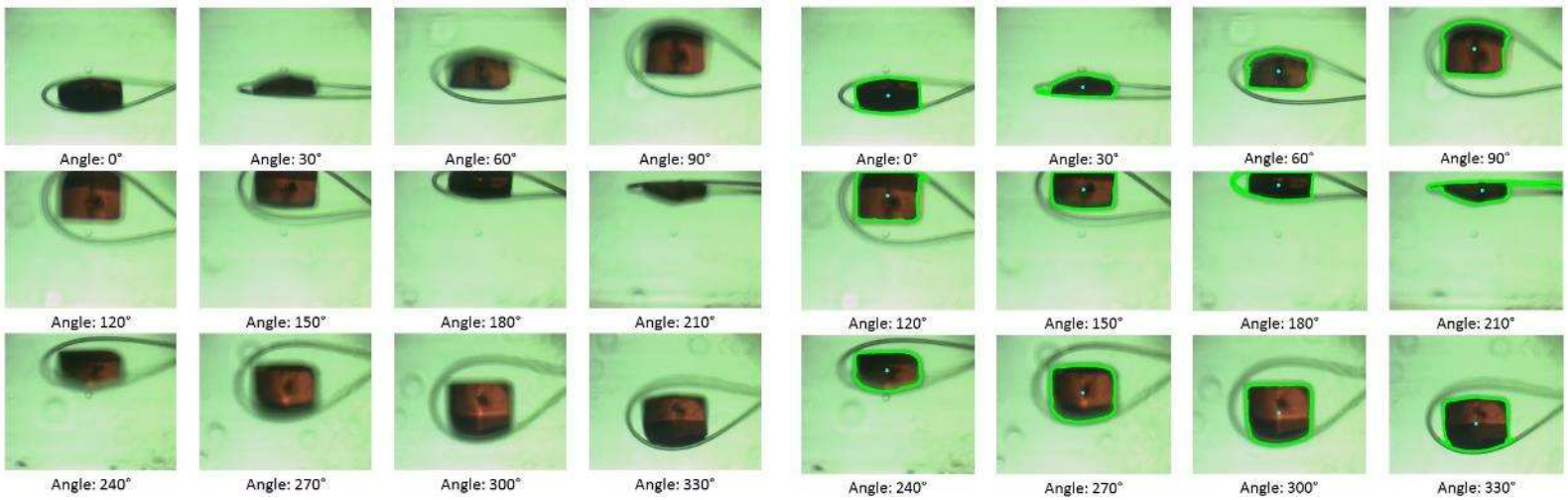
KB mirror (VFM)

Computer Vision

Computer Vision for beamline with openCV:

- Development of image analysis software backed by OpenCV
- Development of easy use python module to access OpenCV functions
- Optimized results for fast computation via C/C++ backed code, along with Intel IPP/TBB libraries.
- GPU role.
- Camera -> areaDetector { | IP based} -> Comp. Vision -> Results {sorted size, intensity, position, pattern,…} -> EPICS PV { | file | control software | …}
- Automate processes such as:
    - Positon, spread, and intensity of X-Ray Beams
    - Isolate crystals and X-Ray streaks
    - Provide assistance to sample mounting
    - Calibrate Goniostat Rotation and Robotic Vision
- 3D microscope

Computer Vision – loop center (5s)



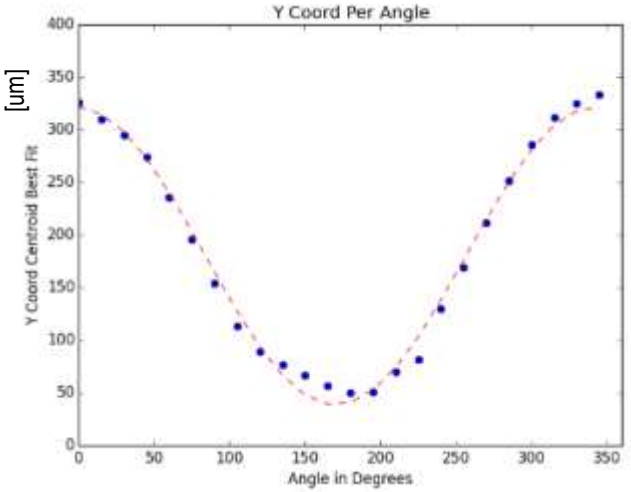


Figure 10:
Top: Input and Tracking Results
Left: Graph Produced, along with fitted sinusoidal curve

Equation:

$$141.58 \times \sin(angle + 1.61) + 180.19$$

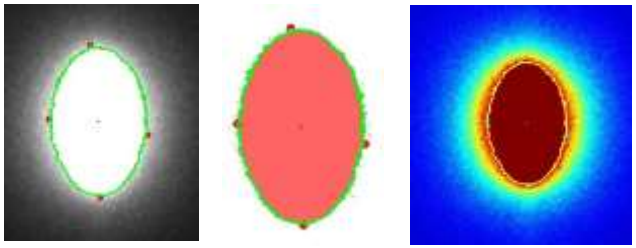
Adjustment:

$$x = -\frac{MC}{PEL} \times Amplitude \times \sin(phase)$$

$$y = -\frac{MC}{PEL} \times Amplitude \times \cos(phase)$$

Computer Vision

IXS – BPM1

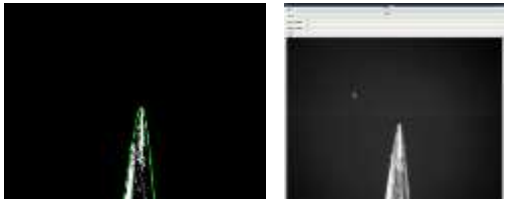


Console Output:

Object Details:

```
perimeter: 2356.99022925
orientation: 179.838363647
max: (925, 198)
height: 372
extrema: {'B': (938, 568),
          'R': (1054, 415),
          'L': (813, 377),
          'T': (914, 196)}
```

```
area: 65058.5
min: (1047, 564)
sum intensity: 20426526
width: 241
centroid: (933, 382)
mean intensity: 227.842390577
```



IXS Point & Click: Left: Image Result with Contour and Top Extrema
Right: Point & Click GUI Interface (X, Y, MC Scale Bars, Green Cursor)
Console Output: `PIN TOP: (1723, 1306)`

IXS: Merlin Data Results for First (Largest) Object

Console Output:

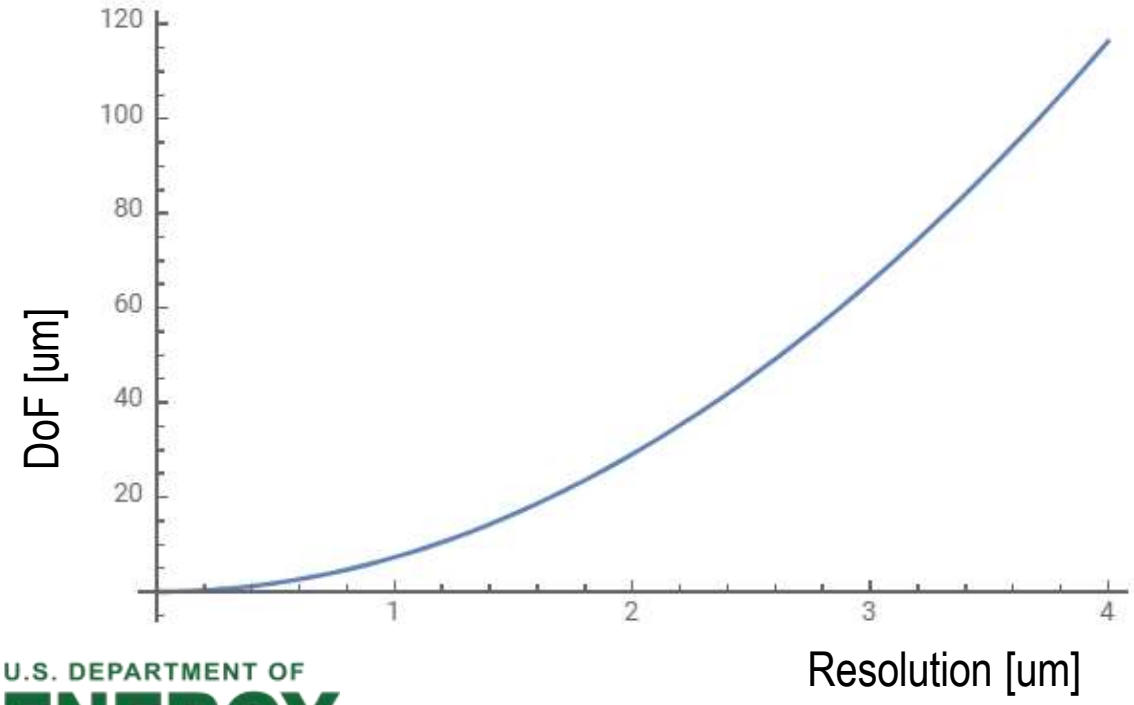
Object 1:

```
perimeter: 125.840619564
orientation: 179.981033325
max: (131, 78)
height: 55
extrema: {'B': (129, 122), 'R': (135, 98),
          'L': (126, 92), 'T': (132, 67)}
area: 270.5
min: (134, 83)
sum intensity: 62689
width: 9
centroid: (130, 95)
mean intensity: 126.644444444
```

AMX: Robot gripper – sample detection

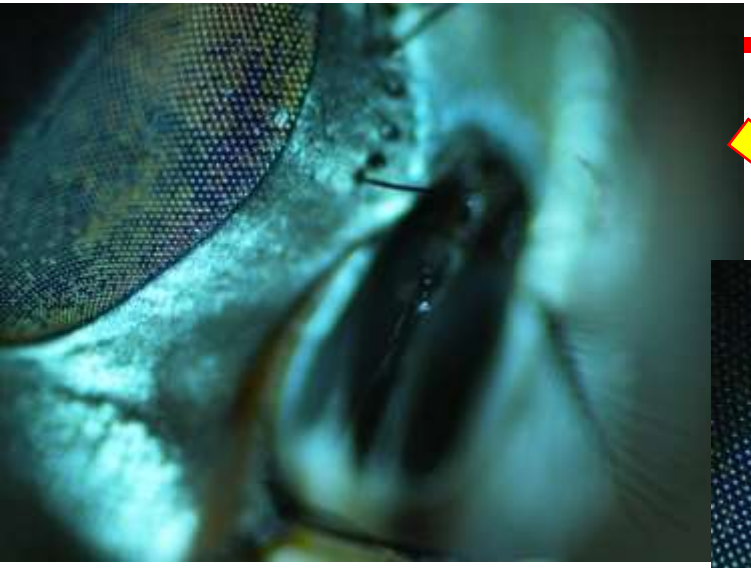
Resolution and Depth of Focus

- Resolution R = wavelength / (2*NA)
- Depth of field = wavelength*n/(NA)^2; n=1 for air
- Depth of field = 4*n*R^2/(wavelength)



Shallow DoF is a Weakness

Focus – Horse Fly



?

How it Works

- To begin, a set of images is taken by a shallow depth of field camera, all of the same sample.
- The images must then be aligned, in order for the merge to create a seamless image.
- A Euclidean transform was the alignment method used, after Oriented-Fast and Rotated-Brief proved ineffective when working with blurry images.

How it Works

- Next, each image is blurred with a Gaussian blurring filter, to further emphasize the in-focus regions of the image
- Finally, a Laplacian kernel is run across each image, assigning a sharpness value to each pixel.

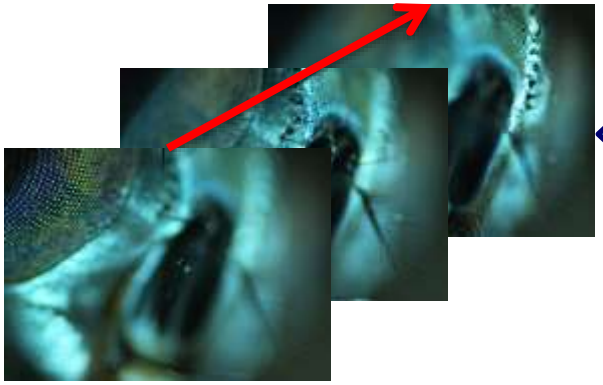
How it Works

- Finally, we iterate over each pixel choosing the one from the image with the highest “sharpness” assigned to it
- Additionally, during this step, we take note of which image provided each pixel, effectively creating a depth map of the image.

How it Works

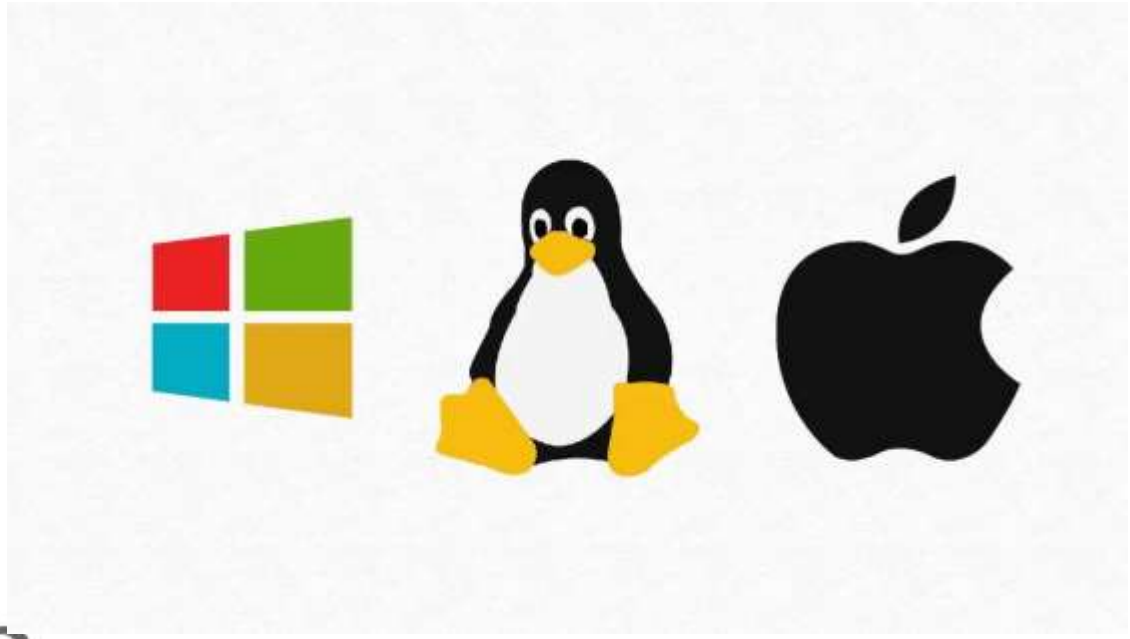
- Finally, using the depth map, a 3D model is constructed by a custom OpenGL pipeline, and the merged image is placed over this model.
- The 3D model, once rendered is rotated, and made into a video for easier evaluation.
- Additionally, using the EPICS python wrapper, automated image stack collection was implemented for supported cameras.

Focus Stacking – Horse Fly (5x)

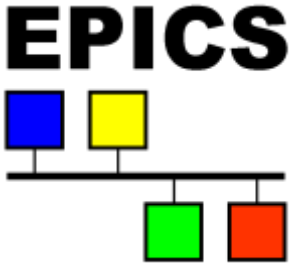


Original 38 Image Stack

The code



Real Time



Key Points

- Code portability was important: Python chosen as coding language because of this and because of ability to call efficient C and C++ functions through wrappers
- OpenCV and OpenGL used for Computer Vision and Graphics respectively due to them being powerful, open source, and compatible with all major operating systems.
- Python wrappers for OpenCV and OpenGL used to improve performance over native Python code

Beettle (10ID, 3x/3um)

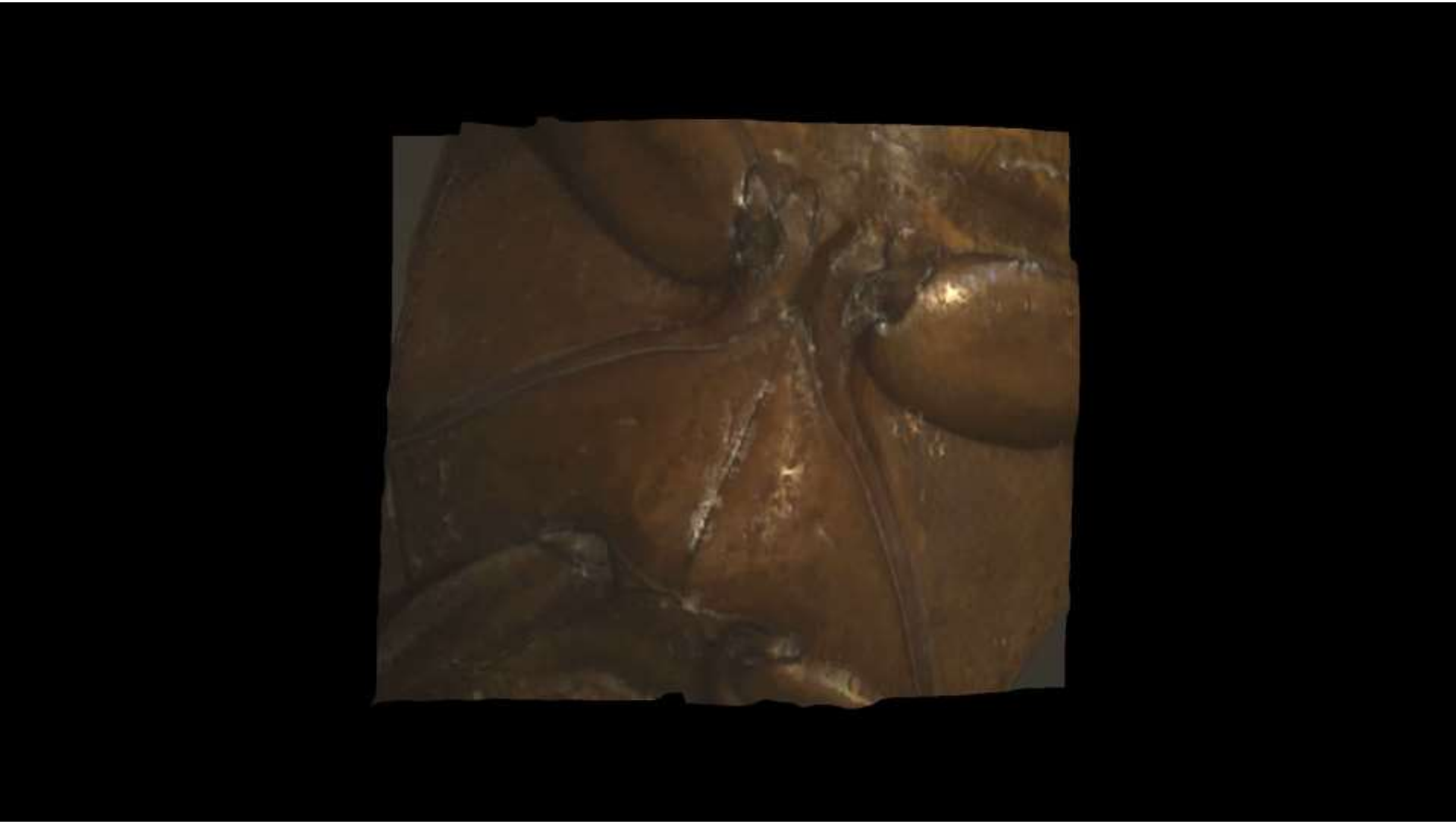
Two examples out of a stack of 19 images. First we performed focus stacking and gamma correction. Images were taken at the IXS 10ID beamline.
https://www.youtube.com/channel/UC-SfBpwDIiuw41_r0qqYkZQ?view_as=subscriber

The Beetle – merged/gamma corrected (3x)

10ID

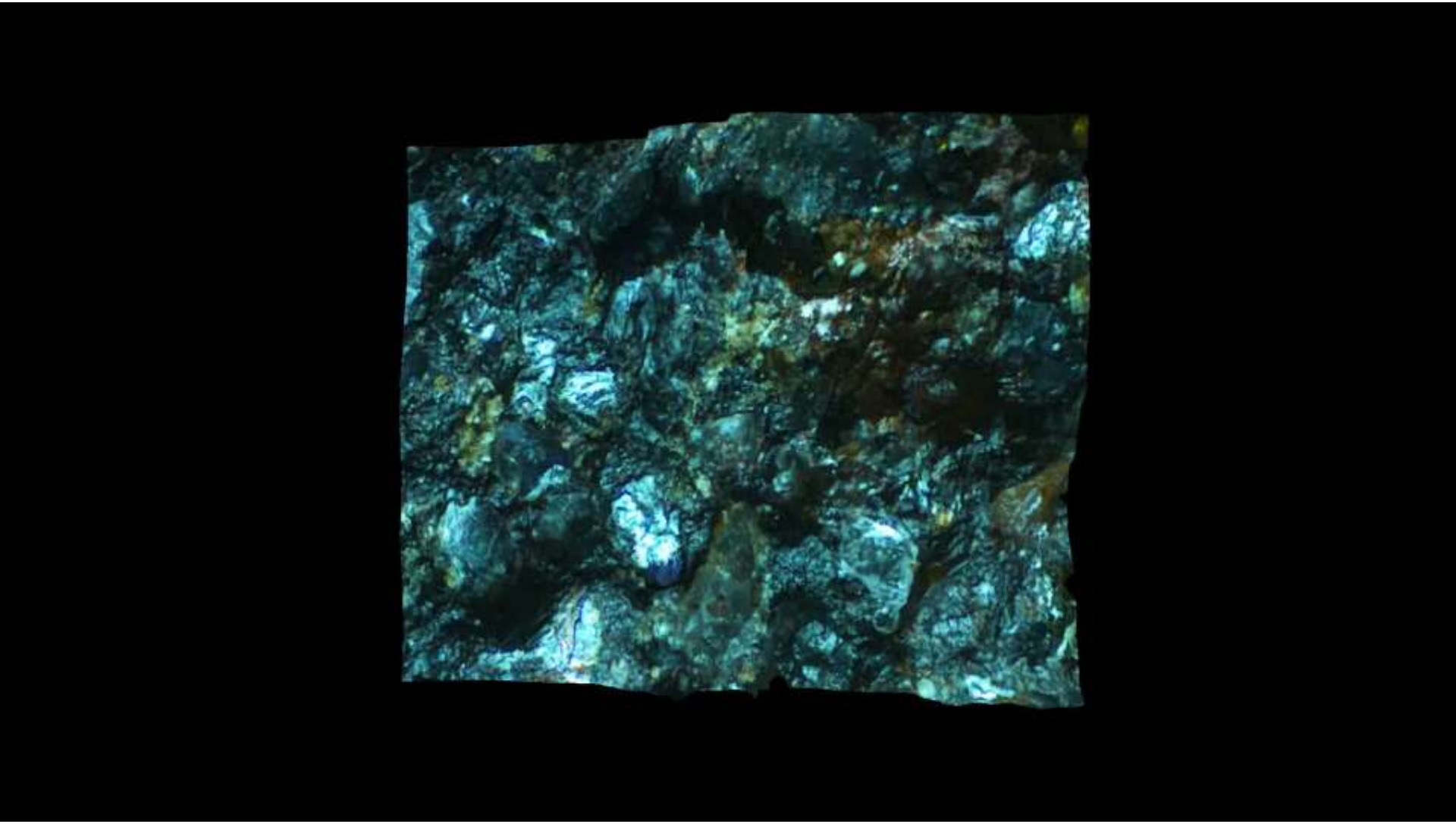
The Beetle – depth map

The Beetle: focus=3rd dimension



https://www.youtube.com/watch?v=UFx2EDouO-k

Mineral



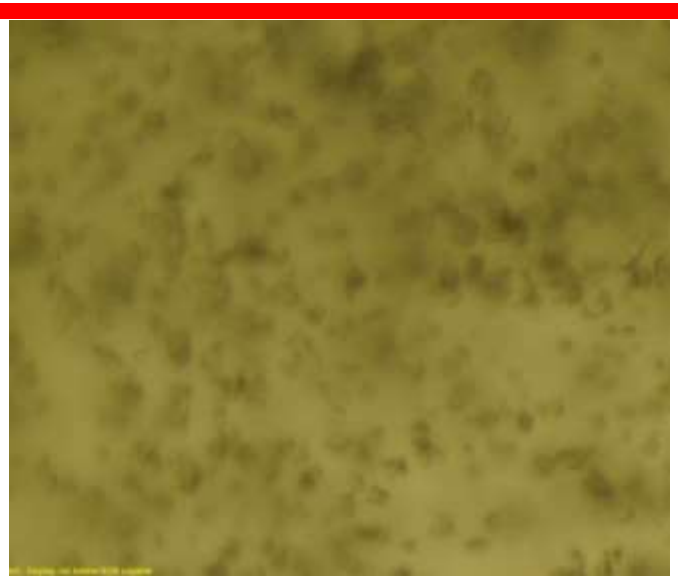
https://www.youtube.com/watch?v=FdpdAdoirwA

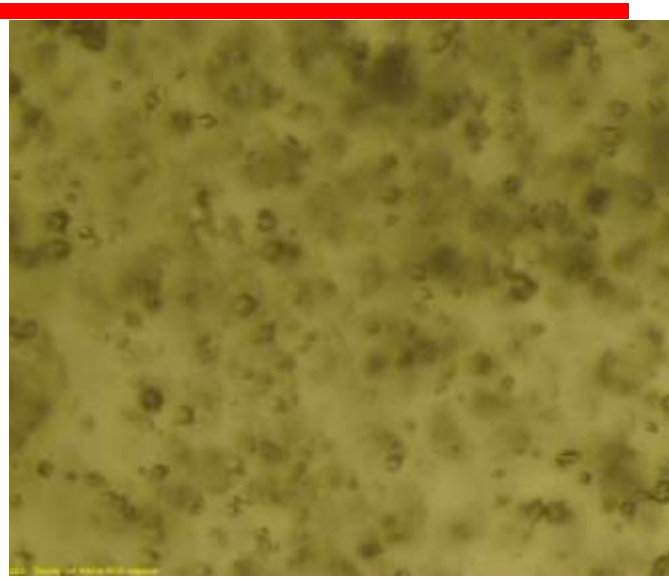
Crystal harvesting – sonic ejection

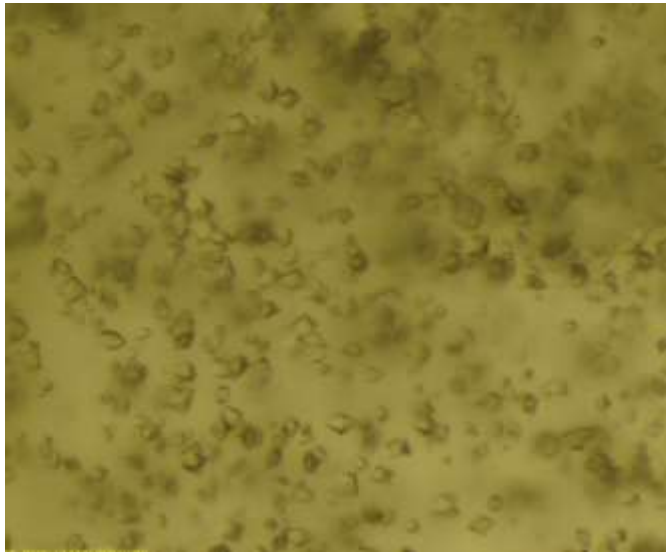
Crystal images are all courtesy of Alexei Soares and the Click to Mount Team. Focus stacks of 15 images.

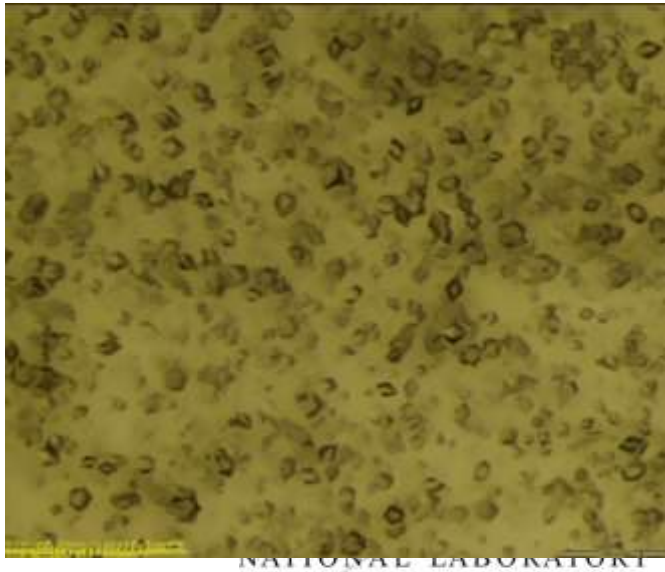
Crystal are sonicly ejected only from top layer, and surface layer moves (with crystals) to “heal” the surface.

Crystals from lower layers are not harvested.

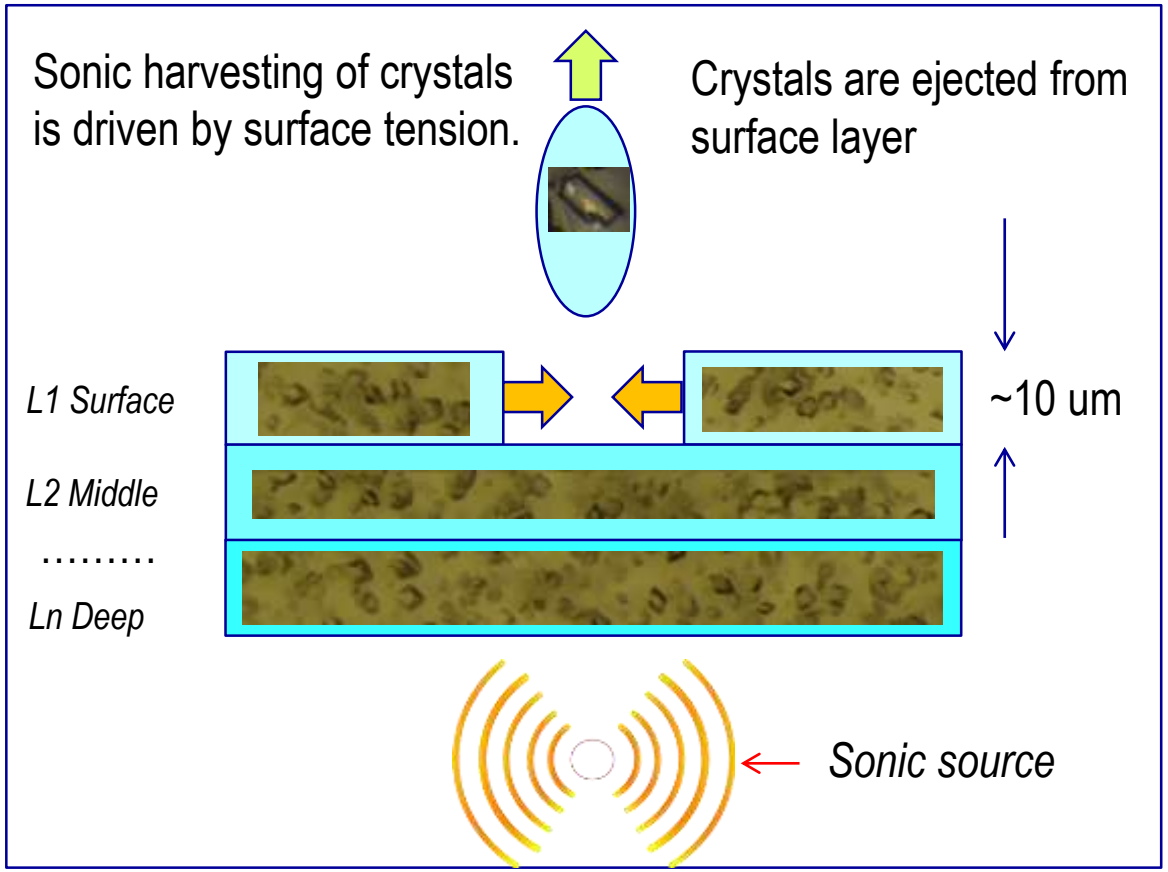






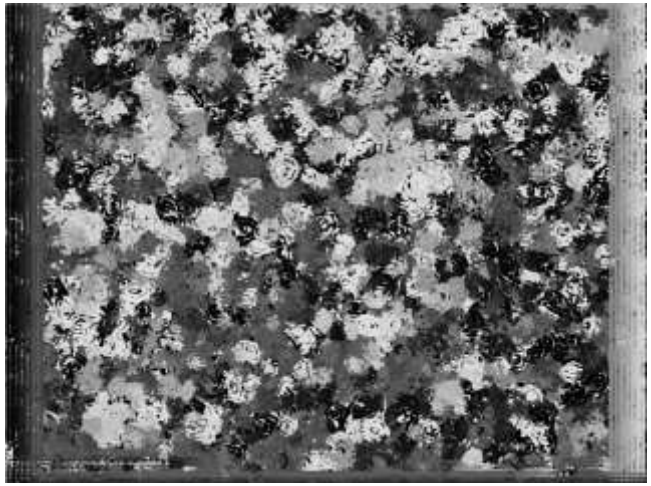


Crystal harvesting



Conclusion 1:
Prevent crystals from settling to the bottom. Thus consider using Bingham fluid solutions.
(to be published)

Depth map



SUMMARY: **Converted weakness (shallow DoF) into strength (recovered 3rd depth dimension)**

AD Plugin: Barcode reader

```
jwlodek@debian:~/Documents/barcodeproject$ ./check
Type : QR-Code
Data : xf10id -barcode test
```

```
/*
Function that does the barcode decoding. It is passed an image and a vector
that will store all of the codes found in the image. The image is converted to gray,
and a zbar scanner is initialized. The image is changed from an opencv to a Image object,
and then it is scanned by zbar. We then iterate over the discovered symbols in the image, and
create a instance of the struct. the struct is added to the vector, and the bars location
data and type are stored, and printed.
 */
static void decode_bar_code(Mat &im, vector<bar_QR_code> &codes_in_image){
  ImageScanner zbarScanner;
  zbarScanner.set_config(ZBAR_NONE, ZBAR_CFG_ENABLE,1);
  Mat imGray;
  cvtColor(im, imGray, CV_BGR2GRAY);
  Image image(im.cols, im.rows, "Y800", (uchar*) imGray.data, im.cols *im.rows);
  int n = zbarScanner.scan(image);

  for(Image::SymbolIterator symbol = image.symbol_begin(); symbol!=image.symbol_end();++symbol){
    bar_QR_code barQR;
    barQR.type = symbol->get_type_name();
    barQR.data = symbol->get_data();

    cout << "Type : " << barQR.type << endl;
    cout << "Data : " << barQR.data << endl;
    setStringParam(NDPluginBarBarcodeType, barQR.type);
    setStringParam(NDPluginBarBarcodeMessage, barQR.data);
    setIntegerParam(NDPluginBarBarcodeFound, 1);
    for(int i = 0; i< symbol->get_location_size(); i++){
      barQR.position.push_back(Point(symbol->get_location_x(i), symbol->get_location_y(i)));
    }
    codes_in_image.push_back(barQR);
  }
}
```

Enabling technology

Open source

- openCV
- zbar

References

Moeller, Michael, et al. “Variational Depth From Focus Reconstruction.” *IEEE Transactions on Image Processing*, vol. 24, no. 12, 2015, pp. 5369–5378.,doi:10.1109/tip.2015.2479469.

“OpenCV Library.” *OpenCV Library*, opencv.org/.

Group, Khronos. “The Industry's Foundation for High Performance Graphics.” *OpenGL.org*, www.opengl.org/.

Credits

Jakub Wlodek (focus stack)
Yong Cai (10ID)
Bill Watson (comp. vision)
Scott Coburn (mechanical)
Stephen Antonnelli (mechanical)
Bruno Martins
Martin Fuchs (17ID)
Jean Jakoncic (17ID)

Work at Brookhaven was supported by the Department of Energy, Office of Basic Energy Sciences under contract DE-AC-02-98CH10886