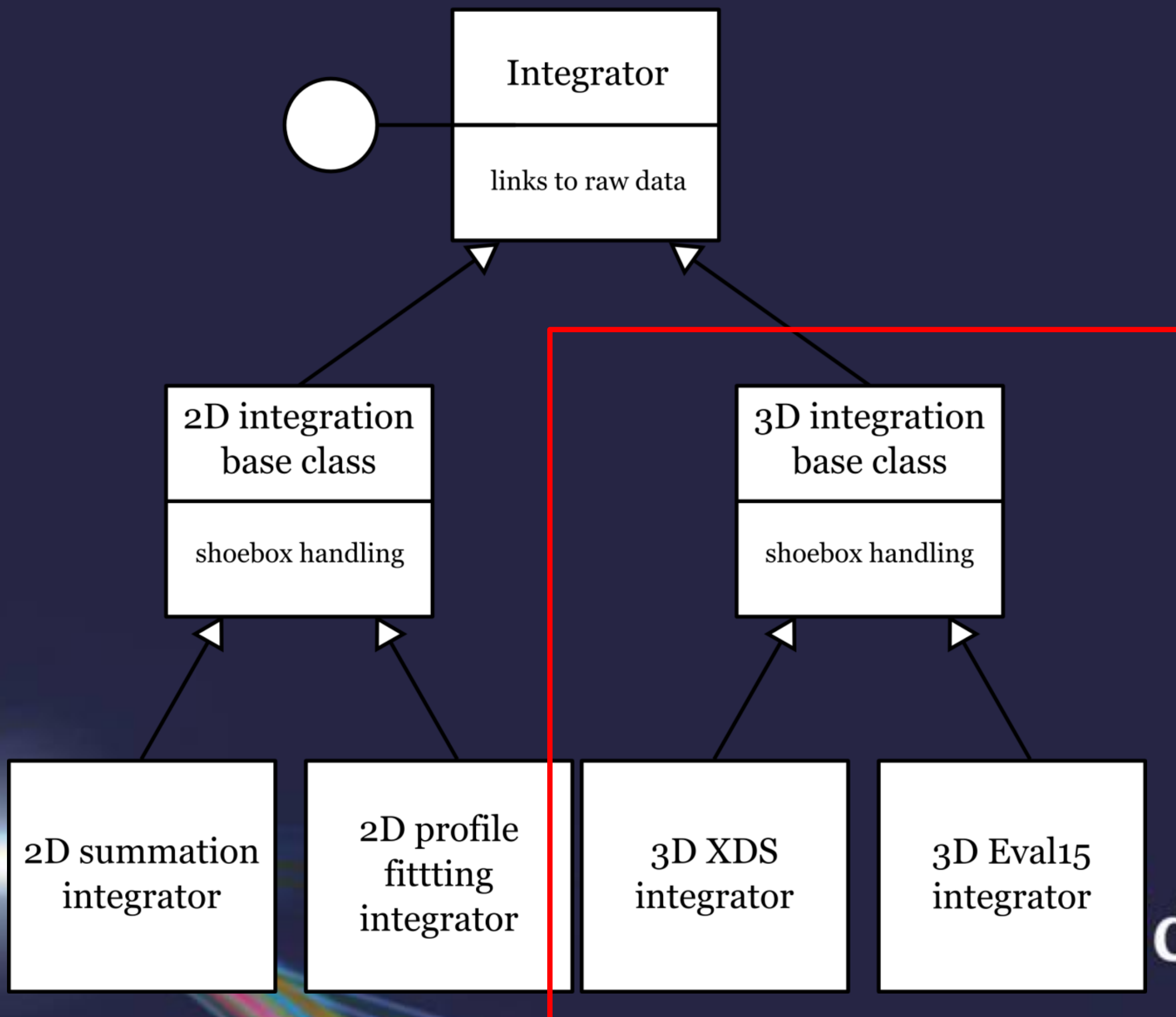
DIALS: 3D integration

James Parkhurst

Design principles (interfaces)

Integrator hierarchy

See Graeme Winter's talk



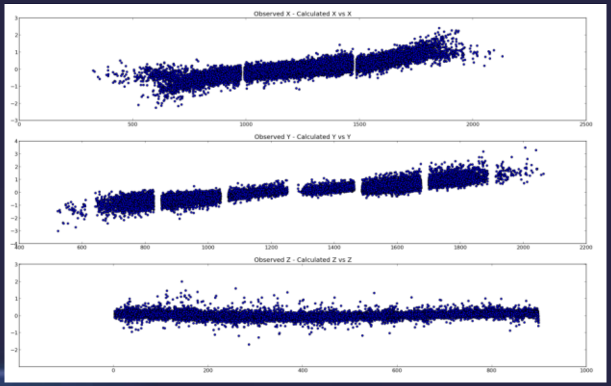
diamond

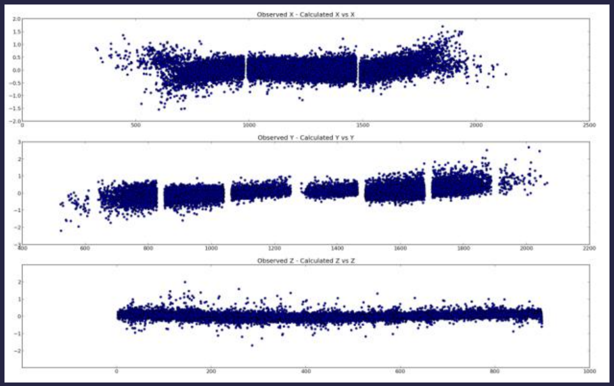
DXTBX: experimental models

- Experimental models provide access to data and methods to abstract from hardware details
- Designed to be extensible through decorators
- For example, the detector model is currently being updated to apply a parallax correction where appropriate

Models	
Beam	Direction Wavelength Polarization
Goniometer	Rotation axis
Scan	Image range Oscillation range Conversion between image number and angle
Detector	Geometry Trusted regions and pixel values Supports multiple detector panels Performs ray intersection with virtual detector plane Supports complex pixel to millimetre mappings

Differences between observed spot centroids and predicted Bragg peaks for 20802 strong reflections without and with parallax correction





diamond

DXTBX: sweep interface

- High-level interface to the DXTBX initial experimental models and image data. Developed in collaboration with LBNL and available in CCTBX.
- Access image data stored across multiple files (e.g. CBF) or a single HDF5 file through the same interface.
- Instantiated by factory function taking a list of filenames allowing creation of a list of sweeps or image-sets.
- Gives access to initial experimental models.
- Simple access to image data using python list syntax and slice notation.

Sweep example

```
In [1]: from dxtbx.imageset import ImageSetFactory
...:    from glob import glob

In [2]: # Initialise with list of filenames
...:    sweep = ImageSetFactory.new(glob('centroid*.cbf'))[0]
...:    print sweep

['centroid_0001.cbf', 'centroid_0002.cbf', 'centroid_0003.cbf',
'centroid_0004.cbf', 'centroid_0005.cbf', 'centroid_0006.cbf',
'centroid_0007.cbf', 'centroid_0008.cbf', 'centroid_0009.cbf']

In [3]: # Access experimental models
...:    b = sweep.get_beam()
...:    d = sweep.get_detector()
...:    g = sweep.get_goniometer()
...:    s = sweep.get_scan()

In [4]: # Easy indexing like python lists
...:    subsweep = sweep[4:7]
...:    print subsweep

['centroid_0005.cbf', 'centroid_0006.cbf', 'centroid_0007.cbf']

In [5]: # Read image data
...:    for image in subsweep:
...:        print image.all()

(2527, 2463)
(2527, 2463)
(2527, 2463)

In [6]: # Extract 3D volume
...:    volume = subsweep.to_array()
...:    print volume.all()

(3, 2527, 2463)
```

diamond

Archiving processed data

- Using HDF5 to save processed data
- Allows good compression (For example ~1.5GB of processed data was compressed to ~50MB)
- Easy to save properties of individual reflections in datasets
- Difficult to save many profiles with different sizes
- Currently saving each profile in it's own dataset (not very efficient!)
- Currently only saving per reflection information, not relationships between reflections (i.e. overlaps etc)

diamond

Algorithm interfaces

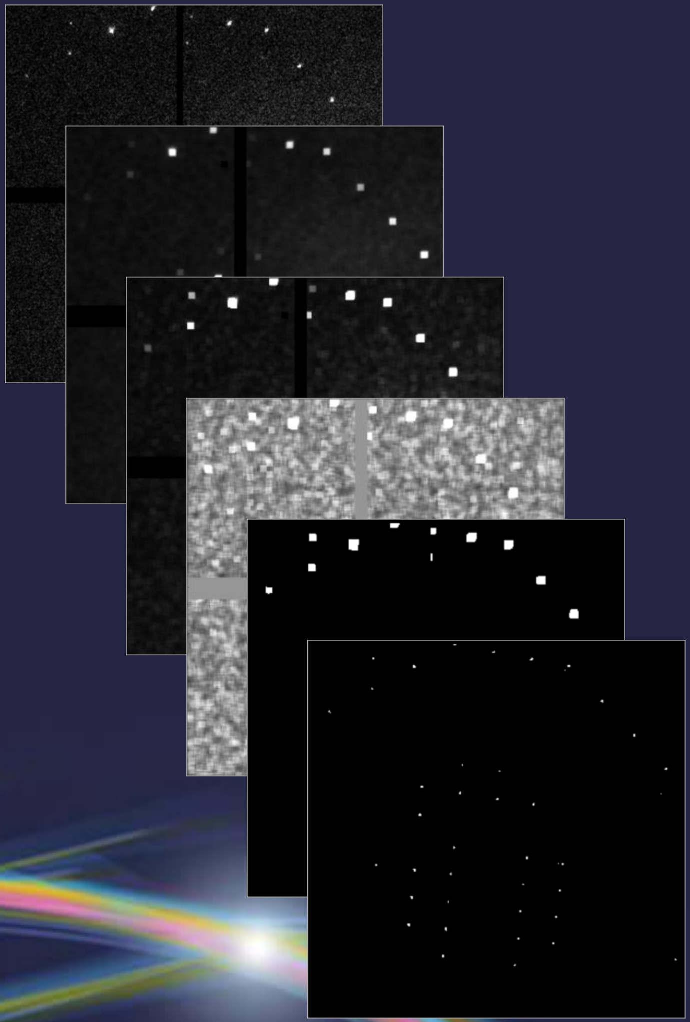
- Algorithms designed to be interchangeable
- Make use of python facilities to achieve this
- All top-level algorithms implement a simple high-level interface
- Separates configuration of algorithm from calling it
- Can specify internal functionality using different strategies

```
class Algorithm(object):

    def __init__(self, **kwargs):
        pass

    def __call__(self, data):
        pass


algorithm = Algorithm(parameter_a=True, parameter_b=False)

result = algorithm(data)
```

```
class Algorithm(object):

    def __init__(self, **kwargs):
        self.strategy = kwargs['strategy']

    def __call__(self, data):
        return self.strategy(data)


algorithm = Algorithm(strategy=DoSomething())

result = algorithm(data)
```

diamond

Finding strong spots

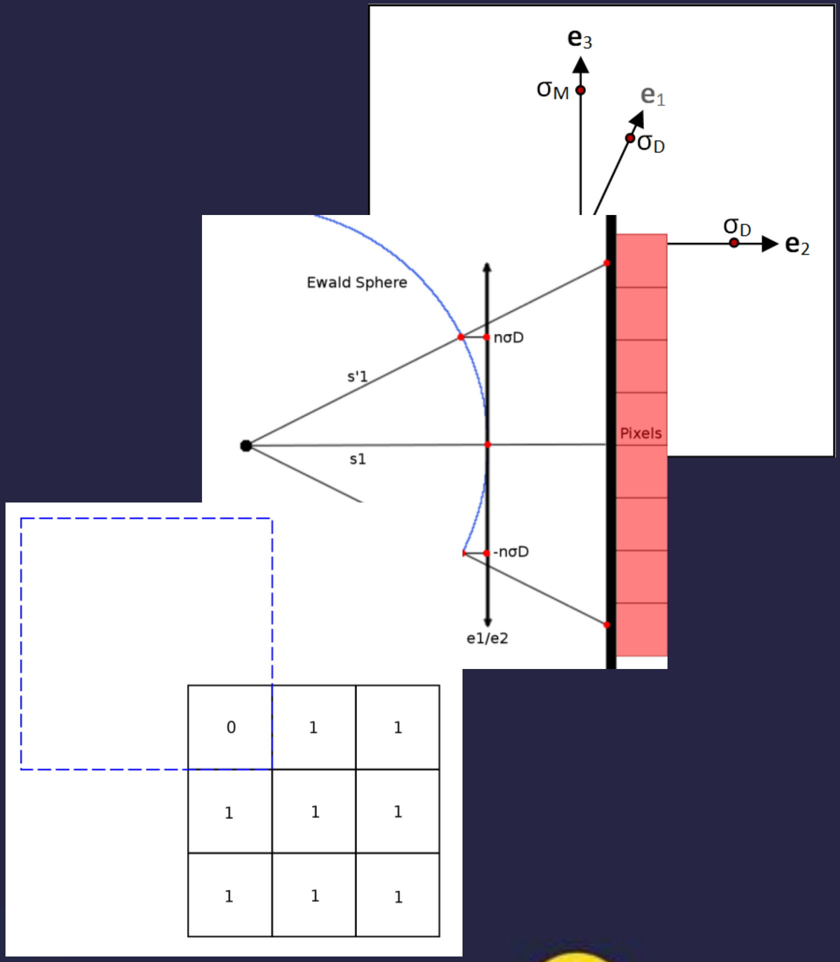


- Filter the image with a mean and variance box filter
- Calculate the index of dispersion for each pixel (σ^2/μ)
- Threshold pixels with (σ^2/μ) > 6 standard deviations greater than the expected value
- Threshold pixels with value > 3 standard deviations than the local mean
- Label connected pixels in 3D as belonging to the same spot
- Discard spots with fewer than 6 pixels
- Discard spots whose centroids differs from the pixel with the greatest intensity by more than 2 pixels

diamond

Reflection mask

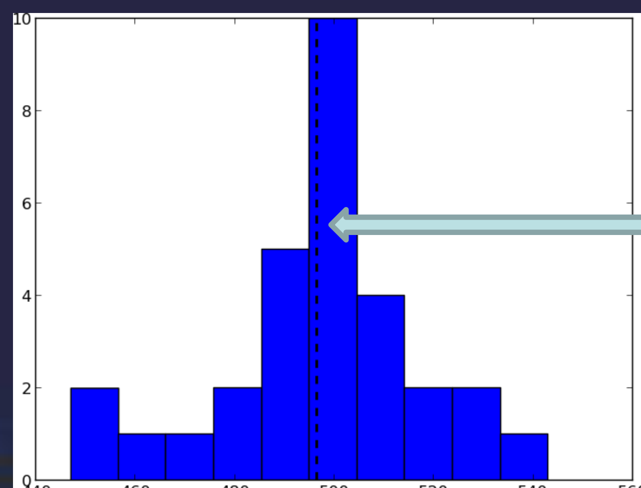
- Calculate shoebox specific to each reflection using standard deviation of beam divergence (σ_D) and mosaicity (σ_M) in reciprocal space
- Extract shoebox profile for each reflection
- Use a fast collision detection algorithm to find overlapping shoeboxes
- Overlapping reflections are recording in an adjacency list
- pixel ownership is recorded using a shoebox mask for each reflection



diamond

Background subtraction

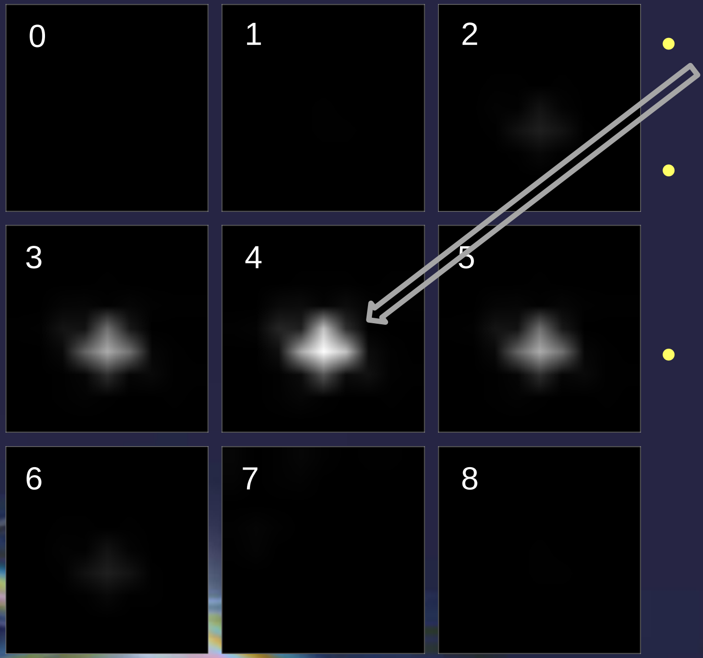
- Assume enough pixels available (> 10) to calculate background
- Assume background intensity distributed normally
- Remove high intensity pixels, one at a time, until intensity is normally distributed
- Select mean of remaining pixels as background intensity
- Correct over-estimated background intensity with modeled distribution (in progress)



Background intensity value

Reciprocal space transform

- Forward and reverse coordinate transforms have been implemented.
- Algorithm to transform reflection profile onto reciprocal space grid has been implemented using pixel sub-division method as in XDS.
- Should be possible to implement more accurate analytical mapping.



- Originally had issues with mis-centred reflection profiles
- This seems to have been solved by the use of a parallax correction (requires testing to verify)
- Is sensitive to subtracted background. Under-estimated background intensity can result in erroneous structure recorded on transformed grid.

diamond

Summary

- Work on DXTBX is more or less complete:
  - experimental models
  - parallax correction
  - high-level sweep interface

- Implemented algorithms for:
  - spot finding
  - spot prediction
  - reflection mask
  - background subtraction
  - reciprocal space transform

- Next steps:
  - Complete and test 3D summation integration
  - Do profile fitting in reciprocal space
  - Rigorously test using different datasets
  - Better calculation of the beam divergence and mosaicity
  - Connect with refinement module