Support for Time in ITK

Patrick Reynolds
Patrick Cheng
John Galeotti
Arnaud Gelas

Use Cases

- Conventional Video (2D+t)
- Ultrasound (3D+t)
- Microscopy (ND?)

Potential Solutions

Ranked by **impact** to current ITK infrastructure - ITK library developer's perspective

1. Treat time as another dimension. (Eg. 4D = 3D+t)
    - **Pros**: No impact on existing architecture, no specialized algorithms for processing time series data
    - **Cons**: Users need to develop their own specialized algorithms based on their assumptions on data structure. Does not support live video. Does not support different time steps.
2. Create a new **itkVideo** class
    - **Pros**: Clear definition/separation of time series data. Relative small impact on ITK. Enable development of specialized video algorithms in ITK, especially GPU accelerated version.
    - **Cons**: Need to adapt many existing algorithm to process itkVideo objects, or requires use of an intermediate interpolating filter. Does not support different time steps without also incorporating timestamps.
3. Separate time dimension, treat it separately (**Timestamp** all ITK data, make it explicit)
    - **Pros**: Less confusion in data structures, as time is explicitly defined. Enable development of generic filters on time series data.
    - **Cons**: Involving changing itkData and itkImage classes. Need to either modify many filter classes to handle time dimension, or requires use of an intermediate interpolating filter.

Needs for ITKv4 Core

- Timestamp on the Data Object (High Precision)

- itk::ImageSet (ImageSequence, ImageBag)
  - Leads to itk::ImageRingBuffer
  - itk::Video
  - itk::ImageSetToImageSetFilter
  - itk::ImageSetToImageFilter
  - itk::MultiIndexImageSet
  - Etc...

Proposed Solution: Timestamps

- Timestamp all ITK data
  - Standardized metadata item
- Required to handle different time steps
- Example 1: External-event triggered “video”
- Example 2: Dynamically adjusting frame-rate vs. crop-size
- Example 3: Looping microscopy z-axis
  - Slice $_{z=1,t=1}$, slice $_{z=2,t=2}$, …, slice $_{z=n,t=n}$, slice $_{z=1,t=n+1}$

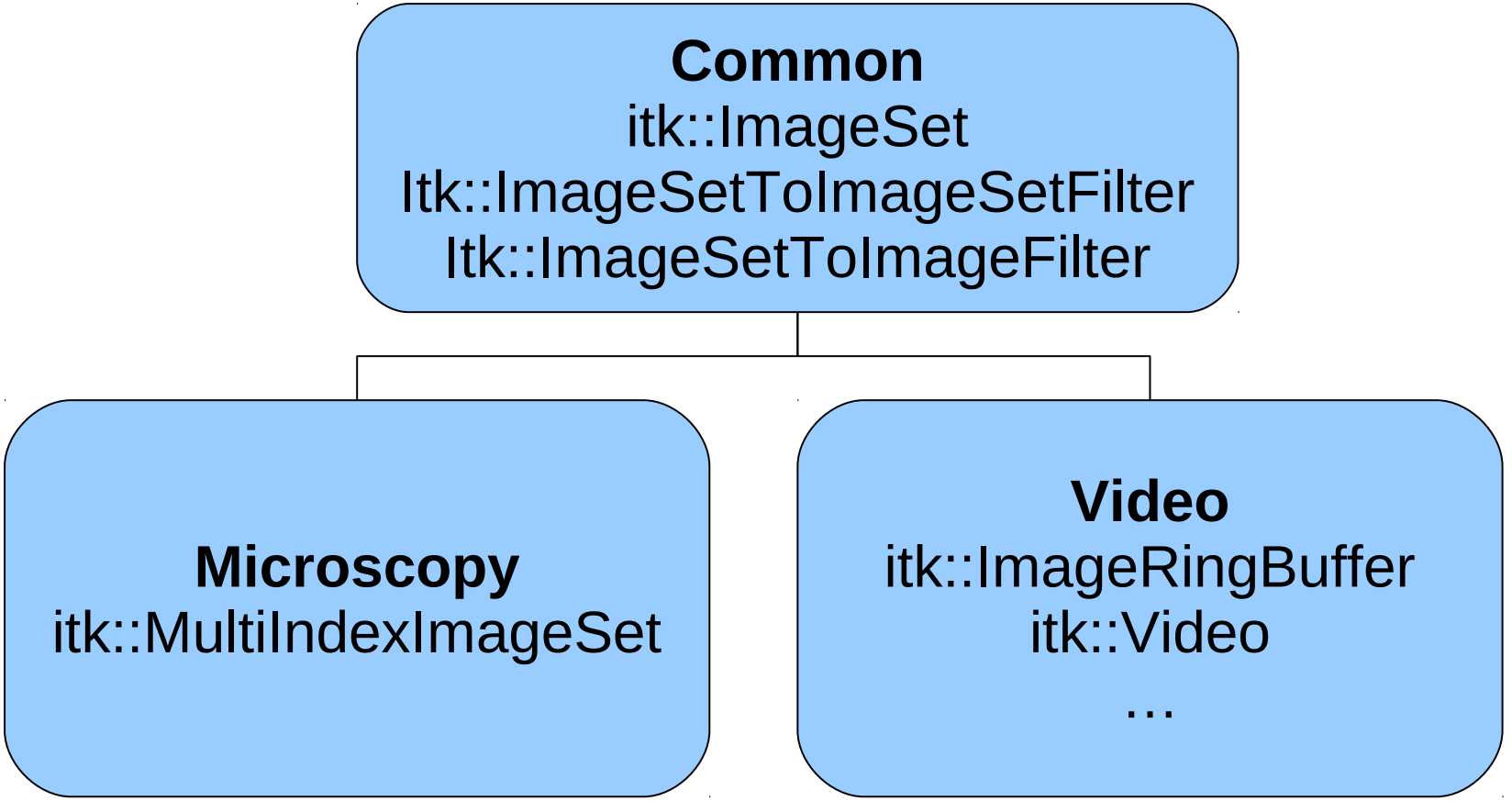
Proposed Solution: Itk::ImageSet Core Base Class

- Base-class is an ordered list
  - Appropriate for video storage, etc.
- Sub-classes (not part of core):
  - Database-like representation, needed for 9-dimensional microscopy data
  - Ring-buffer representation, needed for live video data
- Also provide a generic core ImageSetIterator

Core Filter Base Classes

- ImageSetToImageSetFilter
- ImageSetToImageFilter
  - Default implementation interpolates a set of N-dimensional images into an (N+M)-dimensional image.
  - M is determined by presence of timestamps and global physical coordinates/orientation
    - M=1 if only using timestamps
    - M=2 if moving a camera or ultrasound probe through a 3D volume.
  - This can be *slow* in the general case, but a GPU implementation could really help
  - This would be the “normal” way to interface time-stamped images with the rest of ITK

Proposed Division



Video Module

- RingBuffer subclass of ImageSet
- Video File Reading
  - Create native readers, or use OpenCV?
  - Templated to output either 2D+t ImageSet, RingBuffer, or 3D Image Volume.
- Live Video Acquisition to RingBuffer
  - Patrick Cheng's Simple Native Methods
  - OpenCV interface
    - OpenCV supports camera calibration and undistortion
    - OpenCV-interface submodule downloaded by default, but only built if OpenCV is detected?
  - VXL interface
    - VXL submodule off by default?

RingBuffer Pipeline Support

- Pipeline Updates can be triggered by interrupts, events, and/or polling for new frames.
  - Do we support all of these?
- When an input RingBuffer is modified, it is usually appropriate to output a new frame, NOT recompute the entire output.
- What if 5 new input frames arrive at once?
  - Then output at most 5 new frames, up to the size of our output RingBuffer.

Microscopy Module

- itk::MultiIndexImageSet

```
boost::multi_index::multi_index_container<
   FileName,
   boost::multi_index::indexed_by<
    ordered_non_unique<
        tag<PCoord>,
        BOOST_MULTI_INDEX_MEMBER(
            FileName, unsigned int, PCoord)>,
    ordered_non_unique<
        tag<RCoord>,
        etc…
```

Microscopy Module

- What about SimpleITK?
  - Provide optional SQL database support to use in lieu of Boost?
  - SQL might also simplify/standardize reading/writting
  - Matlab and Python can both interface w/ SQL.

Discussion

- Should we have a generic way to gather data asynchronously?
- ImageSetToSpatialObjectFilter for things like SURF and SIFT