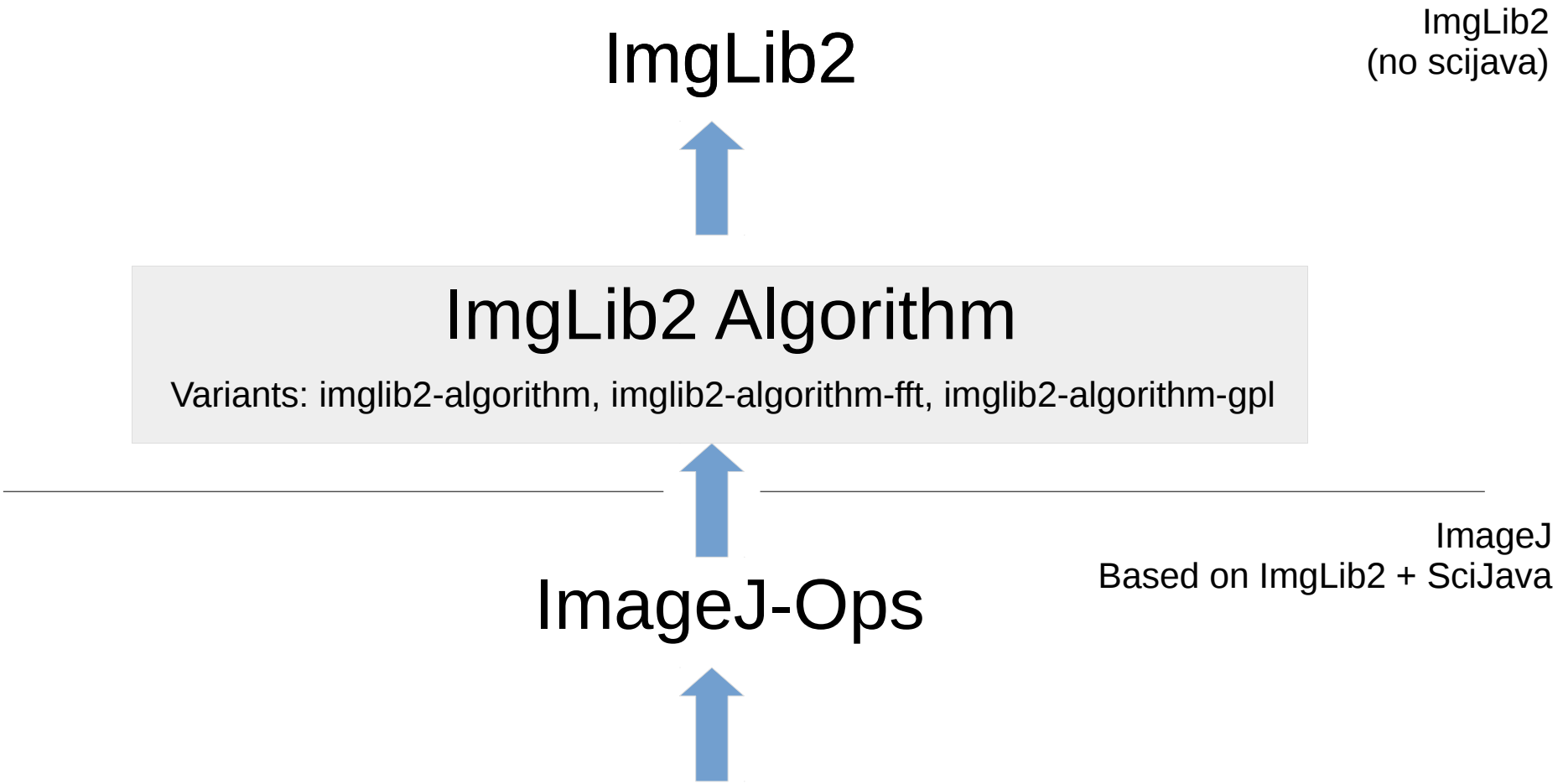

# Algorithm




Matthias Arzt
maarzt
arzt@mpi-cbg.de

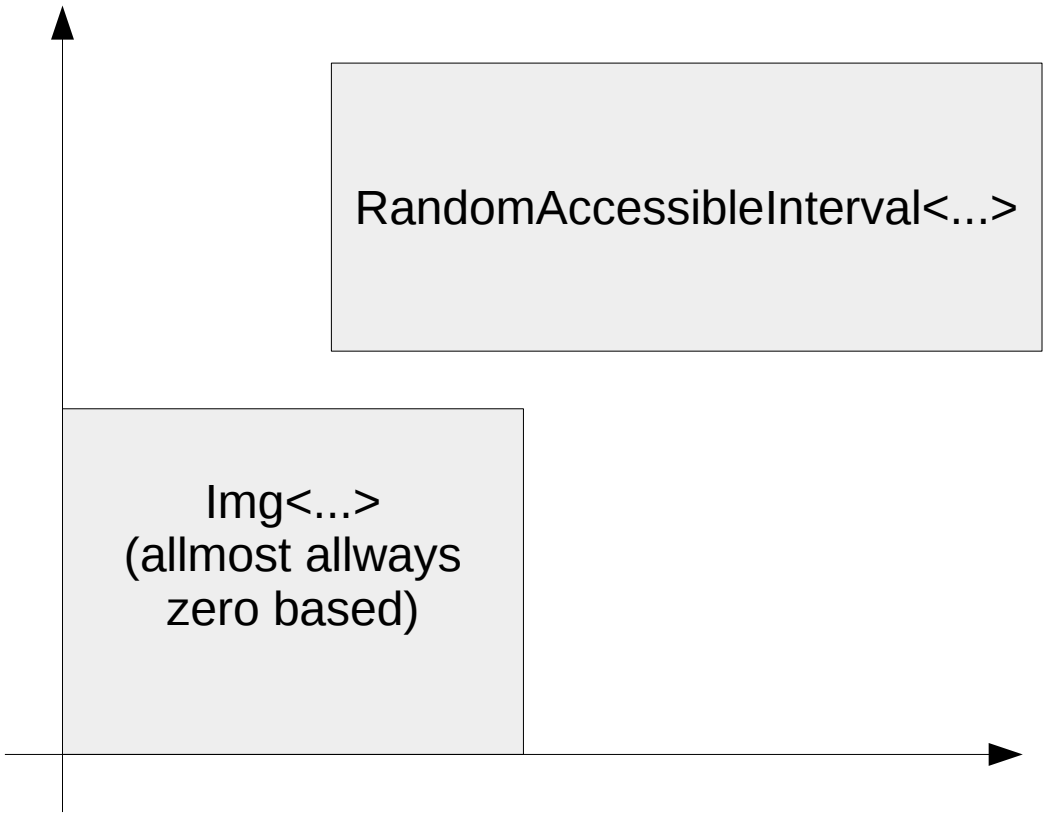

ImgLib2
ImgLib2
(no scijava)
ImgLib2 Algorithm
Variants: imglib2-algorithm, imglib2-algorithm-fft, imglib2-algorithm-gpl
ImageJ-Ops
ImageJ
Based on ImgLib2 + SciJava
FIJI, ImageJ, Scripts, IDE+ImageJ

# Imglib2 Algorithm vs. Ops

- Low level functionality
- Light weight
- Long parameter lists
- Usable on huge images
- Statically Linked
- Not extensible

- High level functionality
- Many Functions
- Easy to use
- Automatic type conversion
- Dynamically Linked extensible via plug ins (Scijava)

# Intervals



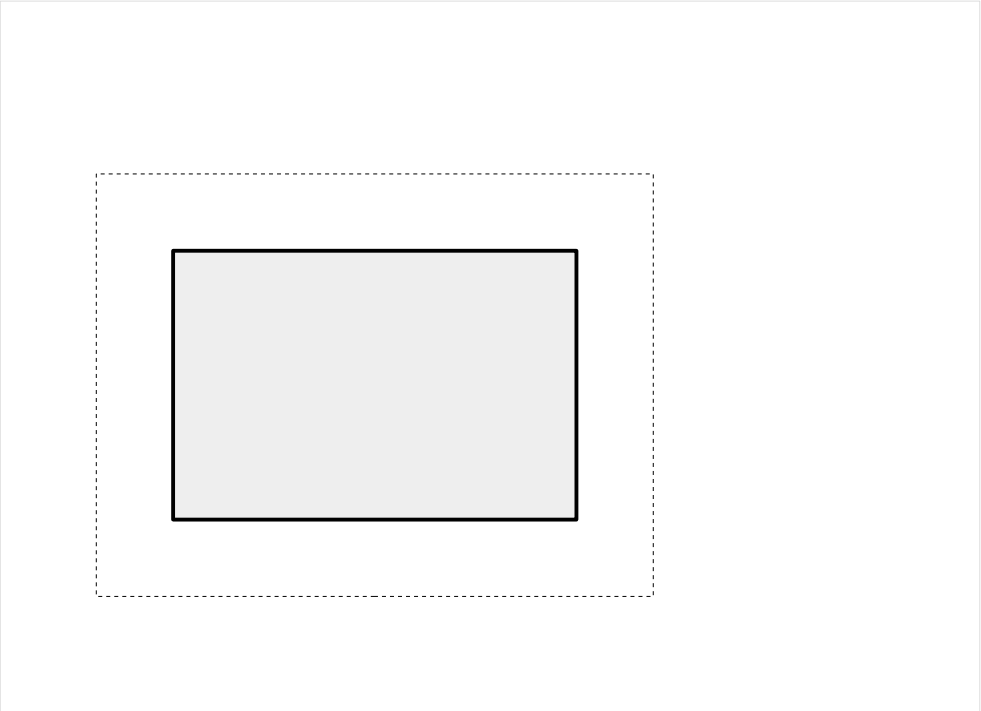

n-dimensional

# Avoid Memory Allocation
# Provide Output Buffer

~~`Img<T> output = Gauss3.gauss(sigma, input);`~~

```
Img<T> output = imageFactory.create(dimensions);
Gauss3.gauss(sigma, input, output);
```

# Infinite Input - Paradigm



```
Not: Gauss3.gauss(sigma, input, equallySizedOutput); >> CRASH

Gauss3.gauss(sigma, Views.extendBorder(input), output);
```

# Loop Over Three Images

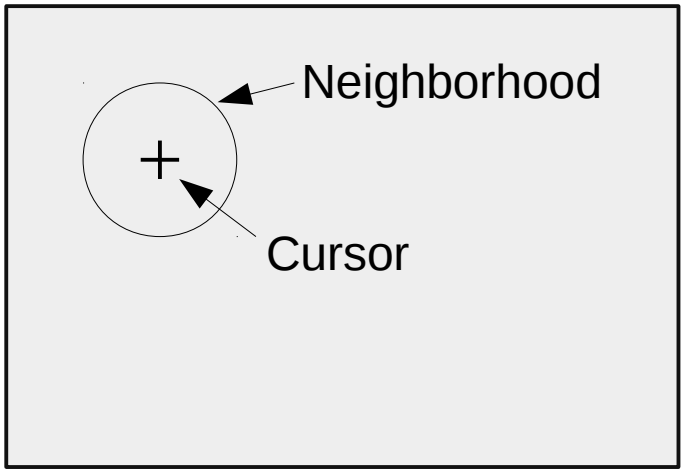
```
Cursor<FloatType> a = Views.flatIterable( imageA ).cursor();
Cursor<FloatType> b = Views.flatIterable( imageA ).cursor();
Cursor<FloatType> c = Views.flatIterable( imageA ).cursor();
while( a.hasNext() ) {
    a.fwd();
    b.fwd();
    c.fwd();
    c.get().setReal( a.get().getRealFloat() + b.get().getRealFloat() );
}
```

# Loop Builder

```
RandomAccessibleInterval< FloatType > imageA = ...
RandomAccessibleInterval< FloatType > imageB = ...
RandomAccessibleInterval< FloatType > imageC = ...

// Example 1
LoopBuilder.setImages( imageA ).forEachPixel(
        pixel -> pixel.set( 255 )
);

// Example 2
LoopBuilder.setImages( imageA, imageB ).forEachPixel(
        ( a, b ) -> b.set( a )
);

// Example 3
LoopBuilder.setImages( imageA, imageB, imageC ).forEachPixel(
        ( a, b, c ) -> c.setReal( a.getRealFloat() + b.getRealFloat() )
);
```

# Neighborhoods

- Many operations need an Neighborhood




Different Neighborhood Shapes:



HypersphereShape



DiamondShape



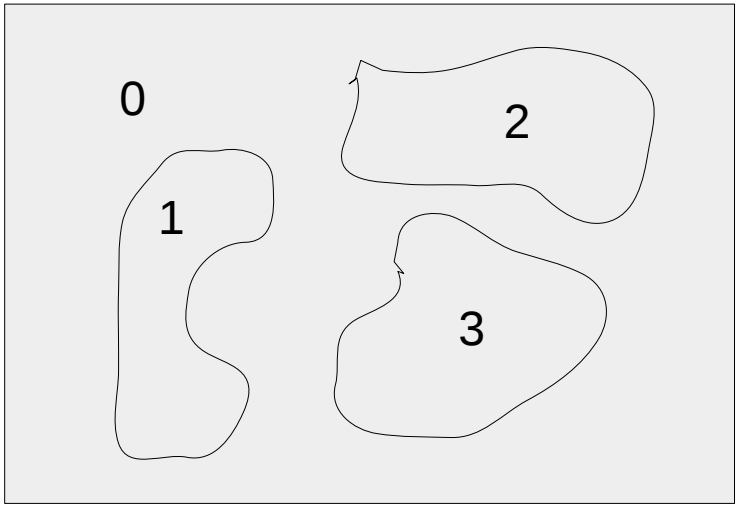
RectangularShape

# Neighborhoods

```
public static void meanFilter (
         int radius,
         RandomAccessible<FloatType> input,
         RandomAccessibleInterval<FloatType> output)
{
     Shape shape = new HyperSphereShape( radius );

     RandomAccessible<Neighborhood<FloatType>> neighborhoods =
              shape.neighborhoodsRandomAccessible( input );

     RandomAccessibleInterval<Neighborhood<FloatType>> neighborhoodsInterval =
              Views.interval( neighborhoods, output );

     LoopBuilder.setImages( neighborhoodsInterval, output ).forEachPixel(
              (neighborhood, out) -> out.setReal( calculateMean(neighborhood) )
     );
}

private static float calculateMean( Neighborhood<FloatType> neighborhood )
{
     double sum = 0;
     for ( FloatType pixel : neighborhood )
          sum += pixel.getRealDouble();
     return (float) ( sum / neighborhood.size() );
}
```

# ImgLabeling

Img<IntegerType>



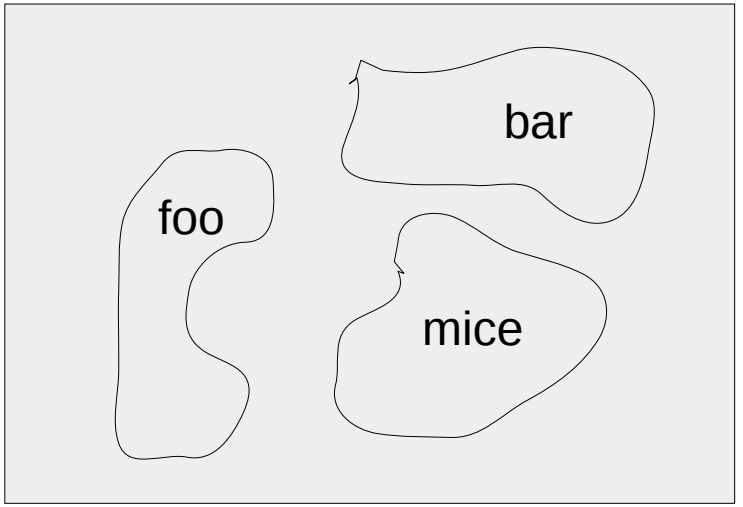

Map<int, Set<String>>

0 → {}

1 → {“foo”}

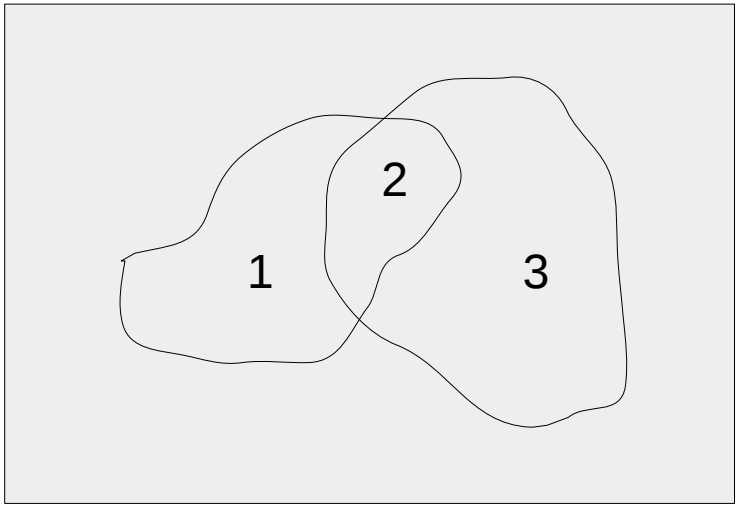
2 → {“bar”}

3 → {“mice”}



# ImgLabeling

Img<IntegerType>



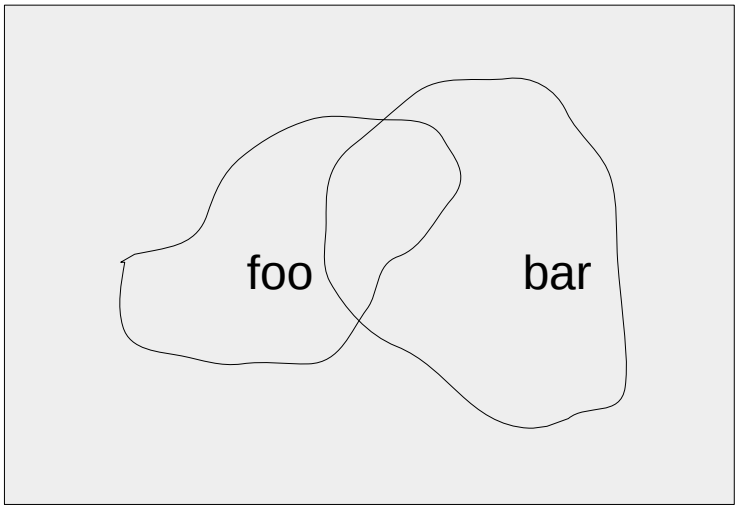

Map<int, Set<String>>

0 → {}

1 → {“foo”}

2 → {“foo”,”bar”}

3 → {“bar”}



# Smallest Cheatsheet

- Image Containers:
  - Img, RandomAccessibleInterval, RandomAccessible, ImgLabeling
- Utility Classes:
  - Views:
    - RandomAccessible<...> ra = Views.extendBorder(rai);
    - RandomAccessibleInterval<...> rai = Views.interval(ra);
    - IterableInterval<...> ii = Views.iterable(rai);
    - RandomAccessibleInterval<Pair<…,...>> = Views.pair(a, b);
    - Views.stack(), Views.hyperSlice(), Views.translate
  - Converters:
    - Converters.convert(rai, (I, o) → o.setReal(i.getRealDouble), new DoubleType());
  - Intervals:
    - Intervals.minAsLongArray(interval), Intervals.translate()
  - LoopBuilder
  - ArrayImgs:
    - Img<DoubleType> image = ArrayImgs.doubles(dimensions)

# What’s in imglib2-algorithm?

- Gauss(3), DifferenceOfGaussians
- Morphology:
  - Opening, Closing, Erosion, Dilation, TopHat
- Connected Components & Watershed
- Neighborhoods
- LocalExtrema
- PartialDerivative:
  - (Forward, Backward & Central Derivatives)
- KDTree
- Histogram Calculation
- Hessian Matrix (?)
- Eigen Values
- FloodFill
- Multi Threaded, Thresholder

- ComponentTree
- SubpixelEdgeDetection
- Edgel
- IntergralImg
  (revert PatialDerivative)

# What is in imglib2-algorithm-fft?

- Fast Fourier Transform
- Also BSD License

# What is in imglib2-algorithm-gpl?

- FFT Convolution
- Localization
- Pick Image Peaks
- Patial Derivatives & Diffusion

# Exercises

- Calculate gradient using forward differences
- Count connected components
- Find local minima