ROI Specification [DRAFT]

Release 0.0.1

Roger Leigh

January 18, 2013

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Purpose

This document is a formal specification for the definition, storage and exchange of regions of interest (ROIs). This specification will be implementable in any programming language, and is intended to provide a common set of ROI types which will be usable in all image analysis software programs.

1.2 Scope

This specification defines abstract definitions of regions of interest, including details of how certain data structures and algorithms must be defined and behave, in order to ensure that ROIs work uniformly between the different programs and libraries implementing the specification. It also provides examples of serialised forms which may be used for storage and/or exchange. However, it does not define a file format; it is the responsibility of the implementors to integrate this model into their file formats as they see fit.

1.3 Reference implementation

This specification is accompanied by a reference implementation of the model. This implementation is intended to validate and test the correctness of the specification. It may be usable directly, however this is not the primary intention for its existence. Note that the reference implementation strives for complete correctness, and implementors of this specification may wish to provide additional optimisations to improve performance.

1.4 Concepts

- A ROI is an evaluation of a shape object
- A shape is defined by the rules which transform its representation (e.g. geometry, range within a dimension) into a a bitmask and/or greymask
- Each shape has a unique name (type) and number; the number is used for serialisation and versioning
- Each shape is described by one (or more) representations, these are the primitives which define the geometry or range within a dimension

- A shape object can be composed of one or more shapes, which can include transforms and shapes in arbitrary dimensions
- Each representation has a unique name and number; the number is used for serialisation and versioning
- Shapes which share representations may be freely interconverted; conversion is not required to be possible in both directions (e.g. square to rectangle or polyline to/from polygon)
- A shape is essentially a serialised expression which must be evaluated to create a usable ROI; given that certain shapes can contain other shapes, this provides for ROIs which are both extensible and of arbitrary complexity.
- All shapes can be serialised as a sequence of numbers
- Given that each shape can be reconstructed using its shape and representation numbers, which specify the exact sequence of numbers to deserialise to reconstruct the object, it is possible to exchange ROIs as simple text, or alternately as binary; more structured (but space inefficient) representations could be realised using XML.
- The object/function serialisation methodology used here is inspired by (but not derived from) the SSH FXP specification which defines the wire protocol for SFTP.

CHAPTER

TWO

REQUIREMENTS

2.1 Barcelona meeting

The following points are taken from the meeting notes.

Note: These may be incomplete, please do correct if necessary.

- Iterate through all points in a ROI. Order is important for some use cases, but not all.
- Type mechanism needed. Use concrete types.
- Version types to allow for deprecation, and translation to new versions.
- Define a persistent data model.
- ROIs must be serialisable. Needed for exchange and persistence.
- Serialisation may be implementation dependent. This could be XML, text or binary. Which should be used as a transfer format (if any)?
- Allow conversion between different ROI types.
- Need the ability to specify an interval in an arbitrary dimension.
- **Hierarchy of interfaces.**
  - Interval, Renderable.
- Need to specify how to draw and display (and edit?) types such as curves so that it does not vary between implementations.
- Persistence and drawing are separate problems.
- **Transforms**
  - Abililty to attach transforms
  - **Non-affine transforms.**
    * Need examples to understand the problem better
  - Store transforms with ROI?
  - Apply multiple transforms to a ROI in sequence; nested list of transforms
  - Modelling spaces and objects in space; maybe define transforms separately and reuse them

  - Tree of transformations and operations
- **Union ROIs**
  - Only works in transform domain / "view space"
  - Union of hypervolumes. [How to represent different shapes at different times?]
- Needs to be able to scale up to millions-billions of ROIs
- **Specific ROI types**
  - Include checkerboard (uneven integers)
  - Hierarchy of ROIs; compound list
- **Rendering:**
  - jHotDraw and other drawing toolkit independence
  - Need objects to be manipulable
  - List of control points
- **Editing:**
  - Needed to manipulate shapes
- **Tree of operations**
  - compiler/interpreter
  - obtain a "result"
- Grouping
- **Comparison of models:**
  - OME: ROI is union of shapes
  - ImgLib: Group is union of ROIs

CHAPTER

THREE

ROI DISCUSSION

3.1 ROIs in three dimensions

Some of the 3D primitives described below may appear to be redundant; it's certainly possible for example, to represent a shape in 3D right now using multiple shapes, one per z plane. However, being able to use native 3D primitives is more powerful: it permits additional measurements involving volume, surface area and shape.

Some shapes may be represented equivalently in different ways; it might be worth considering adding support for these because it firstly allows the shape to be computed in different ways, which can differ depending upon the problem being solved, and it also contains information about how the measurement was made, i.e. the intent of the person doing the measuring, which is lost if converted to a canonical form.

3.2 Bitmasks

Mask Could we have a pointer to an IFD/file reference plus two coordinates so specify a clip region, then we can pack potentially hundreds of masks in a single plane.

3.3 Meshes

Mesh 2D mesh described as e.g a face-vertex mesh.

3DMesh 3D mesh described as e.g. a face-vertex mesh.

Meshes could be computed from masks, polygons, extruded shapes where there is a z range, or from thresholding.

3.4 Paths

3DPath As for Path, but with additional vector to describe motion along the prescribed plane?

CHAPTER

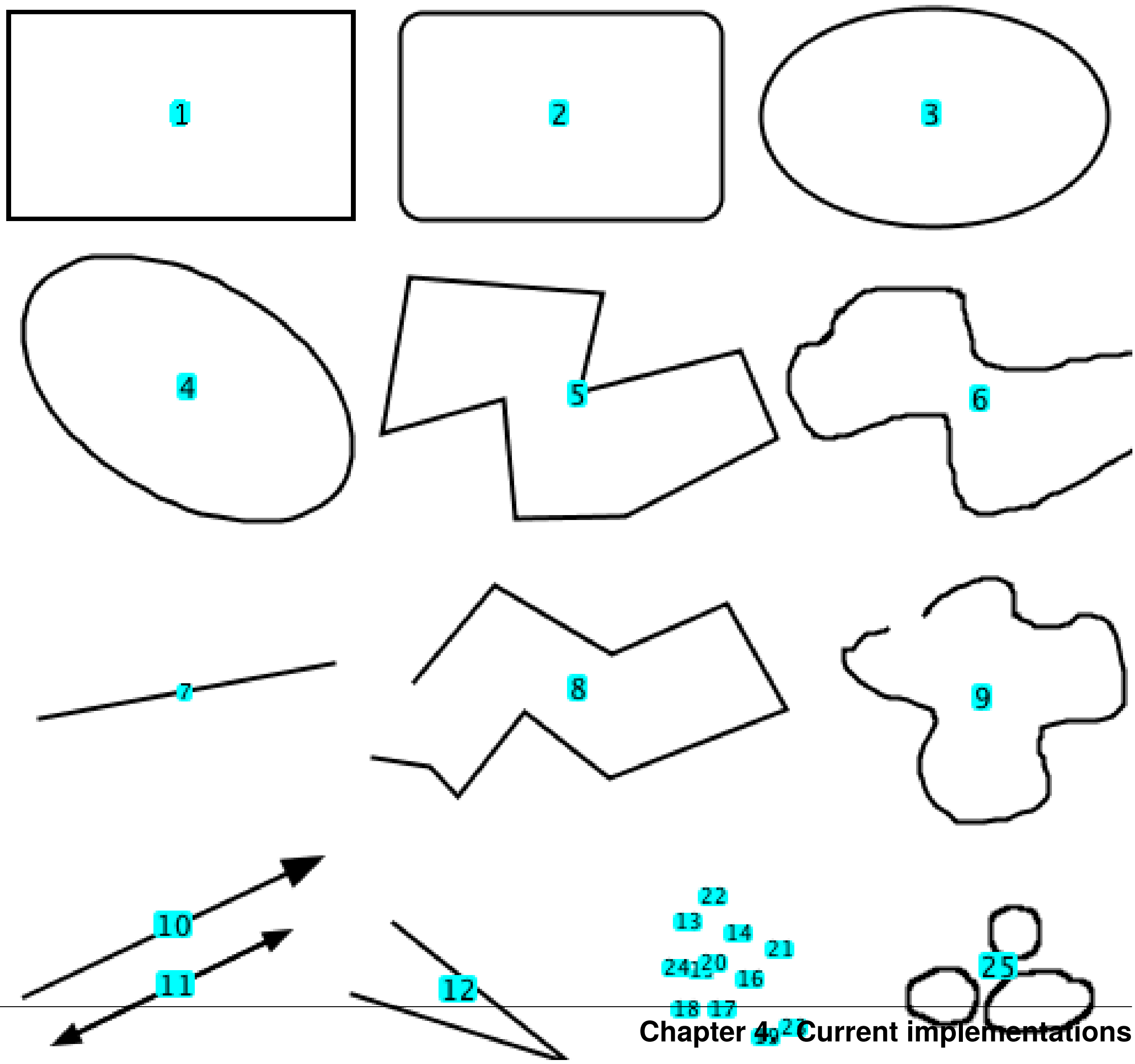
FOUR

CURRENT IMPLEMENTATIONS

4.1 AxioVision

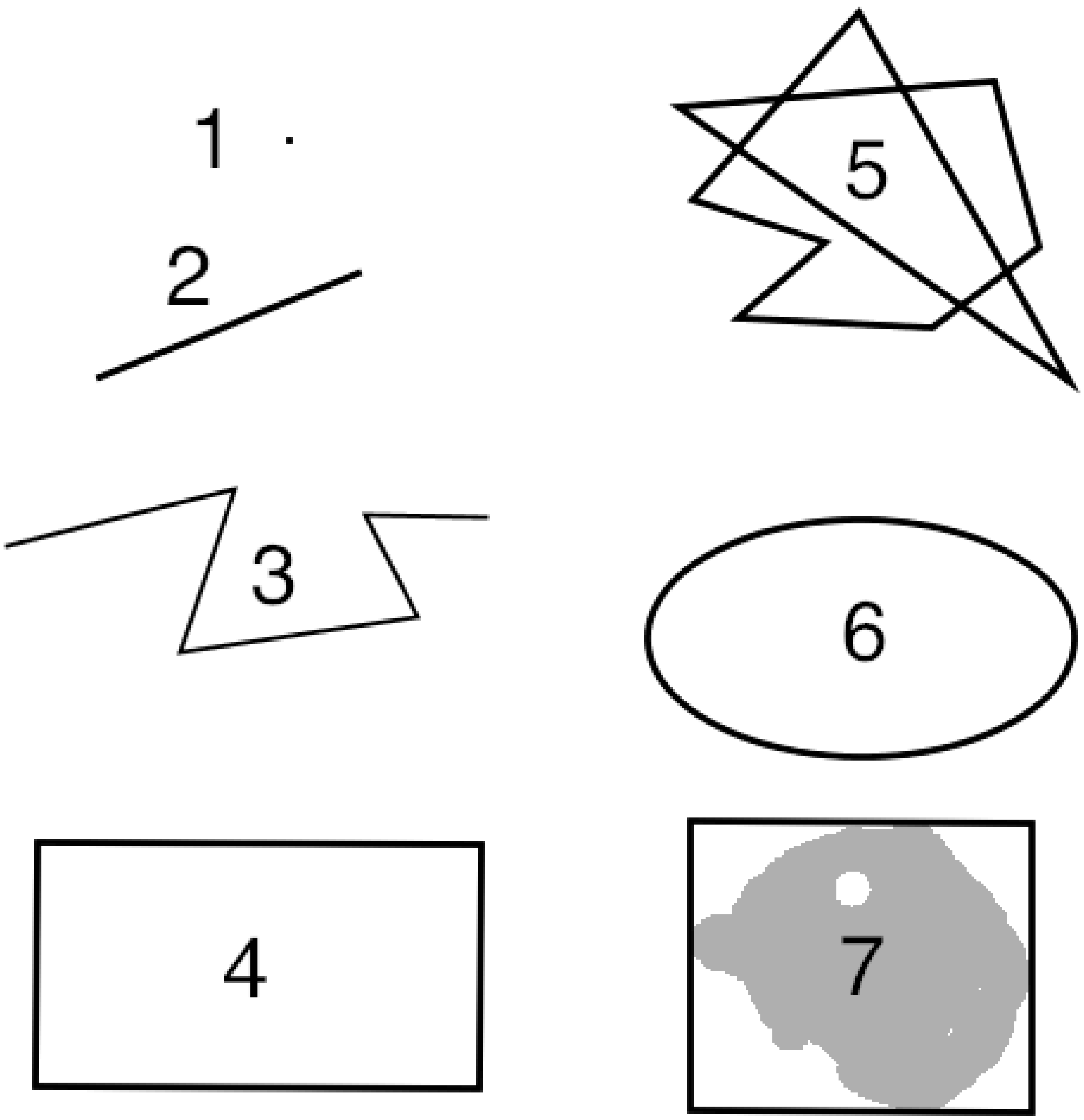
4.2 Cell Profiler

4.3 ImageJ



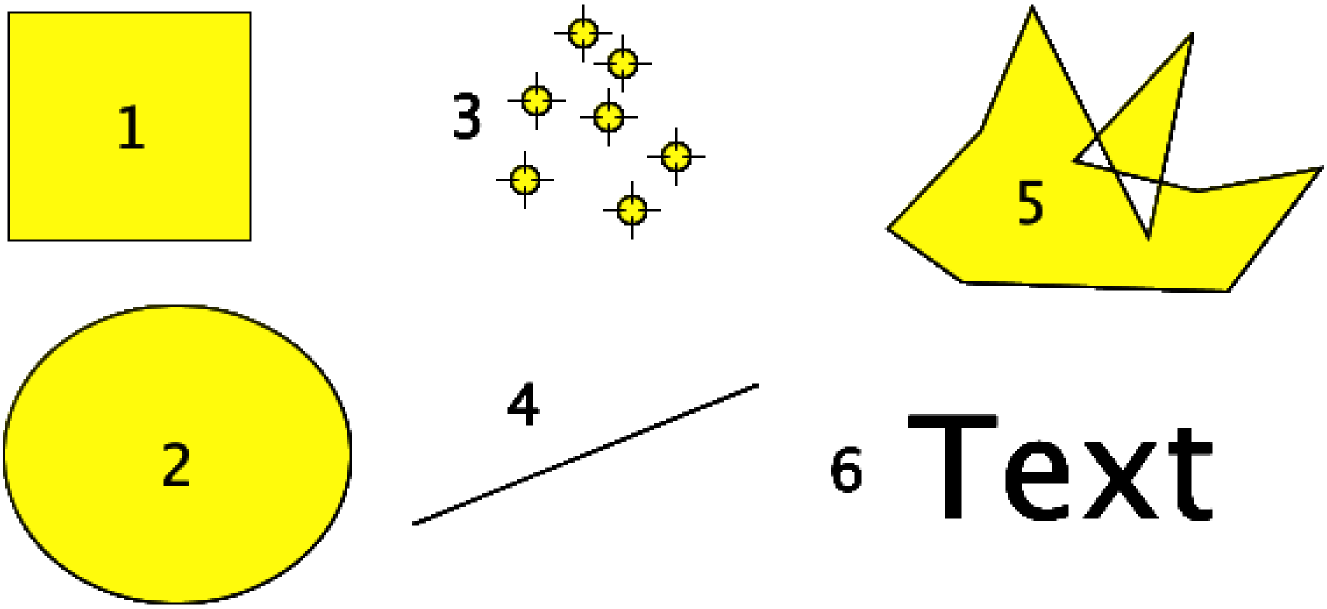
ROI	Description
1	Rectangle, square corners
2	Rectangle, rounded corners
3	Oval
4	Ellipse
5	Closed polyline
6	Closed polyline, “freehand”
7	Line
8	Open polyline
9	Open polyline, “freehand”
10	Arrow
11	Arrow, doubleheaded
12	Angle
13-24	Points
25	Bitmask
26	Text

4.4 Icy



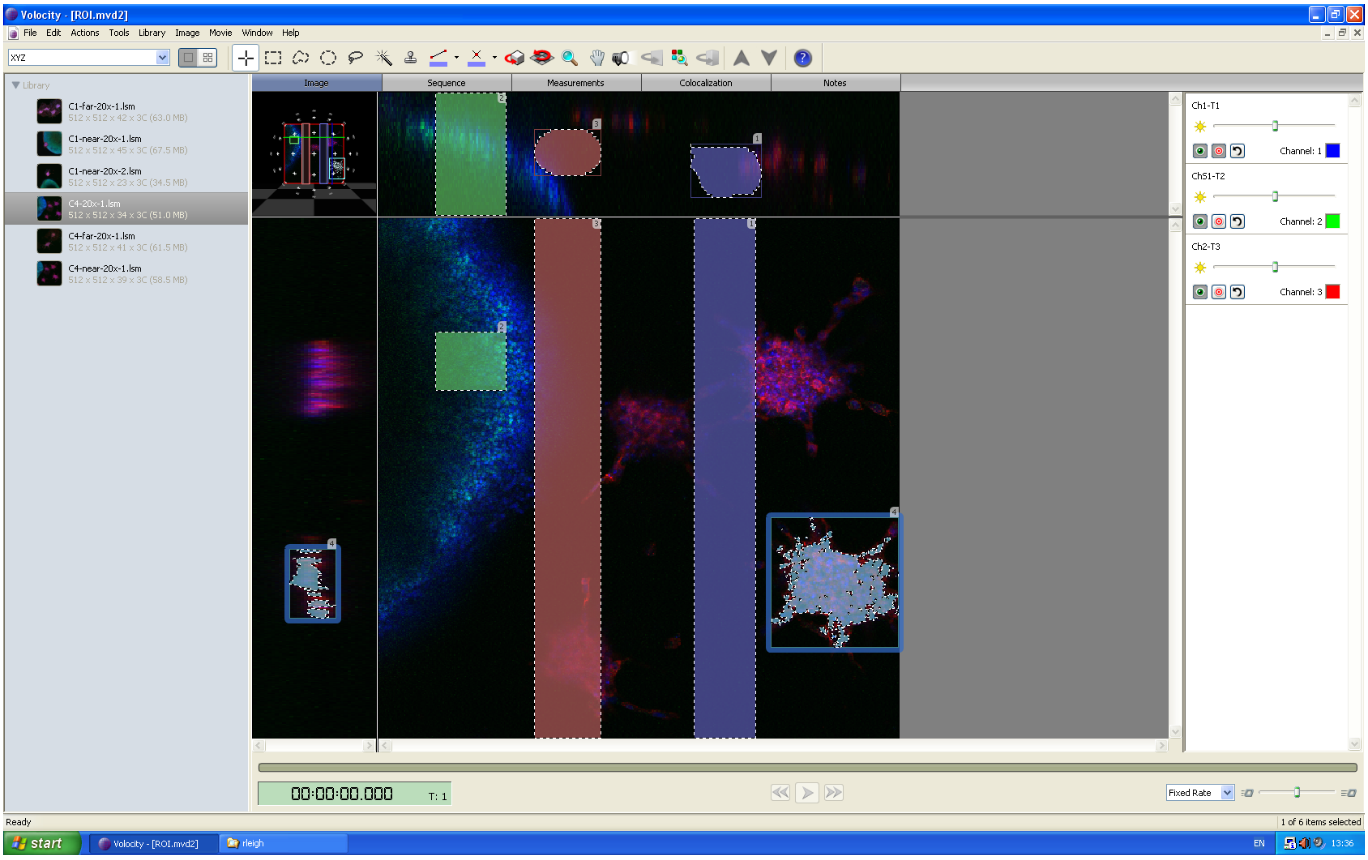
ROI	Description
1	Point
2	Line
3	Open polyline
4	Rectangle
5	Closed polyline
6	Ellipse
7	Bitmask

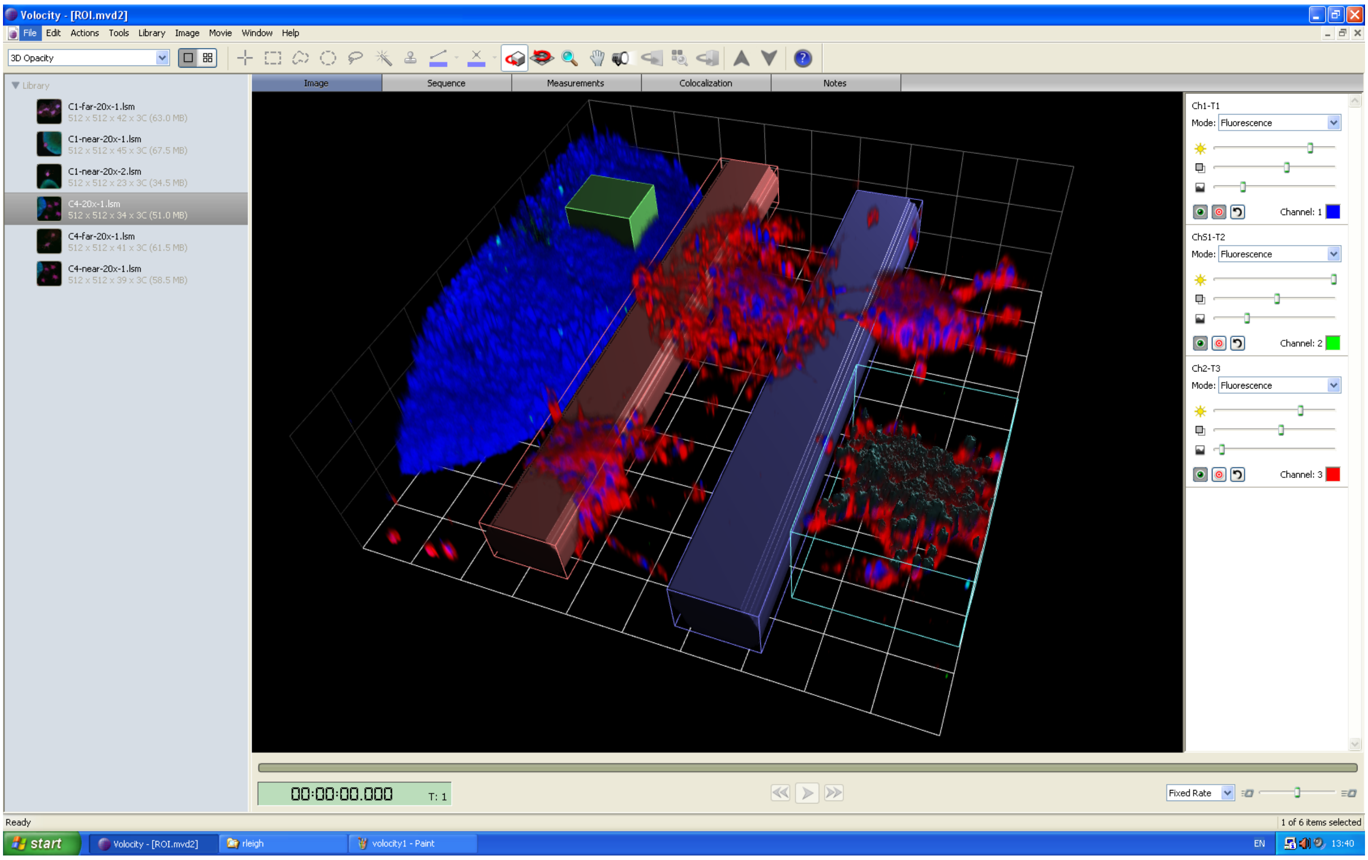
4.5 Insight

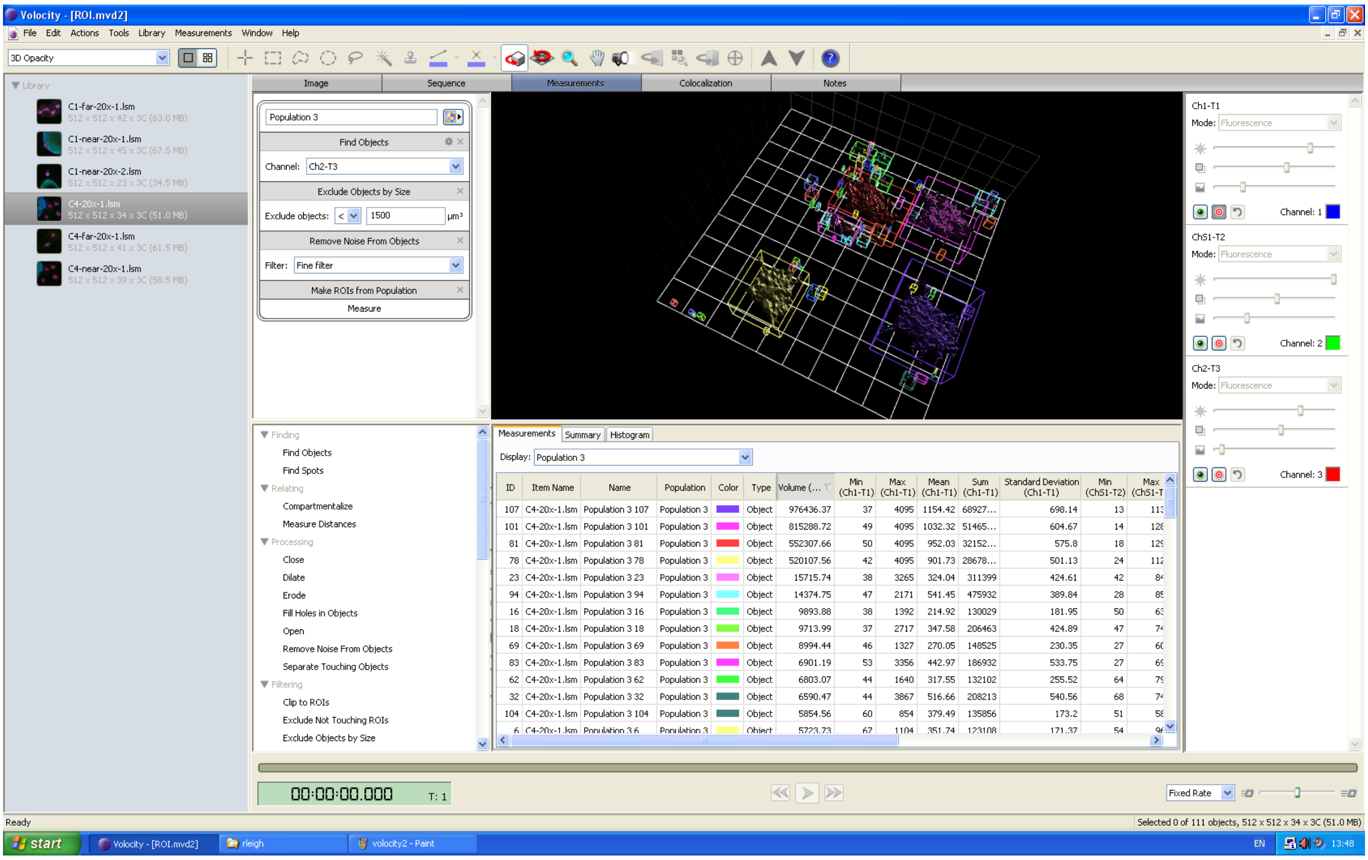


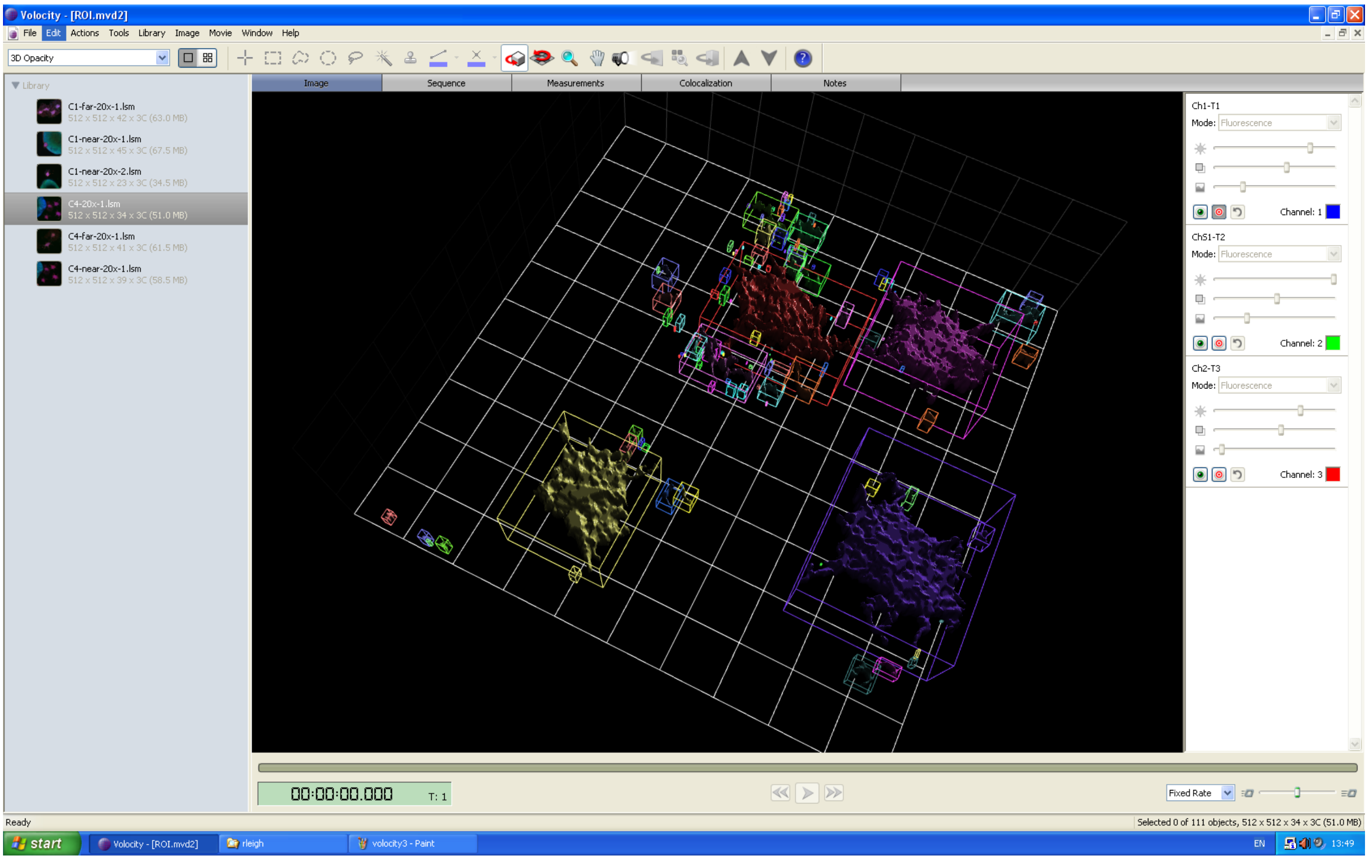
ROI	Description
1	Rectangle
2	Ellipse
3	Points
4	Line
5	Closed polyline
6	Text
7	Bitmask

4.6 Volocity


Volocity - [ROI.mvd2]
Image
Sequence
Measurements
Colocalization
Notes
00:00:00.000


Volocity - [ROI.mvd2]
Image
Sequence
Measurements
Colocalization
Notes
00:00:00.000


Volocity - [ROI.mvd2]


Volocity - [ROI.mvd2]

ROI	Description
1	Rectangle
2	Freehand polyline
3	Circle
4	Lasso
5	Stamp
6	Line
7	Point
8	3D mask / mesh

CHAPTER

FIVE

2D PRIMITIVES IN 3D

5.1 Conversion to 3D primitives

The existing 2D primitives may be represented by the equivalent 3D primitive for the 2D primitive, extruded in z to a single z plane thickness.

While this is desirable for reducing code complexity, retaining the 2D primitives is necessary for 2D measurements (area/perimeter). These can be obtained from the 3D shape by dividing the volume or surface area by the z thickness, respectively. Having the 2D primitives will provide the context for conversion of measurements from 3D volume to 2D surface, since these are otherwise meaningless for 3D ROIs which are not extruded 2D ROIs.

3D ROIs, where appropriate, could provide alternative forms for 2D use. For example, a 3D cylinder would, when extruded from a 2D circle, not have end faces (i.e. would be open), in order for 2D surface area measurements to be correct.

5.2 Use of 2D primitives in 3D space

While it would be possible to translate and rotate 2D primitives in 3D using a 4×4 matrix, it would be simpler for users if rotation could be specified using a unit vector which can specify the angle of the primitive in 3D space; the matrix transform can be trivially construct ed from the vector. However, note that while current transforms occur only in 2D, where the x and y pixel sizes are typically the same, this is not usually the case in z, and so the transformations may need performing in physical units; therefore adding proper support for units would also be desirable to fully support 3D transforms. Note that this would also solve the existing problem in 2D that prevents ellipses and rectangles being rotated (without the use of a matrix transform), though where the rotation centre should be may be shape- and context-dependent. The unit vector to (0,0,-1) which would specify the existing behaviour.

Note: Define behaviour of orientation of unit vector for rotation; which direction are primitives facing by default?

5.2.1 2D extrusion

Reconstruction of 3D shapes from 2D planes distributed in z/t. -> set of 3D objects in t.

5.2.2 2D decomposition

Decompose 3D shape into 3D planes distributed in z.

CHAPTER

SIX

DATA TYPES

Table 6.1: Data types

Name	TypeID
scijava.roi.Iterable	N/A
scijava.roi.RegionOfInterest	N/A
scijava.roi.RegionOfInterestSet	N/A
scijava.roi.Serialisable	N/A
scijava.roi.annotation.Grid	1002
scijava.roi.annotation.Scale	1001
scijava.roi.annotation.Text	1000
scijava.roi.dimconstraint.Extrude	3010
scijava.roi.dimconstraint.Range	3002
scijava.roi.dimconstraint.Set	3011
scijava.roi.dimconstraint.Value	3000
scijava.roi.dimconstraint.Values	3001
scijava.roi.measurement.Area	N/A
scijava.roi.measurement.Length	N/A
scijava.roi.measurement.Volume	N/A
scijava.roi.shape.AbstractTransform	2051
scijava.roi.shape.AffineTransform	2050
scijava.roi.shape.Arc	2008
scijava.roi.shape.BitMask	2040
scijava.roi.shape.Bitwise	2052
scijava.roi.shape.Cuboid	2020
scijava.roi.shape.Custom	650
scijava.roi.shape.Cylinder	2022
scijava.roi.shape.Ellipsoid	2021
scijava.roi.shape.GreyMask	2041
scijava.roi.shape.Line	2002
scijava.roi.shape.Lines	2003
scijava.roi.shape.Mesh	2023
scijava.roi.shape.PhysicalShape	N/A
scijava.roi.shape.Point	2000
scijava.roi.shape.Points	2001
scijava.roi.shape.Polygon	2005
scijava.roi.shape.PolygonSpline	2007

Continued on next page

Table 6.1 – continued from previous page

Name	TypeID
scijava.roi.shape.Polyline	2004
scijava.roi.shape.PolylineSpline	2006
scijava.roi.shape.Set	2060
scijava.roi.types.AbstractTransform1D	720
scijava.roi.types.AbstractTransform2D	721
scijava.roi.types.AbstractTransform3D	722
scijava.roi.types.AffineTransform1D	700
scijava.roi.types.AffineTransform2D	701
scijava.roi.types.AffineTransform3D	702
scijava.roi.types.AffineTransformnD	703
scijava.roi.types.AlignedBitMask1D	500
scijava.roi.types.AlignedBitMask2D	501
scijava.roi.types.AlignedBitMask3D	502
scijava.roi.types.AlignedCube1	160
scijava.roi.types.AlignedCube2	161
scijava.roi.types.AlignedCuboid1	165
scijava.roi.types.AlignedCuboid2	166
scijava.roi.types.AlignedGreyMask1D	510
scijava.roi.types.AlignedGreyMask2D	511
scijava.roi.types.AlignedGreyMask3D	512
scijava.roi.types.AlignedHalfAxes2D	220
scijava.roi.types.AlignedHalfAxes3D	221
scijava.roi.types.AlignedRectangle1	155
scijava.roi.types.AlignedRectangle2	156
scijava.roi.types.AlignedSquare1	150
scijava.roi.types.AlignedSquare2	151
scijava.roi.types.Arc12D	250
scijava.roi.types.Arc13D	251
scijava.roi.types.Arc22D	252
scijava.roi.types.Arc23D	253
scijava.roi.types.Arc32D	254
scijava.roi.types.Arc33D	255
scijava.roi.types.Array	51
scijava.roi.types.BLogic	41
scijava.roi.types.BitMask2D	520
scijava.roi.types.BitMask3D	521
scijava.roi.types.Bitwise1D	94
scijava.roi.types.Bitwise2D	95
scijava.roi.types.Bitwise3D	96
scijava.roi.types.Circle0	200
scijava.roi.types.Circle1	201
scijava.roi.types.Circle2	202
scijava.roi.types.Circle3	203
scijava.roi.types.Circle4	204
scijava.roi.types.Circle5	205
scijava.roi.types.CircularCylinder1	230

Continued on next page

Table 6.1 – continued from previous page

Name	TypeID
scijava.roi.types.CircularCylinder2	231
scijava.roi.types.CircularCylinder3	232
scijava.roi.types.CircularCylinder4	233
scijava.roi.types.Color	31
scijava.roi.types.Count	None
scijava.roi.types.Cube1	180
scijava.roi.types.Cube2	181
scijava.roi.types.Cuboid1	185
scijava.roi.types.Cuboid2	186
scijava.roi.types.Custom	None
scijava.roi.types.DimConstraint	N/A
scijava.roi.types.DimConstraintSet	3021
scijava.roi.types.DirectedGraph	54
scijava.roi.types.EllipticCylinder1	240
scijava.roi.types.EllipticCylinder2	241
scijava.roi.types.EllipticCylinder3	242
scijava.roi.types.EllipticCylinder4	243
scijava.roi.types.Extrude	91
scijava.roi.types.Float32	20
scijava.roi.types.Float64	21
scijava.roi.types.GreyMask2D	530
scijava.roi.types.GreyMask3D	531
scijava.roi.types.HalfAxes2D	225
scijava.roi.types.HalfAxes3D	226
scijava.roi.types.Index	None
scijava.roi.types.Int16	16
scijava.roi.types.Int32	17
scijava.roi.types.Int64	18
scijava.roi.types.Int8	15
scijava.roi.types.Labelling	None
scijava.roi.types.LinePoints1D	120
scijava.roi.types.LinePoints2D	121
scijava.roi.types.LinePoints3D	122
scijava.roi.types.LineVector1D	125
scijava.roi.types.LineVector2D	126
scijava.roi.types.LineVector3D	127
scijava.roi.types.LinesPoints1D	130
scijava.roi.types.LinesPoints2D	131
scijava.roi.types.LinesPoints3D	132
scijava.roi.types.LinesVectors1D	135
scijava.roi.types.LinesVectors2D	136
scijava.roi.types.LinesVectors3D	137
scijava.roi.types.Map	53
scijava.roi.types.Mesh2D	600
scijava.roi.types.Mesh3D	601
scijava.roi.types.Null	0

Continued on next page

Table 6.1 – continued from previous page

Name	TypeID
scijava.roi.types.Operator	40
scijava.roi.types.Pair	50
scijava.roi.types.Points1D	None
scijava.roi.types.Points2D	None
scijava.roi.types.Points3D	None
scijava.roi.types.PointsnD	None
scijava.roi.types.PolylinePoints1D	140
scijava.roi.types.PolylinePoints2D	141
scijava.roi.types.PolylinePoints3D	142
scijava.roi.types.PolylineVector1D	145
scijava.roi.types.PolylineVector2D	146
scijava.roi.types.PolylineVector3D	147
scijava.roi.types.Properties	4000
scijava.roi.types.Property	None
scijava.roi.types.ROI	None
scijava.roi.types.ROISet	None
scijava.roi.types.Range1nD	89
scijava.roi.types.Range2nD	90
scijava.roi.types.Rectangle1	175
scijava.roi.types.Rectangle2	176
scijava.roi.types.RotateTransform2D	716
scijava.roi.types.RotateTransform3D	717
scijava.roi.types.ScaleTransform1D	713
scijava.roi.types.ScaleTransform2D	714
scijava.roi.types.ScaleTransform3D	715
scijava.roi.types.Set	52
scijava.roi.types.ShapeSet	93
scijava.roi.types.Sphere0	210
scijava.roi.types.Sphere1	211
scijava.roi.types.Sphere2	212
scijava.roi.types.Sphere3	213
scijava.roi.types.Sphere4	214
scijava.roi.types.Sphere5	215
scijava.roi.types.Sphere6	216
scijava.roi.types.Square1	170
scijava.roi.types.Square2	171
scijava.roi.types.String	30
scijava.roi.types.Text	None
scijava.roi.types.TranslateTransform1D	710
scijava.roi.types.TranslateTransform2D	711
scijava.roi.types.TranslateTransform3D	712
scijava.roi.types.TypeID	1
scijava.roi.types.UInt16	11
scijava.roi.types.UInt32	12
scijava.roi.types.UInt64	13
scijava.roi.types.UInt8	10

Continued on next page

Table 6.1 – continued from previous page

Name	TypeID
scijava.roi.types.Value	None
scijava.roi.types.ValuenD	None
scijava.roi.types.ValuesnD	88
scijava.roi.types.Vector1D	110
scijava.roi.types.Vector2D	111
scijava.roi.types.Vector3D	112
scijava.roi.types.VectornD	113
scijava.roi.types.Vectors1D	None
scijava.roi.types.Vectors2D	None
scijava.roi.types.Vectors3D	None
scijava.roi.types.VectorsnD	None
scijava.roi.types.Vertex1D	100
scijava.roi.types.Vertex2D	101
scijava.roi.types.Vertex3D	102
scijava.roi.types.VertexList1D	None
scijava.roi.types.VertexList2D	None
scijava.roi.types.VertexList3D	None
scijava.roi.types.VertexListnD	None
scijava.roi.types.VertexnD	103

6.1 scijava.roi.Iterable

An iterable region.

Interface details

Table 6.2: scijava.roi.Iterable Details

Property	Value
TypeID	N/A

6.2 scijava.roi.RegionOfInterest

Region of interest.

Interface details

Table 6.3: scijava.roi.RegionOfInterest Details

Property	Value
TypeID	N/A
Inherits	*Serialisable*, *Iterable*

6.3 scijava.roi.RegionOfInterestSet

Set of ROIs.

Interface details

Table 6.4: scijava.roi.RegionOfInterestSet Details

Property	Value
TypeID	N/A
Inherits	*Serialisable*

6.4 scijava.roi.Serialisable

Object can be serialised and deserialised.

Interface details

Table 6.5: scijava.roi.Serialisable Details

Property	Value
TypeID	N/A

6.5 scijava.roi.annotation.Grid

A scale grid in a defined volume.

TODO: Specify grid spacing

Table 6.6: scijava.roi.annotation.Grid Details

Property	Value
TypeID	1002
Canonical representation	None
Representations in	*Cuboid*
Representations out	None

6.6 scijava.roi.annotation.Scale

A scale bar between two points.

Table 6.7: scijava.roi.annotation.Scale Details

Property	Value
TypeID	1001
Canonical representation	None
Representations in	*Line*
Representations out	None

6.7 scijava.roi.annotation.Text

Text (label).

Text in 3D will need to be based upon a rectangle in 3D (not yet possible without a transform). Should label alignment be specified directly in the representation, or in higher-level metadata?

Table 6.8: scijava.roi.annotation.Text Details

Property	Value
TypeID	1000
Canonical representation	*Text*
Representations in	*Text*
Representations out	*Text*

6.8 scijava.roi.dimconstraint.Extrude

Extrude a shape of arbitrary dimensionality into an additional dimension..

There are no limits in the additional dimension; these must be set by combining with a range instead.

Table 6.9: scijava.roi.dimconstraint.Extrude Details

Property	Value
TypeID	3010
Canonical representation	*Extrude*
Representations in	*Extrude*
Representations out	*Extrude*
Inherits	*DimConstraint*

6.9 scijava.roi.dimconstraint.Range

A range of values in an arbitrary dimension.

Constrain region to a range of values within a specific dimension.

Table 6.10: scijava.roi.dimconstraint.Range Details

Property	Value
TypeID	3002
Canonical representation	*Range1nD*
Representations in	*Range1nD*, *Range2nD*
Representations out	*Range1nD*, *Range2nD*
Inherits	*DimConstraint*

6.10 scijava.roi.dimconstraint.Set

Combine shapes of differing dimensionality.

The result is a shape combining all subset dimensions. It is illegal to have a common dimension between the two shapes.

Table 6.11: scijava.roi.dimconstraint.Set Details

Property	Value
TypeID	3011
Canonical representation	*DimConstraintSet*
Representations in	*DimConstraintSet*
Representations out	*DimConstraintSet*
Inherits	*DimConstraint*

6.11 scijava.roi.dimconstraint.Value

A value in an arbitrary dimension.

Constrain region to a single value within a specific dimension.

Table 6.12: scijava.roi.dimconstraint.Value Details

Property	Value
TypeID	3000
Canonical representation	*ValuenD*
Representations in	*ValuenD*
Representations out	*ValuenD*
Inherits	*DimConstraint*

6.12 scijava.roi.dimconstraint.Values

A set of values in an arbitrary dimension.

Constrain region to multiple values within a specific dimension.

Table 6.13: scijava.roi.dimconstraint.Values Details

Property	Value
TypeID	3001
Canonical representation	*ValuesnD*
Representations in	*ValuesnD*
Representations out	*ValuesnD*
Inherits	*DimConstraint*

6.13 scijava.roi.measurement.Area

Get area and perimeter of object.

Interface details

Table 6.14: scijava.roi.measurement.Area Details

Property	Value
TypeID	N/A

6.14 scijava.roi.measurement.Length

Get length of object.

Interface details

Table 6.15: scijava.roi.measurement.Length Details

Property	Value
TypeID	N/A

6.15 scijava.roi.measurement.Volume

Get volume and surface area of object.

Interface details

Table 6.16: scijava.roi.measurement.Volume Details

Property	Value
TypeID	N/A

6.16 scijava.roi.shape.AbstractTransform

Abstract (implementation-defined) transformation of a shape.

Table 6.17: scijava.roi.shape.AbstractTransform Details

Property	Value
TypeID	2051
Canonical representation	*AbstractTransform3D*
Representations in	*AbstractTransform1D*, *AbstractTransform2D*, *AbstractTransform3D*
Representations out	*AbstractTransform1D*, *AbstractTransform2D*, *AbstractTransform3D*
Inherits	*PhysicalShape*

6.17 scijava.roi.shape.AffineTransform

Affine transformation of a shape.

Table 6.18: scijava.roi.shape.AffineTransform Details

Property	Value
TypeID	2050
Canonical representation	*AffineTransform3D*
Representations in	*ScaleTransform1D*, *TranslateTransform1D*, *ScaleTransform2D*, *ScaleTransform3D*, *RotateTransform2D*, *RotateTransform3D*, *AffineTransform2D*, *AffineTransform3D*, *AffineTransform1D*, *TranslateTransform2D*, *TranslateTransform3D*
Representations out	*AffineTransform1D*, *AffineTransform2D*, *AffineTransform3D*
Inherits	*PhysicalShape*

6.18 scijava.roi.shape.Arc

An arc.

Table 6.19: scijava.roi.shape.Arc Details

Property	Value
TypeID	2008
Canonical representation	*Arc13D*
Representations in	*Arc22D*, *Arc23D*, *Arc33D*, *Arc13D*, *Arc12D*, *Arc32D*
Representations out	*Arc22D*, *Arc23D*, *Arc33D*, *Arc13D*, *Arc12D*, *Arc32D*
Inherits	*PhysicalShape*, *Length*

6.19 scijava.roi.shape.BitMask

A mask with one bit values.

A bitmask may be aligned with the axes (with an aligned bounding box) or unaligned (with an unaligned bounding box). In order to iterate over the mask with a 1:1 correspondence between mask and underlying image pixel data, it must be converted to an aligned form. Additionally, it must be converted to an aligned form with the samples aligned with the pixel grid.

Table 6.20: scijava.roi.shape.BitMask Details

Property	Value
TypeID	2040
Canonical representation	*BitMask3D*
Representations in	*AlignedBitMask1D*, *AlignedBitMask2D*, *AlignedBitMask3D*, *BitMask3D*, *BitMask2D*
Representations out	*AlignedBitMask1D*, *AlignedBitMask2D*, *AlignedBitMask3D*, *BitMask3D*, *BitMask2D*
Inherits	*PhysicalShape*

6.20 scijava.roi.shape.Bitwise

Binary bitwise operation.

Table 6.21: scijava.roi.shape.Bitwise Details

Property	Value
TypeID	2052
Canonical representation	*Bitwise3D*
Representations in	*Bitwise1D*, *Bitwise2D*, *Bitwise3D*
Representations out	*Bitwise1D*, *Bitwise2D*, *Bitwise3D*
Inherits	*PhysicalShape*

6.21 scijava.roi.shape.Cuboid

A cuboid.

Table 6.22: scijava.roi.shape.Cuboid Details

Property	Value
TypeID	2020
Canonical representation	*Cuboid1*
Representations in	*Cube2*, *AlignedSquare2*, *AlignedCube2*, *Rectangle1*, *AlignedRectangle2*, *Square1*, *AlignedCuboid2*, *AlignedRectangle1*, *Cube1*, *AlignedSquare1*, *Square2*, *AlignedCube1*, *Rectangle2*, *Cuboid1*, *Cuboid2*, *AlignedCuboid1*
Representations out	*Cube2*, *AlignedSquare2*, *AlignedCube2*, *Rectangle1*, *AlignedRectangle2*, *Square1*, *AlignedCuboid2*, *AlignedRectangle1*, *Cube1*, *AlignedSquare1*, *Square2*, *AlignedCube1*, *Rectangle2*, *Cuboid1*, *Cuboid2*, *AlignedCuboid1*
Inherits	*PhysicalShape*, *Volume*

6.22 scijava.roi.shape.Custom

A custom (user-definable) 3D shape.

The custom shape type, unlike other shapes, does not define any intrinsic behaviour. This is entirely the responsibility of the user. The typename of the shape is specified by the user, which provides an extension mechanism by allowing this type to be used to specify an arbitrary number of shape types. The shape contains four sets of shapes for measurement, results, editing and visualisation. The intent here is that the shapes required for the user to visualise the ROI are contained in the VISUAL set. This will permit the ROI to be transported to other systems, and allow visualisation without any knowledge of the specific ROI type. The other types are optional, and may be used as the user sees fit. MEASUREMENTS is intended to store any points or other informations used when defining the ROI (which are not already contained in the VISUAL set). RESULTS is intended to store any measurements which are not directly derivable from the other sets. EDIT is intended for storing label offsets, construction lines, and any other information used for editing which is not contained in the MEASUREMENTS or VISUAL sets.

Table 6.23: scijava.roi.shape.Custom Details

Property	Value
TypeID	650
Canonical representation	*Custom*
Representations in	*Custom*
Representations out	*Custom*
Inherits	*PhysicalShape*

6.23 scijava.roi.shape.Cylinder

An elliptic cylinder.

Table 6.24: scijava.roi.shape.Cylinder Details

Property	Value
TypeID	2022
Canonical representation	*EllipticCylinder1*
Representations in	*CircularCylinder1*, *Circle1*, *Circle2*, *CircularCylinder3*, *Circle4*, *Circle5*, *CircularCylinder4*, *EllipticCylinder3*, *Circle0*, *EllipticCylinder1*, *EllipticCylinder2*, *CircularCylinder2*, *Circle3*, *AlignedHalfAxes2D*, *HalfAxes2D*, *EllipticCylinder4*
Representations out	*CircularCylinder1*, *Circle1*, *Circle2*, *CircularCylinder3*, *Circle4*, *Circle5*, *CircularCylinder4*, *EllipticCylinder3*, *Circle0*, *EllipticCylinder1*, *EllipticCylinder2*, *CircularCylinder2*, *Circle3*, *AlignedHalfAxes2D*, *HalfAxes2D*, *EllipticCylinder4*
Inherits	*PhysicalShape*, *Volume*

6.24 scijava.roi.shape.Ellipsoid

An ellipsoid.

Table 6.25: scijava.roi.shape.Ellipsoid Details

Property	Value
TypeID	2021
Canonical representation	*HalfAxes3D*
Representations in	*Sphere1*, *Sphere0*, *Sphere3*, *Sphere4*, *Cuboid*, *HalfAxes3D*, *Sphere5*, *Sphere6*, *AlignedHalfAxes3D*, *Sphere2*
Representations out	*Sphere1*, *Sphere0*, *Sphere3*, *Sphere4*, *Cuboid*, *HalfAxes3D*, *Sphere5*, *Sphere6*, *AlignedHalfAxes3D*, *Sphere2*
Inherits	*PhysicalShape*, *Volume*

6.25 scijava.roi.shape.GreyMask

A mask with multiple grey levels.

A greymask may be aligned with the axes (with an aligned bounding box) or unaligned (with an unaligned bounding box). In order to iterate over the mask with a 1:1 correspondence between mask and underlying image pixel data, it must be converted to an aligned form. Additionally, it must be converted to an aligned form with the samples aligned with the pixel grid.

Table 6.26: scijava.roi.shape.GreyMask Details

Property	Value
TypeID	2041
Canonical representation	*GreyMask3D*
Representations in	*AlignedGreyMask1D*, *AlignedGreyMask2D*, *GreyMask2D*, *AlignedGreyMask3D*, *GreyMask3D*
Representations out	*AlignedGreyMask1D*, *AlignedGreyMask2D*, *GreyMask2D*, *AlignedGreyMask3D*, *GreyMask3D*
Inherits	*PhysicalShape*

6.26 scijava.roi.shape.Line

A single line.

Table 6.27: scijava.roi.shape.Line Details

Property	Value
TypeID	2002
Canonical representation	*LinePoints3D*
Representations in	*LineVector3D*, *LineVector1D*, *LinePoints1D*, *LinePoints2D*, *LinePoints3D*, *LineVector2D*
Representations out	*LineVector3D*, *LineVector1D*, *LinePoints1D*, *LinePoints2D*, *LinePoints3D*, *LineVector2D*
Inherits	*PhysicalShape*, *Length*

6.27 scijava.roi.shape.Lines

A set of lines.

Table 6.28: scijava.roi.shape.Lines Details

Property	Value
TypeID	2003
Canonical representation	*LinesPoints3D*
Representations in	*Line*, *LinesVectors1D*, *LinesPoints2D*, *LinesPoints3D*, *LinesPoints1D*, *LinesVectors2D*, *LinesVectors3D*
Representations out	*LinesVectors1D*, *LinesPoints2D*, *LinesPoints3D*, *LinesPoints1D*, *LinesVectors2D*, *LinesVectors3D*
Inherits	*PhysicalShape*, *Length*

6.28 scijava.roi.shape.Mesh

A mesh.

Table 6.29: scijava.roi.shape.Mesh Details

Property	Value
TypeID	2023
Canonical representation	*Mesh3D*
Representations in	*Mesh2D*, *Mesh3D*
Representations out	*Mesh2D*, *Mesh3D*
Inherits	*PhysicalShape*, *Area*, *Volume*

6.29 scijava.roi.shape.PhysicalShape

Abstract shape.

Interface details

Table 6.30: scijava.roi.shape.PhysicalShape Details

Property	Value
TypeID	N/A
Inherits	*Serialisable*

6.30 scijava.roi.shape.Point

A single point.

Table 6.31: scijava.roi.shape.Point Details

Property	Value
TypeID	2000
Canonical representation	*Vertex3D*
Representations in	*Vertex1D*, *Vertex2D*, *Vertex3D*
Representations out	*Vertex1D*, *Vertex2D*, *Vertex3D*
Inherits	*PhysicalShape*

6.31 scijava.roi.shape.Points

A set of points.

Table 6.32: scijava.roi.shape.Points Details

Property	Value
TypeID	2001
Canonical representation	*Points3D*
Representations in	*Points1D*, *Points2D*, *Points3D*
Representations out	*Points1D*, *Points2D*, *Points3D*
Inherits	*PhysicalShape*

6.32 scijava.roi.shape.Polygon

A set of connected points (closed).

Table 6.33: scijava.roi.shape.Polygon Details

Property	Value
TypeID	2005
Canonical representation	*PolylinePoints3D*
Representations in	*PolylineVector1D*, *PolylinePoints1D*, *PolylinePoints2D*, *PolylinePoints3D*, *PolylineVector2D*, *PolylineVector3D*
Representations out	*PolylineVector1D*, *PolylinePoints1D*, *PolylinePoints2D*, *PolylinePoints3D*, *PolylineVector2D*, *PolylineVector3D*
Inherits	*PhysicalShape*

6.33 scijava.roi.shape.PolygonSpline

A set of connected splines (closed).

Table 6.34: scijava.roi.shape.PolygonSpline Details

Property	Value
TypeID	2007
Canonical representation	*PolylinePoints3D*
Representations in	*PolylinePoints2D*, *PolylineVector2D*, *PolylineVector3D*, *PolylinePoints3D*
Representations out	*PolylinePoints2D*, *PolylineVector2D*, *PolylineVector3D*, *PolylinePoints3D*
Inherits	*PhysicalShape*, *Area*

6.34 scijava.roi.shape.Polyline

A set of connected points (open).

Table 6.35: scijava.roi.shape.Polyline Details

Property	Value
TypeID	2004
Canonical representation	*PolylinePoints3D*
Representations in	*PolylineVector1D*, *PolylinePoints1D*, *PolylinePoints2D*, *PolylinePoints3D*, *PolylineVector2D*, *PolylineVector3D*
Representations out	*PolylineVector1D*, *PolylinePoints1D*, *PolylinePoints2D*, *PolylinePoints3D*, *PolylineVector2D*, *PolylineVector3D*
Inherits	*PhysicalShape*, *Length*, *Area*

6.35 scijava.roi.shape.PolylineSpline

A set of connected splines (open).

Table 6.36: scijava.roi.shape.PolylineSpline Details

Property	Value
TypeID	2006
Canonical representation	*PolylinePoints3D*
Representations in	*PolylinePoints2D*, *PolylineVector2D*, *PolylineVector3D*, *PolylinePoints3D*
Representations out	*PolylinePoints2D*, *PolylineVector2D*, *PolylineVector3D*, *PolylinePoints3D*
Inherits	*PhysicalShape*, *Length*

6.36 scijava.roi.shape.Set

A set of shapes.

All operations operate individually upon the contained shapes. This implies that transforms are performed upon each shape, with rotation centres in the centre of each shape.

Table 6.37: scijava.roi.shape.Set Details

Property	Value
TypeID	2060
Canonical representation	*ShapeSet*
Representations in	*ShapeSet*
Representations out	*ShapeSet*
Inherits	*PhysicalShape*

6.37 scijava.roi.types.AbstractTransform1D

An abstract (implementation-defined) transform in 1D.

Serialisation compound structure

Table 6.38: scijava.roi.types.AbstractTransform1D Details

Property	Value
TypeID	720

6.38 scijava.roi.types.AbstractTransform2D

An abstract (implementation-defined) transform in 2D.

Serialisation compound structure

Table 6.39: scijava.roi.types.AbstractTransform2D Details

Property	Value
TypeID	721

6.39 scijava.roi.types.AbstractTransform3D

An abstract (implementation-defined) transform in 3D.

Serialisation compound structure

Table 6.40: scijava.roi.types.AbstractTransform3D Details

Property	Value
TypeID	722

6.40 scijava.roi.types.AffineTransform1D

An affine transform in 1D described by a transformation matrix and 1D shape to transform.

Serialisation compound structure

Table 6.41: scijava.roi.types.AffineTransform1D Details

Property	Value
TypeID	700

6.41 scijava.roi.types.AffineTransform2D

An affine transform in 2D described by a transformation matrix and 2D shape to transform.

Serialisation compound structure

Table 6.42: scijava.roi.types.AffineTransform2D Details

Property	Value
TypeID	701

6.42 scijava.roi.types.AffineTransform3D

An affine transform in 3D described by a transformation matrix and 3D shape to transform.

Serialisation compound structure

Table 6.43: scijava.roi.types.AffineTransform3D Details

Property	Value
TypeID	702

6.43 scijava.roi.types.AffineTransformnD

An affine transform in nD described by a transformation matrix and nD shape to transform.

Table 6.44: scijava.roi.types.AffineTransformnD Details

Property	Value
TypeID	703

6.44 scijava.roi.types.AlignedBitMask1D

A bitmask in 1D described by aligned bounding line, dimensions and mask data.

The mask is applied to the bounding line. Dimensions specify the x size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.45: scijava.roi.types.AlignedBitMask1D Details

Property	Value
TypeID	500

6.45 scijava.roi.types.AlignedBitMask2D

A bitmask in 2D described by aligned bounding rectangle, dimensions and mask data.

The mask is applied to the aligned bounding rectangle. Dimensions specify the x and y size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.46: scijava.roi.types.AlignedBitMask2D Details

Property	Value
TypeID	501

6.46 scijava.roi.types.AlignedBitMask3D

A bitmask in 3D described by aligned bounding cuboid, dimensions and mask data.

The mask is applied to the aligned bounding cuboid. Dimensions specify the x, y and z size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.47: scijava.roi.types.AlignedBitMask3D Details

Property	Value
TypeID	502

6.47 scijava.roi.types.AlignedCube1

A cube in 3D aligned with the axes described by a corner point and adjacent corner.

Serialisation compound structure

Table 6.48: scijava.roi.types.AlignedCube1 Details

Property	Value
TypeID	160

6.48 scijava.roi.types.AlignedCube2

A cube in 3D aligned with the axes described by a corner point and vector to an adjacent corner.

Serialisation compound structure

Table 6.49: scijava.roi.types.AlignedCube2 Details

Property	Value
TypeID	161

6.49 scijava.roi.types.AlignedCuboid1

An aligned cuboid described by two points in 3D.

Serialisation compound structure

Table 6.50: scijava.roi.types.AlignedCuboid1 Details

Property	Value
TypeID	165

6.50 scijava.roi.types.AlignedCuboid2

An aligned cuboid described by a point and a vector.

Serialisation compound structure

Table 6.51: scijava.roi.types.AlignedCuboid2 Details

Property	Value
TypeID	166

6.51 scijava.roi.types.AlignedGreyMask1D

A greymask in 1D described by aligned bounding line, dimensions and mask data.

The mask is applied to the aligned bounding line. Dimensions specify the x size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.52: scijava.roi.types.AlignedGreyMask1D Details

Property	Value
TypeID	510

6.52 scijava.roi.types.AlignedGreyMask2D

A greymask in 2D described by aligned bounding rectangle, dimensions and mask data.

The mask is applied to the aligned bounding rectangle. Dimensions specify the x and y size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.53: scijava.roi.types.AlignedGreyMask2D Details

Property	Value
TypeID	511

6.53 scijava.roi.types.AlignedGreyMask3D

A greymask in 3D described by aligned bounding cuboid, dimensions and mask data.

The mask is applied to the aligned bounding cuboid. Dimensions specify the x, y and z size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.54: scijava.roi.types.AlignedGreyMask3D Details

Property	Value
TypeID	512

6.54 scijava.roi.types.AlignedHalfAxes2D

An ellipse in 2D aligned with the axes described by two half axes.

Serialisation compound structure

Table 6.55: scijava.roi.types.AlignedHalfAxes2D Details

Property	Value
TypeID	220

6.55 scijava.roi.types.AlignedHalfAxes3D

An ellipsoid in 3D aligned with the axes.

Serialisation compound structure

Table 6.56: scijava.roi.types.AlignedHalfAxes3D Details

Property	Value
TypeID	221

6.56 scijava.roi.types.AlignedRectangle1

An aligned rectangle described by two points in 2D.

Serialisation compound structure

Table 6.57: scijava.roi.types.AlignedRectangle1 Details

Property	Value
TypeID	155

6.57 scijava.roi.types.AlignedRectangle2

An aligned rectangle described by a point and a vector.

Serialisation compound structure

Table 6.58: scijava.roi.types.AlignedRectangle2 Details

Property	Value
TypeID	156

6.58 scijava.roi.types.AlignedSquare1

A square in 2D aligned with the axes described by a corner point and adjacent corner.

Serialisation compound structure

Table 6.59: scijava.roi.types.AlignedSquare1 Details

Property	Value
TypeID	150

6.59 scijava.roi.types.AlignedSquare2

A square in 2D aligned with the axes described by a corner point and vector to an adjacent corner.

Serialisation compound structure

Table 6.60: scijava.roi.types.AlignedSquare2 Details

Property	Value
TypeID	151

6.60 scijava.roi.types.Arc12D

An arc in 2D described by a line (points) and vector.

Serialisation compound structure

Table 6.61: scijava.roi.types.Arc12D Details

Property	Value
TypeID	250

6.61 scijava.roi.types.Arc13D

An arc in 3D described by a line (points) and vector.

Serialisation compound structure

Table 6.62: scijava.roi.types.Arc13D Details

Property	Value
TypeID	251

6.62 scijava.roi.types.Arc22D

An arc in 2D described by a line (vector) and a vector.

Serialisation compound structure

Table 6.63: scijava.roi.types.Arc22D Details

Property	Value
TypeID	252

6.63 scijava.roi.types.Arc23D

An arc in 3D described by a line (vector) and a vector.

Serialisation compound structure

Table 6.64: scijava.roi.types.Arc23D Details

Property	Value
TypeID	253

6.64 scijava.roi.types.Arc32D

An arc in 2D described by three points; vector inferred from third point.

Serialisation compound structure

Table 6.65: scijava.roi.types.Arc32D Details

Property	Value
TypeID	254

6.65 scijava.roi.types.Arc33D

An arc in 3D described by three points; vector inferred from third point.

Serialisation compound structure

Table 6.66: scijava.roi.types.Arc33D Details

Property	Value
TypeID	255

6.66 scijava.roi.types.Array

Fixed length ordered array.

Serialisation compound structure

Table 6.67: scijava.roi.types.Array Details

Property	Value
TypeID	51

6.67 scijava.roi.types.BLogic

.

Enumeration values

Table 6.68: scijava.roi.types.BLogic Details

Property	Value
TypeID	41

6.68 scijava.roi.types.BitMask2D

A bitmask in 2D described by bounding rectangle, dimensions and mask data.

The mask is applied to the bounding rectangle. Dimensions specify the x and y size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.69: scijava.roi.types.BitMask2D Details

Property	Value
TypeID	520

6.69 scijava.roi.types.BitMask3D

A bitmask in 3D described by bounding cuboid, dimensions and mask data.

The mask is applied to the bounding cuboid. Dimensions specify the x, y and z size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.70: scijava.roi.types.BitMask3D Details

Property	Value
TypeID	521

6.70 scijava.roi.types.Bitwise1D

Binary bitwise operation.

Serialisation compound structure

Table 6.71: scijava.roi.types.Bitwise1D Details

Property	Value
TypeID	94

6.71 scijava.roi.types.Bitwise2D

Binary bitwise operation.

Serialisation compound structure

Table 6.72: scijava.roi.types.Bitwise2D Details

Property	Value
TypeID	95

6.72 scijava.roi.types.Bitwise3D

Binary bitwise operation.

Serialisation compound structure

Table 6.73: scijava.roi.types.Bitwise3D Details

Property	Value
TypeID	96

6.73 scijava.roi.types.Circle0

A circle in 2D described by a centre point and circumference point.

Serialisation compound structure

Table 6.74: scijava.roi.types.Circle0 Details

Property	Value
TypeID	200

6.74 scijava.roi.types.Circle1

A circle in 2D described by a centre point and 1D radius.

Serialisation compound structure

Table 6.75: scijava.roi.types.Circle1 Details

Property	Value
TypeID	201

6.75 scijava.roi.types.Circle2

A circle in 2D described by a centre point and 2D radius.

Serialisation compound structure

Table 6.76: scijava.roi.types.Circle2 Details

Property	Value
TypeID	202

6.76 scijava.roi.types.Circle3

A circle in 2D described by a circumference point and vector to the centre point.

Serialisation compound structure

Table 6.77: scijava.roi.types.Circle3 Details

Property	Value
TypeID	203

6.77 scijava.roi.types.Circle4

A circle in 2D described by two circumference points [diameter].

Serialisation compound structure

Table 6.78: scijava.roi.types.Circle4 Details

Property	Value
TypeID	204

6.78 scijava.roi.types.Circle5

A circle in 2D described by three circumference points.

Serialisation compound structure

Table 6.79: scijava.roi.types.Circle5 Details

Property	Value
TypeID	205

6.79 scijava.roi.types.CircularCylinder1

A circular cylinder in 3D described by the centres of both faces and a radius.

A basic circular cylinder with faces at right angles.

Serialisation compound structure

Table 6.80: scijava.roi.types.CircularCylinder1 Details

Property	Value
TypeID	230

6.80 scijava.roi.types.CircularCylinder2

A circular cylinder in 3D described by the centre of one face, vector to second face and a radius.

A basic circular cylinder with faces at right angles.

Serialisation compound structure

Table 6.81: scijava.roi.types.CircularCylinder2 Details

Property	Value
TypeID	231

6.81 scijava.roi.types.CircularCylinder3

A circular cylinder in 3D with faces at different angles described by the centres of both faces and vectors specifying the radius and angles of the faces.

Face angles other than right-angles let chains of cylinders be used for tubular structures without gaps at the joins.

Serialisation compound structure

Table 6.82: scijava.roi.types.CircularCylinder3 Details

Property	Value
TypeID	232

6.82 scijava.roi.types.CircularCylinder4

A circular cylinder in 3D with faces at different angles described by the centre of one face, vector to second face and vectors specifying the radius and angles of the faces.

Face angles other than right-angles let chains of cylinders be used for tubular structures without gaps at the joins.

Serialisation compound structure

Table 6.83: scijava.roi.types.CircularCylinder4 Details

Property	Value
TypeID	233

6.83 scijava.roi.types.Color

Color in RGBA (0,1) range.

double[4] = 32 bytes; a more compact representation could be used

Serialisation compound structure

Table 6.84: scijava.roi.types.Color Details

Property	Value
TypeID	31

6.84 scijava.roi.types.Count

Number of objects.

Table 6.85: scijava.roi.types.Count Details

Property	Value
TypeID	None

6.85 scijava.roi.types.Cube1

An aligned cuboid described by two points in 3D.

Serialisation compound structure

Table 6.86: scijava.roi.types.Cube1 Details

Property	Value
TypeID	180

6.86 scijava.roi.types.Cube2

An aligned cuboid described by a point and a vector.

Serialisation compound structure

Table 6.87: scijava.roi.types.Cube2 Details

Property	Value
TypeID	181

6.87 scijava.roi.types.Cuboid1

A cuboid in 3D described by two adjacent corners and two vectors.

Serialisation compound structure

Table 6.88: scijava.roi.types.Cuboid1 Details

Property	Value
TypeID	185

6.88 scijava.roi.types.Cuboid2

A cuboid in 3D described by a corner and three vectors.

Serialisation compound structure

Table 6.89: scijava.roi.types.Cuboid2 Details

Property	Value
TypeID	186

6.89 scijava.roi.types.Custom

Custom (user-definable) representation.

Serialisation compound structure

Table 6.90: scijava.roi.types.Custom Details

Property	Value
TypeID	None

6.90 scijava.roi.types.DimConstraint

Abstract dimensional constraints.

Interface details

Table 6.91: scijava.roi.types.DimConstraint Details

Property	Value
TypeID	N/A
Inherits	*Serialisable*

6.91 scijava.roi.types.DimConstraintSet

A set of dimensional constraints.

Serialisation compound structure

Table 6.92: scijava.roi.types.DimConstraintSet Details

Property	Value
TypeID	3021

6.92 scijava.roi.types.DirectedGraph

Fixed length directed graph.

Serialisation compound structure

Table 6.93: scijava.roi.types.DirectedGraph Details

Property	Value
TypeID	54

6.93 scijava.roi.types.EllipticCylinder1

An elliptic cylinder in 3D described by the centres both faces and half axes.

A basic elliptic cylinder with faces at right angles.

Serialisation compound structure

Table 6.94: scijava.roi.types.EllipticCylinder1 Details

Property	Value
TypeID	240

6.94 scijava.roi.types.EllipticCylinder2

An elliptic cylinder in 3D described by the centre of one face, vector to second face and half axes.

A basic elliptic cylinder with faces at right angles.

Serialisation compound structure

Table 6.95: scijava.roi.types.EllipticCylinder2 Details

Property	Value
TypeID	241

6.95 scijava.roi.types.EllipticCylinder3

An elliptic cylinder in 3D with faces at different angles described by the centres both faces and half axes and angles.

Face angles other than right-angles let chains of cylinders be used for tubular structures without gaps at the joins.

Serialisation compound structure

Table 6.96: scijava.roi.types.EllipticCylinder3 Details

Property	Value
TypeID	242

6.96 scijava.roi.types.EllipticCylinder4

An elliptic cylinder in 3D with faces at different angles described by the centre of one face, vector to second face and half axes and angles.

Face angles other than right-angles let chains of cylinders be used for tubular structures without gaps at the joins.

Serialisation compound structure

Table 6.97: scijava.roi.types.EllipticCylinder4 Details

Property	Value
TypeID	243

6.97 scijava.roi.types.Extrude

A shape extruded in an additional dimension.

Serialisation compound structure

Table 6.98: scijava.roi.types.Extrude Details

Property	Value
TypeID	91

6.98 scijava.roi.types.Float32

Single precision floating point number.

Table 6.99: scijava.roi.types.Float32 Details

Property	Value
TypeID	20

6.99 scijava.roi.types.Float64

Double precision floating point number.

Table 6.100: scijava.roi.types.Float64 Details

Property	Value
TypeID	21

6.100 scijava.roi.types.GreyMask2D

A greymask in 2D described by bounding rectangle, dimensions and mask data.

The mask is applied to the bounding rectangle. Dimensions specify the x and y size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.101: scijava.roi.types.GreyMask2D Details

Property	Value
TypeID	530

6.101 scijava.roi.types.GreyMask3D

A greymask in 3D described by bounding cuboid, dimensions and mask data.

The mask is applied to the bounding cuboid. Dimensions specify the x, y and z size of the mask. DATA is the mask pixel data.

Serialisation compound structure

Table 6.102: scijava.roi.types.GreyMask3D Details

Property	Value
TypeID	531

6.102 scijava.roi.types.HalfAxes2D

An ellipse in 2D described by two half axes.

Serialisation compound structure

Table 6.103: scijava.roi.types.HalfAxes2D Details

Property	Value
TypeID	225

6.103 scijava.roi.types.HalfAxes3D

An ellipsoid in 3D described by three half axes.

Serialisation compound structure

Table 6.104: scijava.roi.types.HalfAxes3D Details

Property	Value
TypeID	226

6.104 scijava.roi.types.Index

Index into an array.

Table 6.105: scijava.roi.types.Index Details

Property	Value
TypeID	None

6.105 scijava.roi.types.Int16

Signed 16-bit integer.

Table 6.106: scijava.roi.types.Int16 Details

Property	Value
TypeID	16

6.106 scijava.roi.types.Int32

Signed 32-bit integer.

Table 6.107: scijava.roi.types.Int32 Details

Property	Value
TypeID	17

6.107 scijava.roi.types.Int64

Signed 64-bit integer.

Table 6.108: scijava.roi.types.Int64 Details

Property	Value
TypeID	18

6.108 scijava.roi.types.Int8

Signed 8-bit integer.

Table 6.109: scijava.roi.types.Int8 Details

Property	Value
TypeID	15

6.109 scijava.roi.types.Labelling

A labelling (collection of bitmasks).

Table 6.110: scijava.roi.types.Labelling Details

Property	Value
TypeID	None
Inherits	*RegionOfInterestSet*

6.110 scijava.roi.types.LinePoints1D

A line described by two points in 1D.

Serialisation compound structure

Table 6.111: scijava.roi.types.LinePoints1D Details

Property	Value
TypeID	120

6.111 scijava.roi.types.LinePoints2D

A line described by two points in 2D.

Serialisation compound structure

Table 6.112: scijava.roi.types.LinePoints2D Details

Property	Value
TypeID	121

6.112 scijava.roi.types.LinePoints3D

A line described by two points in 3D.

Serialisation compound structure

Table 6.113: scijava.roi.types.LinePoints3D Details

Property	Value
TypeID	122

6.113 scijava.roi.types.LineVector1D

A line described by a point and a vector.

Serialisation compound structure

Table 6.114: scijava.roi.types.LineVector1D Details

Property	Value
TypeID	125

6.114 scijava.roi.types.LineVector2D

A line described by a point and a vector.

Serialisation compound structure

Table 6.115: scijava.roi.types.LineVector2D Details

Property	Value
TypeID	126

6.115 scijava.roi.types.LineVector3D

A line described by a point and a vector.

Serialisation compound structure

Table 6.116: scijava.roi.types.LineVector3D Details

Property	Value
TypeID	127

6.116 scijava.roi.types.LinesPoints1D

A list of lines described by two points in 1D.

Serialisation compound structure

Table 6.117: scijava.roi.types.LinesPoints1D Details

Property	Value
TypeID	130

6.117 scijava.roi.types.LinesPoints2D

A list of lines described by two points in 2D.

Serialisation compound structure

Table 6.118: scijava.roi.types.LinesPoints2D Details

Property	Value
TypeID	131

6.118 scijava.roi.types.LinesPoints3D

A list of lines described by two points in 3D.

Serialisation compound structure

Table 6.119: scijava.roi.types.LinesPoints3D Details

Property	Value
TypeID	132

6.119 scijava.roi.types.LinesVectors1D

A list of lines described by a point and a vector in 1D; can be used to represent a vector field.

Serialisation compound structure

Table 6.120: scijava.roi.types.LinesVectors1D Details

Property	Value
TypeID	135

6.120 scijava.roi.types.LinesVectors2D

A list of lines described by a point and a vector in 2D; can be used to represent a vector field.

Serialisation compound structure

Table 6.121: scijava.roi.types.LinesVectors2D Details

Property	Value
TypeID	136

6.121 scijava.roi.types.LinesVectors3D

A list of lines described by a point and a vector in 3D; can be used to represent a vector field.

Serialisation compound structure

Table 6.122: scijava.roi.types.LinesVectors3D Details

Property	Value
TypeID	137

6.122 scijava.roi.types.Map

Fixed length unordered map.

Serialisation compound structure

Table 6.123: scijava.roi.types.Map Details

Property	Value
TypeID	53

6.123 scijava.roi.types.Mesh2D

A face-vertex mesh in 2D described by face and vertex lists.

Vertex references are indexes into the VERTS array. Vertex-face mapping is implied, and will require the implementor to construct the mapping.

Serialisation compound structure

Table 6.124: scijava.roi.types.Mesh2D Details

Property	Value
TypeID	600

6.124 scijava.roi.types.Mesh3D

A face-vertex mesh in 3D described by face and vertex lists.

Vertex references are indexes into the VERTS array. Vertex-face mapping is implied, and will require the implementor to construct the mapping.

Serialisation compound structure

Table 6.125: scijava.roi.types.Mesh3D Details

Property	Value
TypeID	601

6.125 scijava.roi.types.Null

Null type (used to indicate the absence of optional type).

Table 6.126: scijava.roi.types.Null Details

Property	Value
TypeID	0

6.126 scijava.roi.types.Operator

.

Enumeration values

Table 6.127: scijava.roi.types.Operator Details

Property	Value
TypeID	40

6.127 scijava.roi.types.Pair

Pair of values (for map and graph containers).

Serialisation compound structure

Table 6.128: scijava.roi.types.Pair Details

Property	Value
TypeID	50

6.128 scijava.roi.types.Points1D

A list of points in 1D.

Serialisation compound structure

Table 6.129: scijava.roi.types.Points1D Details

Property	Value
TypeID	None

6.129 scijava.roi.types.Points2D

A list of points in 2D.

Serialisation compound structure

Table 6.130: scijava.roi.types.Points2D Details

Property	Value
TypeID	None

6.130 scijava.roi.types.Points3D

A list of points in 3D.

Serialisation compound structure

Table 6.131: scijava.roi.types.Points3D Details

Property	Value
TypeID	None

6.131 scijava.roi.types.PointsnD

A list of points in nD.

Table 6.132: scijava.roi.types.PointsnD Details

Property	Value
TypeID	None

6.132 scijava.roi.types.PolylinePoints1D

A list of points in a polyline in 1D [could use RPoints1D directly].

Serialisation compound structure

Table 6.133: scijava.roi.types.PolylinePoints1D Details

Property	Value
TypeID	140

6.133 scijava.roi.types.PolylinePoints2D

A list of points in a polyline in 2D [could use RPoints2D directly].

Serialisation compound structure

Table 6.134: scijava.roi.types.PolylinePoints2D Details

Property	Value
TypeID	141

6.134 scijava.roi.types.PolylinePoints3D

A list of points in a polyline in 3D [could use RPoints3D directly].

Serialisation compound structure

Table 6.135: scijava.roi.types.PolylinePoints3D Details

Property	Value
TypeID	142

6.135 scijava.roi.types.PolylineVector1D

A list of points in a polyline represented by a starting point and list of vectors in 1D.

Serialisation compound structure

Table 6.136: scijava.roi.types.PolylineVector1D Details

Property	Value
TypeID	145

6.136 scijava.roi.types.PolylineVector2D

A list of points in a polyline represented by a starting point and list of vectors in 2D.

Serialisation compound structure

Table 6.137: scijava.roi.types.PolylineVector2D Details

Property	Value
TypeID	146

6.137 scijava.roi.types.PolylineVector3D

A list of points in a polyline represented by a starting point and list of vectors in 3D.

Serialisation compound structure

Table 6.138: scijava.roi.types.PolylineVector3D Details

Property	Value
TypeID	147

6.138 scijava.roi.types.Properties

Property list.

We could use an RShape representation here so that we could set a shape as a property.

Serialisation compound structure

Table 6.139: scijava.roi.types.Properties Details

Property	Value
TypeID	4000

6.139 scijava.roi.types.Property

A custom (user-definable) object property.

Serialisation compound structure

Table 6.140: scijava.roi.types.Property Details

Property	Value
TypeID	None

6.140 scijava.roi.types.ROI

A region of interest (top-level container of physical shape and nD constraints).

Table 6.141: scijava.roi.types.ROI Details

Property	Value
TypeID	None
Inherits	*RegionOfInterest*

6.141 scijava.roi.types.ROISet

A set of ROIs.

Table 6.142: scijava.roi.types.ROISet Details

Property	Value
TypeID	None
Inherits	*RegionOfInterestSet*

6.142 scijava.roi.types.Range1nD

A range of values specified as the half-open range [V1, V2).

Serialisation compound structure

Table 6.143: scijava.roi.types.Range1nD Details

Property	Value
TypeID	89

6.143 scijava.roi.types.Range2nD

A range of values specified as an inequality (or equality).

Specified as all values for which the formula "n O1 V1" is true, e.g. "n ≤ 5".

Serialisation compound structure

Table 6.144: scijava.roi.types.Range2nD Details

Property	Value
TypeID	90

6.144 scijava.roi.types.Rectangle1

A rectangle in 2D described by two corner points and a vector.

Serialisation compound structure

Table 6.145: scijava.roi.types.Rectangle1 Details

Property	Value
TypeID	175

6.145 scijava.roi.types.Rectangle2

A rectangle in 2D described by a corner point and two vectors.

Serialisation compound structure

Table 6.146: scijava.roi.types.Rectangle2 Details

Property	Value
TypeID	176

6.146 scijava.roi.types.RotateTransform2D

A rotation transformation in 2D.

Serialisation compound structure

Table 6.147: scijava.roi.types.RotateTransform2D Details

Property	Value
TypeID	716

6.147 scijava.roi.types.RotateTransform3D

A rotation transformation in 3D.

Serialisation compound structure

Table 6.148: scijava.roi.types.RotateTransform3D Details

Property	Value
TypeID	717

6.148 scijava.roi.types.ScaleTransform1D

A scaling transformation in 1D.

Serialisation compound structure

Table 6.149: scijava.roi.types.ScaleTransform1D Details

Property	Value
TypeID	713

6.149 scijava.roi.types.ScaleTransform2D

A scaling transformation in 2D.

Serialisation compound structure

Table 6.150: scijava.roi.types.ScaleTransform2D Details

Property	Value
TypeID	714

6.150 scijava.roi.types.ScaleTransform3D

A scaling transformation in 3D.

Serialisation compound structure

Table 6.151: scijava.roi.types.ScaleTransform3D Details

Property	Value
TypeID	715

6.151 scijava.roi.types.Set

Fixed length unordered set.

Serialisation compound structure

Table 6.152: scijava.roi.types.Set Details

Property	Value
TypeID	52

6.152 scijava.roi.types.ShapeSet

A set of shapes.

Serialisation compound structure

Table 6.153: scijava.roi.types.ShapeSet Details

Property	Value
TypeID	93

6.153 scijava.roi.types.Sphere0

A sphere in 3D described by a centre point and surface point.

Serialisation compound structure

Table 6.154: scijava.roi.types.Sphere0 Details

Property	Value
TypeID	210

6.154 scijava.roi.types.Sphere1

A sphere in 3D described by a centre point and 1D radius.

Serialisation compound structure

Table 6.155: scijava.roi.types.Sphere1 Details

Property	Value
TypeID	211

6.155 scijava.roi.types.Sphere2

A sphere in 3D described by a centre point and 2D radius.

Serialisation compound structure

Table 6.156: scijava.roi.types.Sphere2 Details

Property	Value
TypeID	212

6.156 scijava.roi.types.Sphere3

A sphere in 3D described by a centre point and 3D radius.

Serialisation compound structure

Table 6.157: scijava.roi.types.Sphere3 Details

Property	Value
TypeID	213

6.157 scijava.roi.types.Sphere4

A sphere in 3D described by a surface point and vector to the centre point.

Serialisation compound structure

Table 6.158: scijava.roi.types.Sphere4 Details

Property	Value
TypeID	214

6.158 scijava.roi.types.Sphere5

A sphere in 3D described by a two surface points [diameter].

Serialisation compound structure

Table 6.159: scijava.roi.types.Sphere5 Details

Property	Value
TypeID	215

6.159 scijava.roi.types.Sphere6

A sphere in 3D described by a four surface points.

Serialisation compound structure

Table 6.160: scijava.roi.types.Sphere6 Details

Property	Value
TypeID	216

6.160 scijava.roi.types.Square1

An aligned cuboid described by two points in 3D.

Serialisation compound structure

Table 6.161: scijava.roi.types.Square1 Details

Property	Value
TypeID	170

6.161 scijava.roi.types.Square2

An aligned cuboid described by a point and a vector.

Serialisation compound structure

Table 6.162: scijava.roi.types.Square2 Details

Property	Value
TypeID	171

6.162 scijava.roi.types.String

Text string.

Serialisation compound structure

Table 6.163: scijava.roi.types.String Details

Property	Value
TypeID	30

6.163 scijava.roi.types.Text

Text annotation.

Serialisation compound structure

Table 6.164: scijava.roi.types.Text Details

Property	Value
TypeID	None

6.164 scijava.roi.types.TranslateTransform1D

A translation transformation in 1D.

Serialisation compound structure

Table 6.165: scijava.roi.types.TranslateTransform1D Details

Property	Value
TypeID	710

6.165 scijava.roi.types.TranslateTransform2D

A translation transformation in 2D.

Serialisation compound structure

Table 6.166: scijava.roi.types.TranslateTransform2D Details

Property	Value
TypeID	711

6.166 scijava.roi.types.TranslateTransform3D

A translation transformation in 3D.

Serialisation compound structure

Table 6.167: scijava.roi.types.TranslateTransform3D Details

Property	Value
TypeID	712

6.167 scijava.roi.types.TypeID

Numeric shape identifier.

Table 6.168: scijava.roi.types.TypeID Details

Property	Value
TypeID	1

6.168 scijava.roi.types.UInt16

Unsigned 16-bit integer.

Table 6.169: scijava.roi.types.UInt16 Details

Property	Value
TypeID	11

6.169 scijava.roi.types.UInt32

Unsigned 32-bit integer.

Table 6.170: scijava.roi.types.UInt32 Details

Property	Value
TypeID	12

6.170 scijava.roi.types.UInt64

Unsigned 64-bit integer.

Table 6.171: scijava.roi.types.UInt64 Details

Property	Value
TypeID	13

6.171 scijava.roi.types.UInt8

Unsigned 8-bit integer.

Table 6.172: scijava.roi.types.UInt8 Details

Property	Value
TypeID	10

6.172 scijava.roi.types.Value

Numerical value.

Table 6.173: scijava.roi.types.Value Details

Property	Value
TypeID	None

6.173 scijava.roi.types.ValuenD

A single value.

Table 6.174: scijava.roi.types.ValuenD Details

Property	Value
TypeID	None

6.174 scijava.roi.types.ValuesnD

A set of values.

Serialisation compound structure

Table 6.175: scijava.roi.types.ValuesnD Details

Property	Value
TypeID	88

6.175 scijava.roi.types.Vector1D

Vector in 1D.

Serialisation compound structure

Table 6.176: scijava.roi.types.Vector1D Details

Property	Value
TypeID	110

6.176 scijava.roi.types.Vector2D

Vector in 2D.

Serialisation compound structure

Table 6.177: scijava.roi.types.Vector2D Details

Property	Value
TypeID	111

6.177 scijava.roi.types.Vector3D

Vector in 3D.

Serialisation compound structure

Table 6.178: scijava.roi.types.Vector3D Details

Property	Value
TypeID	112

6.178 scijava.roi.types.VectornD

Vector in nD.

Table 6.179: scijava.roi.types.VectornD Details

Property	Value
TypeID	113

6.179 scijava.roi.types.Vectors1D

A list of vectors in 1D.

Serialisation compound structure

Table 6.180: scijava.roi.types.Vectors1D Details

Property	Value
TypeID	None

6.180 scijava.roi.types.Vectors2D

A list of vectors in 2D.

Serialisation compound structure

Table 6.181: scijava.roi.types.Vectors2D Details

Property	Value
TypeID	None

6.181 scijava.roi.types.Vectors3D

A list of vectors in 3D.

Serialisation compound structure

Table 6.182: scijava.roi.types.Vectors3D Details

Property	Value
TypeID	None

6.182 scijava.roi.types.VectorsnD

A list of vectors in nD.

Table 6.183: scijava.roi.types.VectorsnD Details

Property	Value
TypeID	None

6.183 scijava.roi.types.Vertex1D

Vertex in 1D.

Table 6.184: scijava.roi.types.Vertex1D Details

Property	Value
TypeID	100

6.184 scijava.roi.types.Vertex2D

Vertex in 2D.

Table 6.185: scijava.roi.types.Vertex2D Details

Property	Value
TypeID	101

6.185 scijava.roi.types.Vertex3D

Vertex in 3D.

Table 6.186: scijava.roi.types.Vertex3D Details

Property	Value
TypeID	102

6.186 scijava.roi.types.VertexList1D

A list of vertices in 1D.

Table 6.187: scijava.roi.types.VertexList1D Details

Property	Value
TypeID	None

6.187 scijava.roi.types.VertexList2D

A list of vertices in 2D.

Table 6.188: scijava.roi.types.VertexList2D Details

Property	Value
TypeID	None

6.188 scijava.roi.types.VertexList3D

A list of vertices in 3D.

Table 6.189: scijava.roi.types.VertexList3D Details

Property	Value
TypeID	None

6.189 scijava.roi.types.VertexListnD

A list of vertices in nD.

Table 6.190: scijava.roi.types.VertexListnD Details

Property	Value
TypeID	None

6.190 scijava.roi.types.VertexnD

Vertex in nD.

Table 6.191: scijava.roi.types.VertexnD Details

Property	Value
TypeID	103

CHAPTER

SEVEN

FUNDAMENTAL DATA TYPES

The following defined types are used in the subsequent sections. Implementors should treat these sizes as minimium requirements.

Note: Roger Leigh Depending upon how we wish to persue interoperability between implementations, these may be required to be exact. Using plain text would mitigate this to an extent.

Table 7.1: Raw Primitives

Name	Bin-Type	Description
scijava.roi.types.AffineTransform1D	*double[4]*	An affine transform in 1D described by a transformation matrix and 1D shape to transform
scijava.roi.types.AffineTransform2D	*double[9]*	An affine transform in 2D described by a transformation matrix and 2D shape to transform
scijava.roi.types.AffineTransform3D	*double[16]*	An affine transform in 3D described by a transformation matrix and 3D shape to transform
scijava.roi.types.AffineTransformnD	TODO	An affine transform in nD described by a transformation matrix and nD shape to transform
scijava.roi.types.Color	*double[4]*	Color in RGBA (0,1) range
scijava.roi.types.Count	uint32_t	Number of objects
scijava.roi.types.Index	uint32_t	Index into an array
scijava.roi.types.Int16	int16_t	Signed 16-bit integer
scijava.roi.types.Int32	int32_t	Signed 32-bit integer
scijava.roi.types.Int64	int64_t	Signed 64-bit integer
scijava.roi.types.Int8	int8_t	Signed 8-bit integer
scijava.roi.types.TypeID	uint16_t	Numeric shape identifier
scijava.roi.types.UInt16	uint16_t	Unsigned 16-bit integer
scijava.roi.types.UInt32	uint32_t	Unsigned 32-bit integer
scijava.roi.types.UInt64	uint64_t	Unsigned 64-bit integer
scijava.roi.types.UInt8	uint8_t	Unsigned 8-bit integer
scijava.roi.types.Value	double	Numerical value
scijava.roi.types.Vector1D	*double[1]*	Vector in 1D
scijava.roi.types.Vector2D	*double[2]*	Vector in 2D
scijava.roi.types.Vector3D	*double[3]*	Vector in 3D
scijava.roi.types.VectornD	TODO	Vector in nD
scijava.roi.types.Vertex1D	double[1]	Vertex in 1D
scijava.roi.types.Vertex2D	double[2]	Vertex in 2D
scijava.roi.types.Vertex3D	double[3]	Vertex in 3D
scijava.roi.types.VertexnD	TODO	Vertex in nD

Table 7.2: C++ Primitives

Name	C++ Type
scijava.roi.types.AffineTransform1D	*glm::detail::tmat2x2<double>*
scijava.roi.types.AffineTransform2D	*glm::detail::tmat3x3<double>*
scijava.roi.types.AffineTransform3D	*glm::detail::tmat4x4<double>*
scijava.roi.types.AffineTransformnD	TODO
scijava.roi.types.Color	*double[4]*
scijava.roi.types.Count	uint32_t
scijava.roi.types.Index	uint32_t
scijava.roi.types.Int16	int16_t
scijava.roi.types.Int32	int32_t
scijava.roi.types.Int64	int64_t
scijava.roi.types.Int8	int8_t
scijava.roi.types.String	*std::string*
scijava.roi.types.TypeID	uint16_t
scijava.roi.types.UInt16	uint16_t
scijava.roi.types.UInt32	uint32_t
scijava.roi.types.UInt64	uint64_t
scijava.roi.types.UInt8	uint8_t
scijava.roi.types.Value	double
scijava.roi.types.Vector1D	*double*
scijava.roi.types.Vector2D	*glm::detail::tvec2<double>*
scijava.roi.types.Vector3D	*glm::detail::tvec3<double>*
scijava.roi.types.VectornD	TODO
scijava.roi.types.Vertex1D	double
scijava.roi.types.Vertex2D	glm::detail::tvec2<double>
scijava.roi.types.Vertex3D	glm::detail::tvec2<double>
scijava.roi.types.VertexnD	TODO

Table 7.3: Java Primitives

Name	Java Type
scijava.roi.types.Color	*double[4]*
scijava.roi.types.Count	int
scijava.roi.types.Float32	float
scijava.roi.types.Float64	double
scijava.roi.types.Index	int
scijava.roi.types.Int16	short
scijava.roi.types.Int32	int
scijava.roi.types.Int64	long
scijava.roi.types.Int8	byte
scijava.roi.types.String	*String*
scijava.roi.types.TypeID	short
scijava.roi.types.UInt16	int
scijava.roi.types.UInt32	long
scijava.roi.types.UInt64	java.math.BigInteger
scijava.roi.types.UInt8	short
scijava.roi.types.Value	double

Table 7.4: Shape state/attributes

Property	Type	Description
DIMORDER	scijava.roi.types.Array<Index>	Dimension order
TRANSFORM	Affine3D	Affine transformation
BOUNDS	RAlignedCuboid1	Bounding cuboid
LINECOL	Colour	Line (and surface) colour
FILLCOL	Colour	Fill colour
TEXTCOL	Colour	Text colour
DRAWWIDTH	double	Width for drawing
DRAWPLACEMENT	double	Line width is centred (0), fully inside (-1) or fully outside (1) or in between
DRAWSTYLE	enum	
FILLSTYLE	enum	Style to use for filling shapes (could be impemented internally in the form of a Grid Shape+transform)
POINTSTYLE	enum	Style to use for drawing points (could be implemented internally in the form of a Shape)
LINESTYLE	enum	Line style (alternating fill/clear pattern) (could be impemented internally in the form of RVectors1D)
LINESTARTMARKER	enum	Line end marker (arrowhead, etc.) (could be implemented internally in the form of a Shape)
LINEENDMARKER	enum	Line end marker (arrowhead, etc.) (could be implemented internally in the form of a Shape)
MARKERSIZE	double	Size of points and line start/end markers; scales marker
TEXTFONT	scijava.roi.types.String	Font description (format? freetype-style font-desc?)
TEXTPLACEMENT	double[2]	Text placement in bounding box (-1,+1) for x and y limits, (0,0) being centred
TEXTSIZE	double	Font size

Note: **Barry DeZonia** Support different coordinate spaces as needed (int, long, double). Should be possible to iterate some regions.

CHAPTER

EIGHT

INTERFACE TYPES

8.1 scijava.roi.Iterable

Table 8.1: scijava.roi.Iterable

Implemented by
scijava.roi.RegionOfInterest
scijava.roi.types.ROI

8.2 scijava.roi.RegionOfInterest

Table 8.2: scijava.roi.RegionOfInterest

Implemented by
scijava.roi.types.ROI

8.3 scijava.roi.RegionOfInterestSet

Table 8.3: scijava.roi.RegionOfInterestSet

Implemented by
scijava.roi.types.Labelling
scijava.roi.types.ROISet

8.4 scijava.roi.Serialisable

Table 8.4: scijava.roi.Serialisable

Implemented by
scijava.roi.RegionOfInterest
Continued on next page

Table 8.4 – continued from previous page

Implemented by
scijava.roi.RegionOfInterestSet
scijava.roi.dimconstraint.Extrude
scijava.roi.dimconstraint.Range
scijava.roi.dimconstraint.Set
scijava.roi.dimconstraint.Value
scijava.roi.dimconstraint.Values
scijava.roi.shape.AbstractTransform
scijava.roi.shape.AffineTransform
scijava.roi.shape.Arc
scijava.roi.shape.BitMask
scijava.roi.shape.Bitwise
scijava.roi.shape.Cuboid
scijava.roi.shape.Custom
scijava.roi.shape.Cylinder
scijava.roi.shape.Ellipsoid
scijava.roi.shape.GreyMask
scijava.roi.shape.Line
scijava.roi.shape.Lines
scijava.roi.shape.Mesh
scijava.roi.shape.PhysicalShape
scijava.roi.shape.Point
scijava.roi.shape.Points
scijava.roi.shape.Polygon
scijava.roi.shape.PolygonSpline
scijava.roi.shape.Polyline
scijava.roi.shape.PolylineSpline
scijava.roi.shape.Set
scijava.roi.types.DimConstraint
scijava.roi.types.Labelling
scijava.roi.types.ROI
scijava.roi.types.ROISet

8.5 scijava.roi.measurement.Area

Table 8.5: scijava.roi.measurement.Area

Implemented by
scijava.roi.shape.Mesh
scijava.roi.shape.PolygonSpline
scijava.roi.shape.Polyline

8.6 scijava.roi.measurement.Length

Table 8.6: scijava.roi.measurement.Length

Implemented by
scijava.roi.shape.Arc
scijava.roi.shape.Line
scijava.roi.shape.Lines
scijava.roi.shape.Polyline
scijava.roi.shape.PolylineSpline

8.7 scijava.roi.measurement.Volume

Table 8.7: scijava.roi.measurement.Volume

Implemented by
scijava.roi.shape.Cuboid
scijava.roi.shape.Cylinder
scijava.roi.shape.Ellipsoid
scijava.roi.shape.Mesh

8.8 scijava.roi.shape.PhysicalShape

Table 8.8: scijava.roi.shape.PhysicalShape

Implemented by
scijava.roi.shape.AbstractTransform
scijava.roi.shape.AffineTransform
scijava.roi.shape.Arc
scijava.roi.shape.BitMask
scijava.roi.shape.Bitwise
scijava.roi.shape.Cuboid
scijava.roi.shape.Custom
scijava.roi.shape.Cylinder
scijava.roi.shape.Ellipsoid
scijava.roi.shape.GreyMask
scijava.roi.shape.Line
scijava.roi.shape.Lines
scijava.roi.shape.Mesh
scijava.roi.shape.Point
scijava.roi.shape.Points
scijava.roi.shape.Polygon
scijava.roi.shape.PolygonSpline
scijava.roi.shape.Polyline
scijava.roi.shape.PolylineSpline
scijava.roi.shape.Set

8.9 scijava.roi.types.DimConstraint

Table 8.9: scijava.roi.types.DimConstraint

Implemented by
scijava.roi.dimconstraint.Extrude
scijava.roi.dimconstraint.Range
scijava.roi.dimconstraint.Set
scijava.roi.dimconstraint.Value
scijava.roi.dimconstraint.Values

CHAPTER

NINE

ENUMERATED TYPES

9.1 scijava.roi.types.BLogic

Table 9.1: scijava.roi.types.BLogic

Name	Number	Symbol	Description
AND	0	AND	And
OR	1	OR	Or
NOT	2	NOT	Not
XOR	3	XOR	Exclusive or

9.2 scijava.roi.types.Operator

Table 9.2: scijava.roi.types.Operator

Name	Number	Symbol	Description
EQ	0	=	Equals
NE	1	≠	Not equals
LT	2	<	Less than
LE	3	≤	Less than or equal to
GT	4	>	Greater than
GE	5	≥	Greater than or equal to

CHAPTER

TEN

COMPOUND TYPES

10.1 scijava.roi.types.AbstractTransform1D

Table 10.1: scijava.roi.types.AbstractTransform1D

SeqNo	Type	Name	Description
0	scijava.roi.types.String	NAME	Name of transformation
1	scijava.roi.types.String	ARGS	Arguments
2	Shape	SHAPE	Shape

10.2 scijava.roi.types.AbstractTransform2D

Table 10.2: scijava.roi.types.AbstractTransform2D

SeqNo	Type	Name	Description
0	scijava.roi.types.String	NAME	Name of transformation
1	scijava.roi.types.String	ARGS	Arguments
2	Shape	SHAPE	Shape

10.3 scijava.roi.types.AbstractTransform3D

Table 10.3: scijava.roi.types.AbstractTransform3D

SeqNo	Type	Name	Description
0	scijava.roi.types.String	NAME	Name of transformation
1	scijava.roi.types.Array<scijava.roi.types.String>	ARGS	Arguments
2	Shape	SHAPE	Shape

10.4 scijava.roi.types.AffineTransform1D

Table 10.4: scijava.roi.types.AffineTransform1D

SeqNo	Type	Name	Description
0	Affine1D	TRANS	Transform
1	Shape	SHAPE	Shape

10.5 scijava.roi.types.AffineTransform2D

Table 10.5: scijava.roi.types.AffineTransform2D

SeqNo	Type	Name	Description
0	Affine2D	TRANS	Transform
1	Shape	SHAPE	Shape

10.6 scijava.roi.types.AffineTransform3D

Table 10.6: scijava.roi.types.AffineTransform3D

SeqNo	Type	Name	Description
0	Affine3D	TRANS	Transform
1	Shape	SHAPE	Shape

10.7 scijava.roi.types.AlignedBitMask1D

Table 10.7: scijava.roi.types.AlignedBitMask1D

SeqNo	Type	Name	Description
0	LinePoints1D	B1	Bounding line
1	Vector1D	DIM1	Mask dimensions (x)
2	bool[x]	DATA	Mask data

10.8 scijava.roi.types.AlignedBitMask2D

Table 10.8: scijava.roi.types.AlignedBitMask2D

SeqNo	Type	Name	Description
0	LinePoints2D	B1	Bounding box
1	Vector2D	DIM1	Mask dimensions (x,y)
2	bool[x,y]	DATA	Mask data

10.9 scijava.roi.types.AlignedBitMask3D

Table 10.9: scijava.roi.types.AlignedBitMask3D

SeqNo	Type	Name	Description
0	LinePoints3D	B1	Bounding box
1	Vector3D	DIM1	Mask dimensions (x,y,z)
2	bool[x,y,z]	DATA	Mask data

10.10 scijava.roi.types.AlignedCube1

Table 10.10: scijava.roi.types.AlignedCube1

SeqNo	Type	Name	Description
0	Vertex3D	P1	First corner
1	Vertex1D	P2	x coordinate of adjacent/opposing corner

10.11 scijava.roi.types.AlignedCube2

Table 10.11: scijava.roi.types.AlignedCube2

SeqNo	Type	Name	Description
0	Vertex3D	P1	First corner
1	Vector1D	P2	distance to adjacent corner on x axis (relative to P1)

10.12 scijava.roi.types.AlignedCuboid1

Table 10.12: scijava.roi.types.AlignedCuboid1

SeqNo	Type	Name	Description
0	LinePoints3D	P1	Corner and opposing corner

10.13 scijava.roi.types.AlignedCuboid2

Table 10.13: scijava.roi.types.AlignedCuboid2

SeqNo	Type	Name	Description
0	LineVector3D	P1	Corner and vector to opposing corner

10.14 scijava.roi.types.AlignedGreyMask1D

Table 10.14: scijava.roi.types.AlignedGreyMask1D

SeqNo	Type	Name	Description
0	LinePoints1D	B1	Bounding line
1	Vector1D	DIM1	Mask dimensions (x)
2	double[x]	DATA	Mask data

10.15 scijava.roi.types.AlignedGreyMask2D

Table 10.15: scijava.roi.types.AlignedGreyMask2D

SeqNo	Type	Name	Description
0	LinePoints2D	B1	Bounding box
1	Vector2D	DIM1	Mask dimensions (x,y)
2	double[x,y]	DATA	Mask data

10.16 scijava.roi.types.AlignedGreyMask3D

Table 10.16: scijava.roi.types.AlignedGreyMask3D

SeqNo	Type	Name	Description
0	LinePoints3D	B1	Bounding box
1	Vector3D	DIM1	Mask dimensions (x,y,z)
2	double[x,y,z]	DATA	Mask data

10.17 scijava.roi.types.AlignedHalfAxes2D

Table 10.17: scijava.roi.types.AlignedHalfAxes2D

SeqNo	Type	Name	Description
0	Vertex2D	P1	Centre point
1	Vector2D	V1	Half axes (x,y)

10.18 scijava.roi.types.AlignedHalfAxes3D

Table 10.18: scijava.roi.types.AlignedHalfAxes3D

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vector3D	V1	Half axes (x,y,z)

10.19 scijava.roi.types.AlignedRectangle1

Table 10.19: scijava.roi.types.AlignedRectangle1

SeqNo	Type	Name	Description
0	LinePoints2D	P1	Corner and opposing corner

10.20 scijava.roi.types.AlignedRectangle2

Table 10.20: scijava.roi.types.AlignedRectangle2

SeqNo	Type	Name	Description
0	LineVector2D	P1	Corner and vector to opposing corner

10.21 scijava.roi.types.AlignedSquare1

Table 10.21: scijava.roi.types.AlignedSquare1

SeqNo	Type	Name	Description
0	Vertex2D	P1	First corner
1	Vertex1D	P2	x coordinate of adjacent/opposing corner

10.22 scijava.roi.types.AlignedSquare2

Table 10.22: scijava.roi.types.AlignedSquare2

SeqNo	Type	Name	Description
0	Vertex2D	P1	First corner
1	Vector1D	P2	distance to adjacent corner on x axis (relative to P1)

10.23 scijava.roi.types.Arc12D

Table 10.23: scijava.roi.types.Arc12D

SeqNo	Type	Name	Description
0	LinePoints2D	P1	Centre point and arc start
1	Vector2D	V1	Arc end

10.24 scijava.roi.types.Arc13D

Table 10.24: scijava.roi.types.Arc13D

SeqNo	Type	Name	Description
0	LinePoints3D	P1	Centre point and arc start
1	Vector3D	V1	Arc end

10.25 scijava.roi.types.Arc22D

Table 10.25: scijava.roi.types.Arc22D

SeqNo	Type	Name	Description
0	LineVector2D	P1	Centre point and arc start
1	Vector2D	V1	Arc end

10.26 scijava.roi.types.Arc23D

Table 10.26: scijava.roi.types.Arc23D

SeqNo	Type	Name	Description
0	LineVector3D	P1	Centre point and arc start
1	Vector3D	V1	Arc end

10.27 scijava.roi.types.Arc32D

Table 10.27: scijava.roi.types.Arc32D

SeqNo	Type	Name	Description
0	Vertex2D[3]	P1	Centre point, arc start and arc end (vector inferred)

10.28 scijava.roi.types.Arc33D

Table 10.28: scijava.roi.types.Arc33D

SeqNo	Type	Name	Description
0	Vertex3D[3]	P1	Centre point, arc start and arc end (vector inferred)

10.29 scijava.roi.types.Array<TYPE>

Table 10.29: scijava.roi.types.Array<TYPE>

SeqNo	Type	Name	Description
(T0)	scijava.roi.types.TypeID	TYPE	Type stored in container
0	scijava.roi.types.Count	NELEM	Number of elements
1	TYPE[NELEM]	ELEM	Elements

10.30 scijava.roi.types.BitMask2D

Table 10.30: scijava.roi.types.BitMask2D

SeqNo	Type	Name	Description
0	Rectangle2	B1	Bounding box
1	Vector2D	DIM1	Mask dimensions (x,y)
2	bool[x,y]	DATA	Mask data

10.31 scijava.roi.types.BitMask3D

Table 10.31: scijava.roi.types.BitMask3D

SeqNo	Type	Name	Description
0	Cuboid2	B1	Bounding box
1	Vector3D	DIM1	Mask dimensions (x,y,z)
2	bool[x,y,z]	DATA	Mask data

10.32 scijava.roi.types.Bitwise1D

Table 10.32: scijava.roi.types.Bitwise1D

SeqNo	Type	Name	Description
0	BLogic	O1	Bitwise logic operator
1	BitMask1D	M1	Mask 1
2	BitMask1D	M2	Mask 2

10.33 scijava.roi.types.Bitwise2D

Table 10.33: scijava.roi.types.Bitwise2D

SeqNo	Type	Name	Description
0	BLogic	O1	Bitwise logic operator
1	BitMask2D	M1	Mask 1
2	BitMask2D	M2	Mask 2

10.34 scijava.roi.types.Bitwise3D

Table 10.34: scijava.roi.types.Bitwise3D

SeqNo	Type	Name	Description
0	BLogic	O1	Bitwise logic operator
1	BitMask3D	M1	Mask 1
2	BitMask3D	M2	Mask 2

10.35 scijava.roi.types.Circle0

Table 10.35: scijava.roi.types.Circle0

SeqNo	Type	Name	Description
0	Vertex2D	P1	Centre point
1	Vertex2D	P2	Circumference point

10.36 scijava.roi.types.Circle1

Table 10.36: scijava.roi.types.Circle1

SeqNo	Type	Name	Description
0	Vertex2D	P1	Centre point
1	Vector1D	V1	Radius

10.37 scijava.roi.types.Circle2

Table 10.37: scijava.roi.types.Circle2

SeqNo	Type	Name	Description
0	Vertex2D	P1	Centre point
1	Vector2D	V1	Radius

10.38 scijava.roi.types.Circle3

Table 10.38: scijava.roi.types.Circle3

SeqNo	Type	Name	Description
0	Vertex2D	P1	Point on circumference
1	Vector2D	V1	Vector to centre

10.39 scijava.roi.types.Circle4

Table 10.39: scijava.roi.types.Circle4

SeqNo	Type	Name	Description
0	Vertex2D[2]	P1	Two points on circumference

10.40 scijava.roi.types.Circle5

Table 10.40: scijava.roi.types.Circle5

SeqNo	Type	Name	Description
0	Vertex2D[3]	P1	Three points on circumference

10.41 scijava.roi.types.CircularCylinder1

Table 10.41: scijava.roi.types.CircularCylinder1

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vertex3D	P2	Centre of second face
2	Vector1D	V1	Radius

10.42 scijava.roi.types.CircularCylinder2

Table 10.42: scijava.roi.types.CircularCylinder2

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vector3D	V1	Distance to centre of second face
2	Vector1D	V2	Radius

10.43 scijava.roi.types.CircularCylinder3

Table 10.43: scijava.roi.types.CircularCylinder3

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vertex3D	P2	Centre of second face
2	Vector3D	V1	Radius and angle of first face
3	Vector3D	V2	Angle of second face

10.44 scijava.roi.types.CircularCylinder4

Table 10.44: scijava.roi.types.CircularCylinder4

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vector3D	V1	Distance to centre of second face
2	Vector3D	V2	Radius and angle of first face
3	Vector3D	V3	Angle of second face

10.45 scijava.roi.types.Color

Table 10.45: scijava.roi.types.Color

SeqNo	Type	Name	Description
0	scijava.roi.types.Float64	R	Red value (0,1)
1	scijava.roi.types.Float64	G	Green value (0,1)
2	scijava.roi.types.Float64	B	Blue value (0,1)
3	scijava.roi.types.Float64	A	Alpha value (0,1)

10.46 scijava.roi.types.Cube1

Table 10.46: scijava.roi.types.Cube1

SeqNo	Type	Name	Description
0	LinePoints3D	P1	Corner and adjacent corner

10.47 scijava.roi.types.Cube2

Table 10.47: scijava.roi.types.Cube2

SeqNo	Type	Name	Description
0	LineVector3D	P1	Corner and vector to adjacent corner

10.48 scijava.roi.types.Cuboid1

Table 10.48: scijava.roi.types.Cuboid1

SeqNo	Type	Name	Description
0	Vertex3D	P1	First corner
1	Vertex3D	P2	Second corner (adjacent to P1)
2	Vector2D	V1	Distance to third corner (adjacent to P2)
3	Vector1D	V2	Distance to fourth corner (opposing P1, adjacent to V1)

10.49 scijava.roi.types.Cuboid2

Table 10.49: scijava.roi.types.Cuboid2

SeqNo	Type	Name	Description
0	Vertex3D	P1	First corner
1	Vector3D	V1	Distance to second corner (relative to P1)
2	Vector2D	V2	Distance to third corner (relative to V1)
3	Vector1D	V3	Distance to fourth corner (relative to V2, opposing P1)

10.50 scijava.roi.types.Custom

Table 10.50: scijava.roi.types.Custom

SeqNo	Type	Name	Description
0	scijava.roi.types.String	TYPE	Name of the custom type
1	Set	MEASUREMENTS	Set of shapes describing how the ROI was measured
2	Set	RESULTS	Set of shapes for describing measurement results
3	Set	EDIT	Set of shapes describing how to edit the ROI
4	Set	VISUAL	Set of shapes describing how to visualise (render) the ROI for visualisation

10.51 scijava.roi.types.DimConstraintSet

Table 10.51: scijava.roi.types.DimConstraintSet

SeqNo	Type	Name	Description
0	scijava.roi.types.Set<scijava.roi.types.DimConstraintSet>	CONSTRAINTS	Set of dimensional constraints

10.52 scijava.roi.types.DirectedGraph<NTYPE, ETYPE>

Table 10.52: scijava.roi.types.DirectedGraph<NTYPE, ETYPE>

SeqNo	Type	Name	Description
(T0)	scijava.roi.types.TypeID	NTYPE	Node type stored in container
(T1)	scijava.roi.types.TypeID	ETYPE	Edge type stored in container
0	scijava.roi.types.Array<NTYPE>	VERTS	Nodes
1	scijava.roi.types.Array<scijava.roi.types.Pair<ETYPE,scijava.roi.types.Pair<Index,Index>>>	EDGES	Edges, including out and in vertex numbers

10.53 scijava.roi.types.EllipticCylinder1

Table 10.53: scijava.roi.types.EllipticCylinder1

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vertex3D	P2	Centre of second face
2	Vector2D	V1	Half axes (xy)
3	Vector1D	V2	Half axes (x)

10.54 scijava.roi.types.EllipticCylinder2

Table 10.54: scijava.roi.types.EllipticCylinder2

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vector3D	V1	Distance to second face
2	Vector3D	V2	Half axes (xy)
3	Vector2D	V3	Half axes (x)

10.55 scijava.roi.types.EllipticCylinder3

Table 10.55: scijava.roi.types.EllipticCylinder3

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vertex3D	P2	Centre of second face
2	Vector3D	V1	Half axes of first face (xyz)
3	Vector2D	V2	Half axes of first face (xy)
4	Vector3D	V3	Angle of second face

10.56 scijava.roi.types.EllipticCylinder4

Table 10.56: scijava.roi.types.EllipticCylinder4

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre of first face
1	Vector3D	V1	Distance to second face
2	Vector3D	V2	Half axes (xyz)
3	Vector2D	V3	Half axes (xy)
4	Vector3D	V4	Angle of second face

10.57 scijava.roi.types.Extrude

Table 10.57: scijava.roi.types.Extrude

SeqNo	Type	Name	Description
0	Index	D1	Dimension
1	Shape	SHAPE	Shape

10.58 scijava.roi.types.GreyMask2D

Table 10.58: scijava.roi.types.GreyMask2D

SeqNo	Type	Name	Description
0	Rectangle2	B1	Bounding box
1	Vector2D	DIM1	Mask dimensions (x,y)
2	double[x,y]	DATA	Mask data

10.59 scijava.roi.types.GreyMask3D

Table 10.59: scijava.roi.types.GreyMask3D

SeqNo	Type	Name	Description
0	Cuboid2	B1	Bounding box
1	Vector3D	DIM1	Mask dimensions (x,y,z)
2	double[x,y,z]	DATA	Mask data

10.60 scijava.roi.types.HalfAxes2D

Table 10.60: scijava.roi.types.HalfAxes2D

SeqNo	Type	Name	Description
0	Vertex2D	P1	Centre point
1	Vector2D	V1	Half axes (xy)
2	Vector1D	V2	Half axes (x)

10.61 scijava.roi.types.HalfAxes3D

Table 10.61: scijava.roi.types.HalfAxes3D

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vector3D	V1	Half axes (xyz)
2	Vector2D	V2	Half axes (xy)
3	Vector1D	V3	Half axes (x)

10.62 scijava.roi.types.LinePoints1D

Table 10.62: scijava.roi.types.LinePoints1D

SeqNo	Type	Name	Description
0	Vertex1D[2]	P1	Line start and end points

10.63 scijava.roi.types.LinePoints2D

Table 10.63: scijava.roi.types.LinePoints2D

SeqNo	Type	Name	Description
0	Vertex2D[2]	P1	Line start and end points

10.64 scijava.roi.types.LinePoints3D

Table 10.64: scijava.roi.types.LinePoints3D

SeqNo	Type	Name	Description
0	Vertex3D[2]	P1	Line start and end points

10.65 scijava.roi.types.LineVector1D

Table 10.65: scijava.roi.types.LineVector1D

SeqNo	Type	Name	Description
0	Vertex1D	P1	Line start

10.66 scijava.roi.types.LineVector2D

Table 10.66: scijava.roi.types.LineVector2D

SeqNo	Type	Name	Description
0	Vertex2D	P1	Line start
1	Vector2D	V1	Line end (relative to P1)

10.67 scijava.roi.types.LineVector3D

Table 10.67: scijava.roi.types.LineVector3D

SeqNo	Type	Name	Description
0	Vertex3D	P1	Line start
1	Vector3D	V1	Line end (relative to P1)

10.68 scijava.roi.types.LinesPoints1D

Table 10.68: scijava.roi.types.LinesPoints1D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLinePoints1D>	LINES	Array of line points

10.69 scijava.roi.types.LinesPoints2D

Table 10.69: scijava.roi.types.LinesPoints2D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLinePoints2D>	LINES	Array of line points

10.70 scijava.roi.types.LinesPoints3D

Table 10.70: scijava.roi.types.LinesPoints3D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLinePoints3D>	LINES	Array of line points

10.71 scijava.roi.types.LinesVectors1D

Table 10.71: scijava.roi.types.LinesVectors1D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLineVector1D>	LINES	Array of line vectors

10.72 scijava.roi.types.LinesVectors2D

Table 10.72: scijava.roi.types.LinesVectors2D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLineVector2D>	LINES	Array of line vectors

10.73 scijava.roi.types.LinesVectors3D

Table 10.73: scijava.roi.types.LinesVectors3D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<RLineVector3D>	LINES	Array of line vectors

10.74 scijava.roi.types.Map<KTYPE, VTYPE>

Table 10.74: scijava.roi.types.Map<KTYPE, VTYPE>

SeqNo	Type	Name	Description
(T0)	scijava.roi.types.TypeID	KTYPE	Key type stored in container
(T1)	scijava.roi.types.TypeID	VTYPE	Value type stored in container
0	scijava.roi.types.Array<scijava.roi.types.Pair<KTYPE,VTYPE>>	ELEM	Array of key-value pairs

10.75 scijava.roi.types.Mesh2D

Table 10.75: scijava.roi.types.Mesh2D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<double[3]>	FACES	Vertex references per face, counterclockwise winding
1	scijava.roi.types.Array<Vertex2D>	VERTS	Vertex coordinates

10.76 scijava.roi.types.Mesh3D

Table 10.76: scijava.roi.types.Mesh3D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<double[3]>	FACES	Vertex references per face, counterclockwise winding
1	scijava.roi.types.Array<Vertex3D>	VERTS	Vertex coordinates

10.77 scijava.roi.types.Pair<LTYPE, RTYPE>

Table 10.77: scijava.roi.types.Pair<LTYPE, RTYPE>

SeqNo	Type	Name	Description
(T0)	scijava.roi.types.TypeID	LTYPE	Left hand type
(T1)	scijava.roi.types.TypeID	RTYPE	Right hand type
0	LTYPE	LEFT	Left hand value
1	LTYPE	RIGHT	Right hand value

10.78 scijava.roi.types.Points1D

Table 10.78: scijava.roi.types.Points1D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vertex1D>	POINTS	Array of point coordinates

10.79 scijava.roi.types.Points2D

Table 10.79: scijava.roi.types.Points2D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vertex2D>	POINTS	Array of point coordinates

10.80 scijava.roi.types.Points3D

Table 10.80: scijava.roi.types.Points3D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vertex3D>	POINTS	Array of point coordinates

10.81 scijava.roi.types.PolylinePoints1D

Table 10.81: scijava.roi.types.PolylinePoints1D

SeqNo	Type	Name	Description
0	Points1D	P1	Array of points

10.82 scijava.roi.types.PolylinePoints2D

Table 10.82: scijava.roi.types.PolylinePoints2D

SeqNo	Type	Name	Description
0	Points2D	P1	Array of points

10.83 scijava.roi.types.PolylinePoints3D

Table 10.83: scijava.roi.types.PolylinePoints3D

SeqNo	Type	Name	Description
0	Points3D	P1	Array of points

10.84 scijava.roi.types.PolylineVector1D

Table 10.84: scijava.roi.types.PolylineVector1D

SeqNo	Type	Name	Description
0	Vertex1D	P1	First point
1	scijava.roi.types.Array<Vector1D>	V1	Array of vectors

10.85 scijava.roi.types.PolylineVector2D

Table 10.85: scijava.roi.types.PolylineVector2D

SeqNo	Type	Name	Description
0	Vertex2D	P1	First point
1	scijava.roi.types.Array<Vector2D>	V1	Array of vectors

10.86 scijava.roi.types.PolylineVector3D

Table 10.86: scijava.roi.types.PolylineVector3D

SeqNo	Type	Name	Description
0	Vertex3D	P1	First point
1	scijava.roi.types.Array<Vector3D>	V1	Array of vectors

10.87 scijava.roi.types.Properties

Table 10.87: scijava.roi.types.Properties

SeqNo	Type	Name	Description
0	scijava.roi.types.Set<scijava.roi.types.Property>	PROPS	Set of properties

10.88 scijava.roi.types.Property

Table 10.88: scijava.roi.types.Property

SeqNo	Type	Name	Description
0	scijava.roi.types.String	KEY	Property name
1	Representation	VALUE	Property value (includes type information)

10.89 scijava.roi.types.Range1nD

Table 10.89: scijava.roi.types.Range1nD

SeqNo	Type	Name	Description
0	Index	D1	Dimension
1	Index	V1	Starting value within dimension
2	Index	V2	Ending value +1 within dimension

10.90 scijava.roi.types.Range2nD

Table 10.90: scijava.roi.types.Range2nD

SeqNo	Type	Name	Description
0	Index	D1	Dimension
1	Operator	O1	Mathematical operator
2	Value	V1	Value for operation

10.91 scijava.roi.types.Rectangle1

Table 10.91: scijava.roi.types.Rectangle1

SeqNo	Type	Name	Description
0	Vertex2D	P1	First corner
1	Vertex2D	P2	Adjacent corner
2	Vector1D	V1	Distance to corner opposing P1 (relative to P2)

10.92 scijava.roi.types.Rectangle2

Table 10.92: scijava.roi.types.Rectangle2

SeqNo	Type	Name	Description
0	Vertex2D	P1	First corner
1	Vector2D	V1	Distance to adjacent corner (relative to P1)
2	Vector1D	V2	Distance to corner opposing P1 (relative to P2)

10.93 scijava.roi.types.RotateTransform2D

Table 10.93: scijava.roi.types.RotateTransform2D

SeqNo	Type	Name	Description
0	double[1]	RA	Rotation angle in z
1	Shape	SHAPE	Shape

10.94 scijava.roi.types.RotateTransform3D

Table 10.94: scijava.roi.types.RotateTransform3D

SeqNo	Type	Name	Description
0	double[3]	RA	Rotation angle in x,y,z
1	Shape	SHAPE	Shape

10.95 scijava.roi.types.ScaleTransform1D

Table 10.95: scijava.roi.types.ScaleTransform1D

SeqNo	Type	Name	Description
0	double[1]	SF1	Scale factor for x
1	Shape1D	SHAPE	Shape

10.96 scijava.roi.types.ScaleTransform2D

Table 10.96: scijava.roi.types.ScaleTransform2D

SeqNo	Type	Name	Description
0	double[2]	SF1	Scale factor for x,y
1	Shape1D	SHAPE	Shape

10.97 scijava.roi.types.ScaleTransform3D

Table 10.97: scijava.roi.types.ScaleTransform3D

SeqNo	Type	Name	Description
0	double[3]	SF1	Scale factor for x,y,z
1	Shape	SHAPE	Shape

10.98 scijava.roi.types.Set<TYPE>

Table 10.98: scijava.roi.types.Set<TYPE>

SeqNo	Type	Name	Description
(T0)	scijava.roi.types.TypeID	TYPE	Type stored in container
0	scijava.roi.types.Count	NELEM	Number of elements
1	TYPE[NELEM]	ELEM	Elements

10.99 scijava.roi.types.ShapeSet

Table 10.99: scijava.roi.types.ShapeSet

SeqNo	Type	Name	Description
0	scijava.roi.types.Set<Shape>	SHAPES	Set of shapes

10.100 scijava.roi.types.Sphere0

Table 10.100: scijava.roi.types.Sphere0

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vertex3D	P2	Surface point

10.101 scijava.roi.types.Sphere1

Table 10.101: scijava.roi.types.Sphere1

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vector1D	V1	Radius

10.102 scijava.roi.types.Sphere2

Table 10.102: scijava.roi.types.Sphere2

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vector2D	V1	Radius

10.103 scijava.roi.types.Sphere3

Table 10.103: scijava.roi.types.Sphere3

SeqNo	Type	Name	Description
0	Vertex3D	P1	Centre point
1	Vector3D	V1	Radius

10.104 scijava.roi.types.Sphere4

Table 10.104: scijava.roi.types.Sphere4

SeqNo	Type	Name	Description
0	Vertex3D	P1	Point on surface
1	Vector3D	V1	Vector to centre

10.105 scijava.roi.types.Sphere5

Table 10.105: scijava.roi.types.Sphere5

SeqNo	Type	Name	Description
0	Vertex3D[2]	P1	Two points on surface

10.106 scijava.roi.types.Sphere6

Table 10.106: scijava.roi.types.Sphere6

SeqNo	Type	Name	Description
0	Vertex3D[4]	P1	Four points on surface

10.107 scijava.roi.types.Square1

Table 10.107: scijava.roi.types.Square1

SeqNo	Type	Name	Description
0	LinePoints2D	P1	Corner and opposing corner

10.108 scijava.roi.types.Square2

Table 10.108: scijava.roi.types.Square2

SeqNo	Type	Name	Description
0	LineVector2D	P1	Corner and vector to opposing corner

10.109 scijava.roi.types.String

Table 10.109: scijava.roi.types.String

SeqNo	Type	Name	Description
0	Count	NCHAR	Number of octets
1	CHARS	uint8[NCHAR]	Array of octets (UTF-8)

10.110 scijava.roi.types.Text

Table 10.110: scijava.roi.types.Text

SeqNo	Type	Name	Description
0	Rectangle2	B1	Text bounds
1	scijava.roi.types.String	TEXT	Text

10.111 scijava.roi.types.TranslateTransform1D

Table 10.111: scijava.roi.types.TranslateTransform1D

SeqNo	Type	Name	Description
0	Vector1D	TR1	Translation in x
1	Shape1D	SHAPE	Shape

10.112 scijava.roi.types.TranslateTransform2D

Table 10.112: scijava.roi.types.TranslateTransform2D

SeqNo	Type	Name	Description
0	Vector2D	TR1	Translation in x,y
1	Shape1D	SHAPE	Shape

10.113 scijava.roi.types.TranslateTransform3D

Table 10.113: scijava.roi.types.TranslateTransform3D

SeqNo	Type	Name	Description
0	Vector3D	TR1	Translation in x,y,z
1	Shape1D	SHAPE	Shape

10.114 scijava.roi.types.ValuesnD

Table 10.114: scijava.roi.types.ValuesnD

SeqNo	Type	Name	Description
0	Index	D1	Dimension
1	scijava.roi.types.Array<Index>	V1	Values within dimension

10.115 scijava.roi.types.Vector1D

Table 10.115: scijava.roi.types.Vector1D

SeqNo	Type	Name	Description
0	Vector1D	V1	Vector

10.116 scijava.roi.types.Vector2D

Table 10.116: scijava.roi.types.Vector2D

SeqNo	Type	Name	Description
0	Vector2D	V1	Vector

10.117 scijava.roi.types.Vector3D

Table 10.117: scijava.roi.types.Vector3D

SeqNo	Type	Name	Description
0	Vector3D	V1	Vector

10.118 scijava.roi.types.Vectors1D

Table 10.118: scijava.roi.types.Vectors1D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vector1D>	VECS	Array of vectors

10.119 scijava.roi.types.Vectors2D

Table 10.119: scijava.roi.types.Vectors2D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vector2D>	VECS	Array of vectors

10.120 scijava.roi.types.Vectors3D

Table 10.120: scijava.roi.types.Vectors3D

SeqNo	Type	Name	Description
0	scijava.roi.types.Array<Vector3D>	VECS	Array of vectors

CHAPTER

ELEVEN

GEOMETRIC SHAPE PRIMITIVES

11.1 Overview

This section specifies how shapes are described in the model. For some shapes, there are several alternative ways of specifying them; which are worth supporting needs further discussion. One point to consider is that the different ways preserve the intent behind the original measurement and what is in the original metadata where this makes sense, even if this does mean some redundancy; this won't impact on the actual drawing/analysis code, which can deal with each shape in a canonical form. This records how the measurement was made by the user, which may have implications in further analysis and/or verification that the measurement was correct.

While some shapes have been included here for completeness, it's quite possible that not all are needed, particularly in all dimensions.

If anyone wants to check the maths behind the geometry, that would be much appreciated, because I'm firstly not an expert in this area, and it's also quite possible I've made some typos. The naming of the shapes is probably also wanting some improvement.

11.2 Alternative shape representations

Using the current ROI model is that there is only one way to describe each shape. e.g. a polyline can only be described as a series of points; it might in some cases be more natural to specify one as a starting point and a series of vectors; while either are fine just to draw the ROI, it would desirable to store what was measured, since converting it to a canonical representation is lossy, and removes the original measurements taken, and hence the intent of the original annotation. This applies to other shapes as well. For example, a circle or ellipse can be described by a bounding box (which may itself be a point and one or two vectors, or a set of points), or by a point and radius or half-axes, or by the Mahalanobis distance (typically for computing from a normal distribution of points). For a cylinder/cone, we can specify this in multiple ways also from a circle/ellipse plus length, or point plus vector (length and direction) plus radius (or half-axes).

The current model is focussed on drawing shapes, while making measurements involves drawing only for visualisation; the important parts are the values for making the measurement, and of course the results. Some programs (e.g. AxioVision) have separate sets of objects for drawing (annotation) and measurement. These are a largely overlapping set, but the former are not used for any length/area/volume/pixel measurements. Objects such as scale bars and labels are for drawing only.

Todo

Common methods for all primitives: Bounding box [AlignedCuboid3D] Rotation centre [Vertex2D/Vertex3D] Control points [may use points and vertex to describe position and movement path] Conversion to 2D (slab through); equivalent to intersection with cuboid. Should all primitives support a minimum of intersection with AlignedCuboid3D? Or Mesh3D for non-square images. Can 2D methods use alternative axes to project in xz/yz? Default to xy. If all 2D shapes must be represented by 3D forms (i.e. are just proxies), then the equivalent 3D can be used quite simply. Get greymap/bitmap. Get 2D/3D mesh. Intersect (only for cuboid?) Need to clip to image volume (optionally). Also useful to reduce to 2D (which can be a cuboid for a single plane). Non-aligned shapes inherit/implement the aligned forms. Shrink and grow: move polygons along surface normals for meshes. For other shapes, this will require recalculation of the geometry.

Add triangle as special case of polygon, which can be a special case of mesh?

Meshes: Need to be able to triangulate if higher order polygons are possible.

Add representation number to start of number list; this will allow shapes to be embedded in other shapes and be self-describing. e.g. all circle types may be used to specify a circular cylinder end. This will simplify the specification of more complex shapes by limiting the number of variants.

11.3 Shape serialisation

All shape primitives are described in terms of the above fundamental primitives. This means that all shape descriptions are serialisable as a list of integer and double-precision floating point values. The specifics of this are implementation-defined. Example formats:

- Plain text, as a list of values
- XML, as element content or a string attribute
- Binary data stream, using big-endian/network byte order

This also means that for compatible shape types, the shape type may be changed while retaining the following data unchanged (e.g. polyline to polygon spline with the same point list).

Note: **Roger Leigh** All 2D shape primitives could be oriented in 3D or using a unit Vector3D, which would allow all 2D shapes to be used as surfaces in 3D. They would additionally require a depth in order to be meaningful (or assume a depth of one z slice).

Or, 2D shapes should specify the pair of x/y/z axes they are using, and will be extruded along the third axis.

Note: ** Sébastien ** Versioning is of concern to people doing analysis.

Key considerations:

- A shape exists in a set of dimensions e.g. xy, xyz, xyt. The shape must define the number of dimensions it exists in, and their identity.
- A shape must be identifiable unambiguously
- A shape must be versioned (to permit correction of any design/analysis bugs without altering any data retrospectively); this permits the replacement of the buggy implementation while not removing it.

- In order to allow code reuse and flexible use of shapes, shapes may include other shapes as part of their primitive specification.

In the following shape descriptions, all shapes are identified by a Shape ID and Representation ID. The shape specifies the geometric shape type. The representation specifies both the primitives required for serialisation, and can also be used for versioning the shape–i.e. it also specifies the behaviour for conversion to greymaps and bitmaps. The behaviour could change in a backward-compatible manner by introducing new Shapes and/or Representations to supersede existing forms, while retaining the unchanged old forms.

11.4 Shape

An abstract description of a shape.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation

Concrete implementations of shapes provide further elements in their representation. The above are only sufficient to describe the shape and its representation. The combination of shape and representation specifies the data required to construct the shape.

Note that one disadvantage of this method is that a reader will be required to understand how to deserialise all shape types; it's not possible to skip unknown shapes due to not knowing their lengths (which may be variable). However, this would be an issue for a purely XML-based implementation as well, so may not be a problem in practice.

Alignment Aligned shape variants are aligned at right-angles to the x and y (2D) or x, y and z (3D) axes.

11.5 Text placement and alignment

In order to annotate text next to measurements, it would be ideal if it were possible to control text placement and orientation. Currently the coordinate of the first letter is required. However, it would be nicer if the text could be also placed to the right of the point or centred on the point. And additionally, to the top, middle or bottom for vertical placement. Rotation would also be useful, though it's probably achievable indirectly via the transformation matrix, i.e. you would effectively have these anchors for placement, where 1 is the current behaviour.

```
7      8      9
4Text h5ere...6
1      2      3
```

This is needed to e.g. align text along measurement lines. Having a rotation angle specified directly would also save the need for complex calculations to work out the rotation origin and transform every time you want to just place a label along a line. It also makes it possible to place text in the centre of a shape.

11.6 Scale bars

Note: A 3D scale may need to be a 3D grid to allow visualisation of perspective, in which case the representation will define the grid bounding cuboid; inherit AlignedCuboid3D representations. Permit scale rotation with Cuboid3D? Allow specification of grid size and only allow sizing in discrete units?

11.7 Additional primitives

3D spline surfaces Natural cubic spline (Catmull-Rom)

The axiovision curve type is most likely a natural cubic spline, the curve passing smoothly through all points, but without local control. It is simply represented as a list of points through which the curve must pass; there are no additional control points. Depending upon if they are doing any custom stuff, it might not be possible to represent with pixel-perfect accuracy.

Curves might be more generally applicable to other formats, and useful in their own right. It might be worth considering adding a spline type with local control where the curve passes straight through the control points such as Catmull-Rom splines. This would make it very simple for non-experts to fit smooth lines while annotating their images.

CHAPTER

TWELVE

REGIONS IN ARBITRARY DIMENSIONS

While it is possible to use geometric shapes to specify regions in physical dimensions, this does not translate meaningfully to arbitrary dimensions. The following primitives work in any "dimension" with a discrete or continuous range by permitting the selection of specific values, or sub-ranges.

These "nD" shapes (Value, Values, Range) and the extrusion and combining shapes (Extrude, Combine), permit the specification of ROIs in multiple arbitrary dimensions, and their combination with geometry in 1D, 2D and 3D.

Just as all 1D, 2D and 3D geometry can be converted to the respective 1D, 2D or 3D bitmask or greymask representing the described shape, all nD primitives in higher dimensions can be converted to 1D bitmask or greymask. A 1D bitmask for each dimension will allow efficient iteration over the higher-order dimensions using the resulting bitmaps.

By default, a ROI is unconstrained within all dimensions. The addition of constraints restricts it to particular dimensions, or subsets thereof.

Note: RL. Should we be unconstrained by default, or completely constrained? Should this behaviour be different for "real" dimensions (xyzt) compared with virtual dimensions such as channels?

Should we constrain the ROI to a single timepoint when tracking?

Should x/y/z be blocked for nD operations? I.e. don't allow the nD shapes to specify regions in 3D space, and restrict them to non-physical dimensions.

CHAPTER

THIRTEEN

COMPOUND ROIS

A ROI may consist of multiple shapes combined in different ways. The result is also a shape.

13.1 Set primitives

Shapes may combined using set operators:

- union
- intersection
- difference
- symmetric difference

The shape is the result of the set operation.

Note: Restrict to combinations of 2D or 3D shapes only?

Note: **J-M Burel** Union in the mathematical sense or aggregation.

13.1.1 Set

A simple collection of shapes. There is no implied relationship unless used with the set operators.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation
NSHAPE	Count	Number of shapes
SHAPE1	Shape	First shape
…	Shape	Further shapes
SHAPEn	Shape	Last shape

13.1.2 Union

Produce the union of the shapes in the provided set.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation
SET	Set*	Set of shapes

13.1.3 Intersection

Produce the intersection of the shapes in the provided set.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation
SET	Set*	Set of shapes

13.1.4 Difference

Produce the set difference of the shapes in the provided set.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation
SET	Set*	Set of shapes

13.1.5 Symmetric difference

Produce the symmetric difference of the shapes in the provided set.

Representation:

Name	Type	Description
S1	ShapeID	Shape
R1	RepID	Representation
SET	Set*	Set of shapes

- Restrict to either 2D or 3D, but not both?

How do we detect if shapes intersect? Edge cases for set operations using masks-false positives for partially occupied pixels.

Event/Events: A simple list of points. The point size/style/colour may be changed to permit different sets to be distinguished.

Caliper/Distance/Multi-Caliper/Multi-Distance. These are all the same measurement(s), a baseline followed by a list of points. The measurement is the distance from each point to the baseline. The differences between the types are solely the visual presentation of the measurements.

Angle3/Angle4. These measure the angle between two lines. Angle4 is two separate lines, while Angle3 is two lines with a common point (i.e. a special case of Angle4). Angle3 could be represented with a three-point polyline. Angle4 would need to be two separate lines. Given that Angle3 is a special case of Angle4, it is not clear that it should be represented as a polyline.

Circle. While the OME model represents this as an ellipse with equal x and y radii, there are three ways to represent a circle here: - radius defined as a line from centre to edge - radius defined as a line from edge to centre (stored as the first type with the point order reversed) - circumference defined using three points. The first two are representable in the model as an ellipse plus a line. The latter is representable as an ellipse plus three points.

Polyline is directly translatable.

Aligned Rectangle is directly translatable as a rectangle (with some trivial differences in coordinates). However, additional tags define metadata to display inside the rectangle (optional) such as channel/slide/acquisition time/exposure time/etc. The verbatim text can be put into a Label, but the specific meaning would be lost–this is an overlay which would change as you navigate through a stack or timecourse etc, varying with the plane-specific parameters. While the specific tags would be retained, a more generic means to overlay image- and plane-specific OME metadata might be generally useful within the context of the OME model.

Ellipse is directly translatable.

Outline/closed polyline is directly translatable.

Text is convertible to Label. However, the OME Label type lacks the alignment attributes mentioned in my earlier mail. This makes it difficult to control the placement of text in complex compound ROIs.

Length is a single line distance measurement line like, but with additional end lines to make it like a technical drawing line outside the object itself, i.e.

```
|******OBJECT*******|
|                   |
|<----------------->|
         50 µm
```

Representable in the model as a simple line, across OBJECT, but with loss of the other lines. It is representable as three separate lines, but with loss of the context of the specific measurement.

Open and closed splines: these are probably natural splines (not Bezier). ZVI currently stores them as polylines given that we don't support splines. But having a spline type would permit them to be stored.

LUT and Profile: Covered in previous mail.

13.2 Storing and manipulating complex compound objects

With these measurements, one thing perhaps worth considering is that there are up to four types of object here:

1. Result context: the object(s) representing the physical measurement. This is what we currently store in the model.
2. Measurement context: line along radius of circle, points along circumference of circle etc. This is "how the measurement was made"
3. Visual context: such as visual cues such as construction lines. This is the visual presentation of the measurement to the viewer.
4. Editing context: values which control the placement of the above. Information for generation of UI manipulation handles, and of the other contexts while editing.

We can represent the actual measurements in most cases using the existing ROI types. However, if we store the additional types, it is no longer possible to distinguish between the measurement and the additional context.

If it was possible to distinguish between these in the model, it would be possible for the objects to be displayed without any advanced knowledge of how an object should be edited. It would also be possible to extract the primitive measurement values. However, the measurement context would provide additional information to editors for manipulation of the object, which would then be able to update all three contexts appropriately.

Doing this would provide a simple but effective means for additional ROI types to be added without requiring support in all programs displaying/modifying ROIs. This does not of course replace the need for namespaces to identify ROI categories, but it does supplement it by allowing programs to selectively display different contexts without any knowledge of the underlying type.

As an example, using this length measurement:

```
|******OBJECT*******|
|                   |
|<----------------->|
        50 µm
```

1. Result context

```
 #******OBJECT*******#

(where the #s are the start and end points of a Line at either end
of the object.  This is the value of the physical measurement.)
```

2. Measurement context

   No additional information needed in this case.

3. Visual context

```
 |                   |
 |                   |
 |<----------------->|
         50 µm

Three lines, one with arrow end markers, plus text label.
This is the visual representation of the measurement.
```

4. Editing context

```
#******OBJECT*******
#
#

(where the #s represent a distance between the measured line and
the drawn line in the visual context.  This information is used to
generate the visual context from the measurement context.)
```

I hope the above does not sound too way out. But the current system is limited to storing only the first of these four contexts, which loses information. While it is possible to delegate all of the presentation and editing to the viewer, the reality is that this is stuff people want. If I'm annotating an image for a paper, I want the annotations to appear exactly the same as I see them if I send them to someone else. And if I'm doing physical measurements, I want the specifics of how I made the measurement to be recorded. All we are doing here is providing additional information to the viewer/editor that it is free to use and/or ignore as it chooses.

Thinking about this a little more, in many cases it will be possible to omit some contexts and infer them from the others. For example, if I have a simple line I will store a line in the result context. The measurement context is the same two points, and so we may simply use the result context points in its place. Likewise, if the measurement is a simple one, the visual context may be omitted and inferred from the result context also. The different contexts really only come into play when we want a more sophisticated visual representation (for example with overlaid textual representations of the measurement value or to visualise the measurement in a more complex manner than the result context alone can provide). And they are essential when using more complex compound ROIs as the last example attached shows.

In the last example, all the information is provided to allow the user to edit the object in a UI. For example, they can adjust the end points of the baseline, and the start points of the lines in the measurement context can be retriangulated from the end points and baseline. The measurement context can be inferred from the endpoints of the lines in the result context. And the endpoints can also be adjusted independently. Following any adjustment, the updated baseline can be stored in the editing context, the measurement lines in the measurement context, and the visual representation in the visual context. The visual context is shown here to include end markers on the distance lines, and text labels with the measured values. But these could be toggled on or off and the settings stored in an annotation specific for this measurement type–there's really no limit to the "extra stuff" you can add here, but the basic measurement remains the same in the result context.

(In this example, the baseline could actually be in the measurement context, since it's part of the measurement; the first example is a better illustration of the editing context.)

The important point is that anyone should be able to open the file and display the visual representation without any knowledge of the specifics of the ROI type or measurements being made. Likewise they can also look at the measured distances in the results context and use them without any knowledge of how they were measured. Only a UI which supports the ROI type in question will need to use the editing and/or measurements context, and they will know how to regenerate the other contexts when editing.

CHAPTER

FOURTEEN

COMPOUND TYPES

Line Profile LUT Scale bar

LUT/gradient boxes are quite specialist. However, they are also quite common in published figures, so it would make sense to have a general implementation. These are particularly useful when you have false colour heat maps where you need a visual scale to interpret the figure. We already support LUTs, so this is really just a view of the LUT for a given channel inside a rectangle.

Line profiles are quite common. But I guess supporting this would depend upon whether you classify the profile as the result of analysis of a ROI, or part of a ROI. It might be handy to be able to overlay a line profile as a set of coloured polylines, for example.

14.1 Zeiss AxioVision ROI types

For the Zeiss types, we can represent these in the model using:

Zeiss type	ROI model type
Event	Point2D
Events	Point2D (union of points)
Line	Line2D
Caliper	Line2D (union of lines)
Multiple caliper	Line2D (union of lines)
Distance	Line2D (union of lines)
Multiple distance	Line2D (union of lines)
Angle3	Line2D and Arc2D
Angle4	Line2D and Arc2D
Circle	Circle2D and Line2D
Scale Bar	Line2D (with end markers)
Polyline [open]	Polyline2D
Aligned Rectangle	AlignedRectangle2D
Rotated Rectangle	Rectangle2D
Ellipse	AlignedEllipse2D
Polyline [closed]	Polygon2D
Text	Label2D
Length	Line2D (union of lines)
Spline [open]	PolylineSpline2D
Spline [closed]	PolygonSpline2D
LUT	AlignedRectangle2D and Label2D
Line profile	Line2D and Polyline2D/Rectangle2D

Annotations don't typically have labels (with the exception of scale bars). Measurements would have one or more labels in the union as well displaying the value(s) of the measurement.

CHAPTER

FIFTEEN

AFFINE TRANSFORMS

To support proper 3D operation, it would make sense to extend the existing support for 3×3 2D affine transforms to 4×4 3D transforms.

For both 2D and 3D transforms, translation, rotation and scaling are supported. Skewing, using the bottom row of the matrix, is not.

15.1 2D transforms

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

15.2 3D transforms

$$\begin{bmatrix} a & d & g & j \\ b & e & h & k \\ c & f & i & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CHAPTER

SIXTEEN

MASKS

16.1 Specification

Bitmasks may be specified directly, e.g. by segmenting an image. Bitmasks may also be derived from any shape, since every shape is reducible to a bitmask.

Greymasks (masks with multiple greylevels) may also be specified directly. They may also be derived from any shape. Shapes may provide direct conversion to a greymask, or alternatively via a high-resolution bitmask, which is then converted into a greymask. This process is illustrated in the following figure.

Masks have aligned and unaligned variants. The difference is not in the mask data, but in the alignment of the bounding box with the axes. Neither guarantee a 1:1 mapping with the pixel grid; this would require a manual conversion step. This might also be be better supported with a pixel-aligned bounding box type. Resizing to the pixel grid might be best performed via thresholding an intermediate greymask.

Any shape transformations must be performed prior to conversion to an aligned mask, otherwise the mask alignment may be lost.

Note:

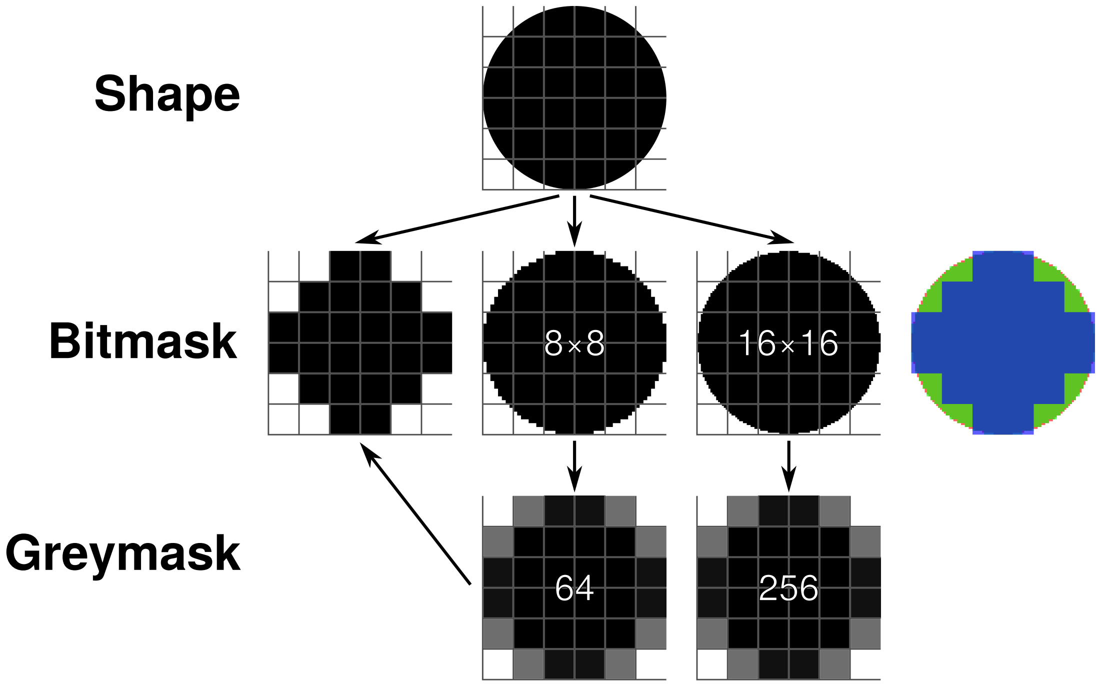
Roger We need to specify the criteria for pixel inclusion when converting from a shape to bitmask. Some shapes may be able to efficiently convert to a greymask, but a threshold value is still needed.

We also need to allow the user to specify the threshold value when converting from a greymask to bitmask.

Also need to have rules for conversion of points and lines, which do not have any intrinsic area, to pixels. Does a point always occupy a single pixel? How about lines, which pass through multiple pixels? The latter could convert to a greymask. Both could have default widths and allow the user to override them. Convert via a shape e.g. implicitly convert line to cuboid and point to sphere?

The current mask representations store the mask data directly in the shape. We might wish to support alternative forms of storage, e.g. IFD (as a sprite sheet), labellings, etc.

A circle, drawn a 6×6 pixel grid may be converted directly as a 6×6 pixel bitmap. Alternatively, the grid may be subdivided further so that each pixel is itself an 8×8 pixel grid, to give a grid size of 48×48 pixels. Each real pixel therefore contains 256 bits of information, from which it is trivial to derive a 6×6 pixel 6-bit greymask with 256 grey levels. The resolution may be further increased so that each pixel is a 16×16 pixel grid from which an 8-bit greymask with 256 greylevels may be derived.



The following grid sizes could be used:

Grid size	Grid bits	Greylevel bits	Greylevels
2×2	4	2	4
4×4	16	4	16
8×8	64	6	64
16×16	256	8	256
32×32	1024	10	1024
64×64	4096	12	4096
128×128	16384	14	16384
256x256	65536	16	65536

Note:

Roger We don’t need to support all these sizes, but supporting 8 bit masks at a minimum would be useful. Larger sizes would have greater precision, but quite a large overhead: a 16 bit greymask requires 8KiB/pixel!

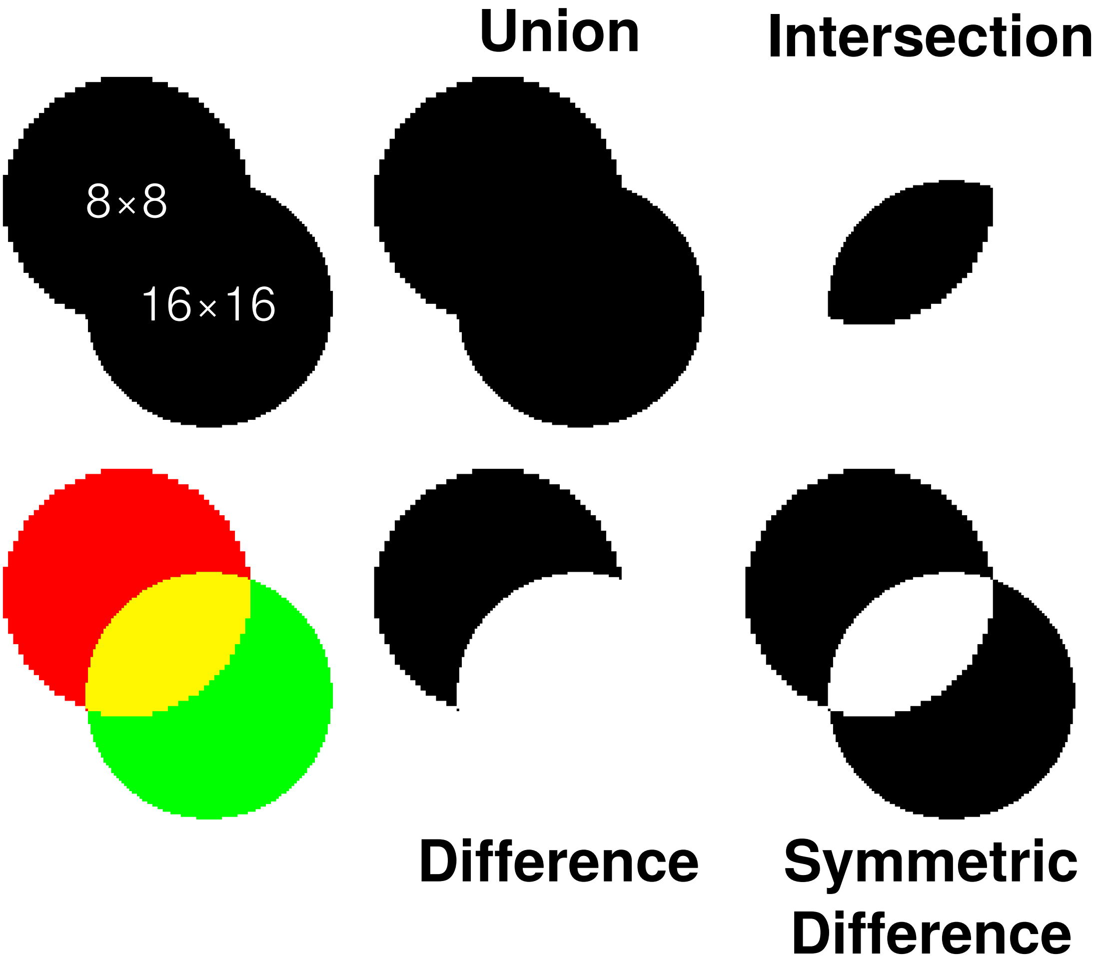
16.2 Point and line conversion

Would it make sense to have the ability to convert point and line shapes to cylinder/sphere or cuboid shapes, respectively? Useful for rendering, and potentially also useful for analysis. Default point size and line width for converting to a mask? Points may be expected to only be one pixel in size; what about lines?

16.3 Set operations

Set operations only make sense to perform at the level of bitmasks. Set operations on basic shape geometry rapidly becomes an intractable problem, since this for example requires that it be possible to describe the union of every shape type with every other shape type, including all combinations of unions. This would be possible if all geometry was reduced to meshes, but this would also result in a loss of precision.

Set operations are trivial to perform using masks. However, as shown in the above figure, there may be loss of precision when converting to a mask. However, it would be possible to do the set operations on a higher-resolution mask prior to conversion to a greymask or lower-resolution bitmask. This includes intersection, set difference, etc.



Note:

Roger Consider a union of two shapes which do not touch, but which overlap a common pixel. It is possible to compute the union using the higher-resolution bitmask because this takes into account the extent to which the shapes overlap (or not), and this can be reflected in the resulting greymap. The user can choose the precision of the operation via the grid size

CHAPTER

SEVENTEEN

TRANSFORMS

The model defines shapes for performing affine transforms and abstract transforms (*scijava.roi.types.AffineTransform3D* and *scijava.roi.types.AbstractTransform3D*). The purpose of the abstract transform is to serve as a hook mechanism for implementation-specific transformations to be supported within the model.

All transforms are shapes. The implication is that all transformations on shapes evalulate to the transformed shape, i.e. the transform shape *is* the transformed shape.

Transforms between pixel space and physical space (using the unit system defined in the image metadata). Provide both transforms.

If shapes can be defined in either space, should any of these transforms be implicit? If so, when are they applied?

Additional transforms required for display? physical to pixels is equivalent to the modelview transformation matrix. Should we additionally take into account projection/perspective/viewport matrices? Or leave further transformation to the implementor, e.g. starting from shapes reduced to meshes, for OpenGL implementations.

Conversion of shapes to masks needs to happen in pixel space?

In the current model, transforms are specified inline in the shape definition. However, it may make sense to have some transforms out of band in the ROI or shape state, such as pixel to physical (and inverse) transforms. This would require a transform representation with a transform ID as one of its data members.

CHAPTER

EIGHTEEN

STATE MACHINE

Evaluating a ROI will require a simple state machine to handle the transformations involved.

18.1 Properties

Table 18.1: State machine properties

Property	Type	Description
LINECOL	Colour	Line (and surface) colour
FILLCOL	Colour	Fill colour
TEXTCOL	Colour	Text colour

CHAPTER

NINETEEN

LAYERS

In the Zeiss AxioVision formats, ROIs (shapes) are contained within Layers. Sets of ROIs are collected in different layers. The UI only uses a single layer, but uses separate layers for acquisition and post-acquisition ROIs. But in the file format one may define arbitrary numbers of layers to act as a grouping mechanism for ROIs.

Adding layers as a top level grouping would permit related ROIs to be grouped together. However, this would also be possible using ROI-ROI links; it could be implemented using Layer-ROI links. Maybe layer could be a ROI type used solely for grouping?

CHAPTER

TWENTY

ROI-ROI LINKS

ROI relationships: When segmenting cell contents, shown as cytoplasm, actin filaments, nucleus and nucleolus, these fall into a strict hierarchy (a nucleus can only be in one cell, though one cell could have more than one nucleus). If we added a ROI type that was a container of ROIs (note: not a union), and added a means of classifying ROIs with tags/labels, this would be very useful for HCS and other types of analysis. Additionally, some relationships are not hierarchical, e.g. tree-like branching and merging in a vessel bed, but could be represented if a ROI could point to one or more other ROIs, which would permit a directed graph of relationships between ROIs.

Tracking Containment User modification (branch/merge) Inherit properties Layer DAG

CHAPTER

TWENTYONE

STORAGE OF VERTEX DATA

For quite a number of the shape primitives, it is possible to support 3D very simply–we just increase the number of dimensions in each vertex, and that's it (obviously just for storage; it will still require some work for rendering). From the point of view of storing the list of vertices, it would be nice if we could specify the dimensions being used e.g. XZT, and then allow missing dimensions to be specified as constants as we now do for theZ. This will also mean that will will be possible to use a 2D primitive with theZ set as equivalent to a 3D primitive with the z value specified separately to the (x,y) points. This would provide one means of keeping the representation compact. Additionally, it is undesirable to have a separate element for each vertex, since for complex shapes e.g. meshes, this would waste a lot of space: when scaling up to thousands of vertices, this would waste multi-megabytes of XML markup for no good reason.

21.1 XML schema

Shape type and representation are stored as unsigned 16 bit integers, counts as unsigned 32 bit integers, and vertices and vectors as double-precision floating point.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualifie
    <xsd:simpleType name="shapeDetailElement">
        <xsd:union memberTypes="xsd:unsignedShort xsd:unsignedInt xsd:double" />
    </xsd:simpleType>
    <xsd:simpleType name="shapeDetail">
        <xsd:list itemType="shapeDetailElement"/>
    </xsd:simpleType>
    <xsd:element name="shape">
        <xsd:complexType>
            <xsd:simpleContent>
                <xsd:extension base="shapeDetail">
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

21.2 Properties

Store at the level of the ROI, not the shape. Since all the shapes within a ROI describe a single entity, there is no need for separate properties (colour, line thickness/style/endings etc.) on each shape.

Note: **J-M Burel**: previous discussion about that. Need to review the notes taken at the time.

CHAPTER

TWENTYTWO

DEFINITION OF TERMS

ROI Region of interest. A subset of samples within an image. This is specified by the boundary or surface of the object.

Shape Geometric shape or mask. A shape is a geometric primitive or bitmask. A ROI is composed of one or more shapes.

CHAPTER

TWENTYTHREE

INDICES AND TABLES

- *genindex*
- *search*

INDEX

T