

Module Interface Specification for ImgBeamer

Joachim de Fourestier

March 20, 2023

1 Revision History

Date	Version	Notes
2023/03/18	0.1.0	Creation
	0.1.1	Update module hierarchy
2023/03/19	0.1.2	Add in module specifications
2023/03/20	0.1.3	Add in minor missing type information.

2 Symbols, Abbreviations and Acronyms

symbol	description
HID	Human Interface Device
URL	Uniform Resource Locator

See the SRS [\[2\]](#) and MG [\[1\]](#) Documentation for additional items.

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	3
6	MIS of Application Control	4
6.1	Module	4
6.2	Uses	4
6.3	Syntax	4
6.3.1	Exported Constants	4
6.3.2	Exported Access Programs	4
6.4	Semantics	4
6.4.1	State Variables	4
6.4.2	Environment Variables	4
6.4.3	Assumptions	4
6.4.4	Access Routine Semantics	4
6.4.5	Local Functions	5
7	MIS of Graphical User Interface (GUI)	6
7.1	Module	6
7.2	Uses	6
7.3	Syntax	6
7.3.1	Exported Constants	6
7.3.2	Exported Access Programs	6
7.4	Semantics	6
7.4.1	State Variables	6
7.4.2	Environment Variables	7
7.4.3	Assumptions	7
7.4.4	Access Routine Semantics	7
7.4.5	Local Functions	8
8	MIS of Information and Metrics Display	9
8.1	Module	9
8.2	Uses	9
8.3	Syntax	9
8.3.1	Exported Constants	9
8.3.2	Exported Access Programs	9

8.4	Semantics	9
8.4.1	State Variables	9
8.4.2	Environment Variables	9
8.4.3	Assumptions	9
8.4.4	Access Routine Semantics	9
8.4.5	Local Functions	10
9	MIS of Image Export	11
9.1	Module	11
9.2	Uses	11
9.3	Syntax	11
9.3.1	Exported Constants	11
9.3.2	Exported Access Programs	11
9.4	Semantics	11
9.4.1	State Variables	11
9.4.2	Environment Variables	11
9.4.3	Assumptions	11
9.4.4	Access Routine Semantics	11
9.4.5	Local Functions	12
10	MIS of Display Control	13
10.1	Module	13
10.2	Uses	13
10.3	Syntax	13
10.3.1	Exported Constants	13
10.3.2	Exported Access Programs	13
10.4	Semantics	14
10.4.1	State Variables	14
10.4.2	Environment Variables	14
10.4.3	Assumptions	14
10.4.4	Access Routine Semantics	14
10.4.5	Local Functions	14
11	MIS of Drawing Stage / Canvas	15
11.1	Module	15
11.2	Uses	15
11.3	Syntax	15
11.3.1	Exported Constants	15
11.3.2	Exported Access Programs	15
11.4	Semantics	15
11.4.1	State Variables	15
11.4.2	Environment Variables	16
11.4.3	Assumptions	16

11.4.4	Access Routine Semantics	16
12	MIS of Image Rendering	17
12.1	Module	17
12.2	Uses	17
12.3	Syntax	17
12.3.1	Exported Constants	17
12.3.2	Exported Access Programs	17
12.4	Semantics	17
12.4.1	State Variables	17
12.4.2	Environment Variables	17
12.4.3	Assumptions	17
12.4.4	Access Routine Semantics	18
12.4.5	Local Functions	18
13	MIS of Image Metrics Calculation	19
13.1	Module	19
13.2	Uses	19
13.3	Syntax	19
13.3.1	Exported Constants	19
13.3.2	Exported Access Programs	19
13.4	Semantics	19
13.4.1	State Variables, Environment Variables, and Assumptions	19
13.4.2	Access Routine Semantics	19
13.4.3	Local Functions	19
14	MIS of Ground Truth Visualization	20
14.1	Module	20
14.2	Uses	20
14.3	Syntax	20
14.3.1	Exported Constants	20
14.3.2	Exported Access Programs	20
14.4	Semantics	20
14.4.1	State Variables	20
14.4.2	Environment Variables and Assumptions	20
14.4.3	Access Routine Semantics	21
14.4.4	Local Functions	21
15	MIS of Subregion Visualization	22
15.1	Module	22
15.2	Uses	22
15.3	Syntax	22
15.3.1	Exported Constants	22

15.3.2	Exported Access Programs	22
15.4	Semantics	22
15.4.1	State Variables	22
15.4.2	Environment Variables	22
15.4.3	Assumptions	22
15.4.4	Access Routine Semantics	23
15.4.5	Local Functions	23
16	MIS of Spot Profile Visualization	24
16.1	Module	24
16.2	Uses	24
16.3	Syntax	24
16.3.1	Exported Constants	24
16.3.2	Exported Access Programs	24
16.4	Semantics	24
16.4.1	State Variables	24
16.4.2	Environment Variables	24
16.4.3	Assumptions	24
16.4.4	Access Routine Semantics	25
16.4.5	Local Functions	25
17	MIS of Spot Content Visualization	26
17.1	Module	26
17.2	Uses	26
17.3	Syntax	26
17.3.1	Exported Constants	26
17.3.2	Exported Access Programs	26
17.4	Semantics	26
17.4.1	State Variables	26
17.4.2	Environment Variables	26
17.4.3	Assumptions	27
17.4.4	Access Routine Semantics	27
17.4.5	Local Functions	27
18	MIS of Spot Signal Visualization	28
18.1	Module	28
18.2	Uses	28
18.3	Syntax	28
18.3.1	Exported Constants	28
18.3.2	Exported Access Programs	28
18.4	Semantics	28
18.4.1	State Variables	28
18.4.2	Environment Variables and Assumptions	28

18.4.3	Access Routine Semantics	29
18.4.4	Local Functions	29
19	MIS of Spot Layout Visualization	30
19.1	Module	30
19.2	Uses	30
19.3	Syntax	30
19.3.1	Exported Constants	30
19.3.2	Exported Access Programs	30
19.4	Semantics	30
19.4.1	State Variables	30
19.4.2	Environment Variables and Assumptions	30
19.4.3	Access Routine Semantics	31
19.4.4	Local Functions	31
20	MIS of Sample Subregion Visualization	32
20.1	Module	32
20.2	Uses	32
20.3	Syntax	32
20.3.1	Exported Constants	32
20.3.2	Exported Access Programs	32
20.4	Semantics	32
20.4.1	State Variables	32
20.4.2	Environment Variables and Assumptions	32
20.4.3	Access Routine Semantics	33
20.4.4	Local Functions	33
21	MIS of Resulting Subregion Visualization	34
21.1	Module	34
21.2	Uses	34
21.3	Syntax	34
21.3.1	Exported Constants	34
21.3.2	Exported Access Programs	34
21.4	Semantics	34
21.4.1	State Variables	34
21.4.2	Environment Variables and Assumptions	35
21.4.3	Access Routine Semantics	35
21.4.4	Local Functions	35
22	MIS of Result Image Visualization	36
22.1	Module	36
22.2	Uses	36
22.3	Syntax	36

22.3.1	Exported Constants	36
22.3.2	Exported Access Programs	36
22.4	Semantics	36
22.4.1	State Variables	36
22.4.2	Environment Variables	36
22.4.3	Assumptions	37
22.4.4	Access Routine Semantics	37
22.4.5	Local Functions	37

3 Introduction

The following document details the Module Interface Specifications for ImgBeamer (SEM image formation demo tool). Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/joedf/CAS741_w23.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper [4], with the addition that template modules have been adapted from [3]. The mathematical notation comes from Chapter 3 of Hoffman and Strooper [4]. For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by ImgBeamer.

Data Type	Notation	Description
integer	\mathbb{Z}	a number without a fractional component in $(-\infty, \infty)$
positive integer	\mathbb{Z}_+	a positive integer (\mathbb{Z}) in $(0, \infty)$
unsigned 8-bit integer	\mathbb{U}	a number without a fractional component in $(0, 255)$
natural number	\mathbb{N}	a number without a fractional component in $[1, \infty)$
real	\mathbb{R}	any number in $(-\infty, \infty)$
positive real	\mathbb{R}_+	any real number in $(0, \infty)$
unit interval	\mathbb{A}	any real number in $(0, 1)$
imageData [6]	$\mathbb{I}_{w,h}$	data: a one dimensional array of positive integers from 0 to 255 in RGBA order (pixel components) start from the top left pixel to the bottom right pixel with a width: \mathbb{Z}_+ width of w and height: \mathbb{Z}_+ height of h .

The specification of ImgBeamer uses some derived data types: sequences, strings, tuples, and **drawingObject**. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. **drawingObject** is a geometry object (provided by Konva): the fill can be an image, a colour, or even another shape. They can have a width, height, rotation, position (x,y), and many other properties. **imageDrawingObject** is essentially **drawingObject** with an image as the fill. In addition, ImgBeamer uses functions, which are defined by the data types of their

inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide [1] document for this project.

Level 1	Level 2	Level 3
Hardware-Hiding Module		
Behaviour-Hiding Module	Application Control	
	Input	Ground Truth Image Input
		Imaging Parameters Input
		Spot Profile Input
	Output	Information and Metrics Display
		Image Export
	Visualization Display	Ground Truth
		Subregion
		Spot Profile
		Spot Content
Spot Signal		
Spot Layout		
Sampled Subregion		
Resulting Subregion		
Resulting Image		
Software Decision Module	Display Control	
	Graphical User Interface	
	Image Manipulation	Drawing Stage / Canvas Module
		Rendering
Metrics Calculation		

Table 1: Module Hierarchy

6 MIS of Application Control

6.1 Module

main (M2)

6.2 Uses

GUI Module Specification (7)

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

None

6.4.3 Assumptions

- The application is run in an HTML5 compliant web browser.
- The GUI is running and displayed without issue.

6.4.4 Access Routine Semantics

main():

- transition: Initializes the GUI, modifies the state and environment variables of the GUI Module Specification (7).

6.4.5 Local Functions

`UpdateBaseImage()`: Updates the GUI and propagates a change in the input ground truth image throughout the application.

7 MIS of Graphical User Interface (GUI)

7.1 Module

gui (M18)

7.2 Uses

Hardware Hiding Module (M1), Display Control Module (M17), Ground Truth Image Input Module (M3), Imaging Parameters Input Module (M4), Spot Profile Input Module (M5), Image Export Module (M6), Information and Metrics Display Module (M7)

7.3 Syntax

7.3.1 Exported Constants

- **G_BoxSize**: A value (\mathbb{N}) describing both the pixel width and height used for each visualization display “box”.
- **G_MATH_TOFIXED**: Used for display for fixed decimal number length rounding (ex. “4.1234” at fixed length “2” results in “4.12”).

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
gui	baseImage ($\mathbb{I}_{w,h}$)	displayReference, event handlers	-

7.4 Semantics

[Didn't do MIS descriptions of the Input modules because they are essentially just buttons or text boxes with event handlers. Can be implemented however a developer wishes as long the SRS value constraints are followed... Or are full descriptions also needed for these? Not sure if they would add much value than already provided here or just informational noise. Maybe I can write this as a note here (instead of a comment)? —Author]

7.4.1 State Variables

- **baseImage**: The ground truth image as processed and given by M3 as $\mathbb{I}_{w,h}$.
- **resultImage**: A reference to resulting image as processed and given by the Display Control M17 as $\mathbb{I}_{w,h}$.
- **imageRows**: Rasterization grid rows given by M4 as \mathbb{Z}_+ .

- **imageCols**: Rasterization grid columns given by M4 as \mathbb{Z}_+ .
- **imageMag**: Magnification of the subregion as given by M4 as \mathbb{R}_+ .
- **spotWidth**: The spot's width given by M5 as \mathbb{Z}_+ .
- **spotHeight**: The spot's height given by M5 as \mathbb{Z}_+ .
- **spotAngle**: The spot's angle given by M5 as \mathbb{R} .
- **dispControl**: a reference to the Display Control (M17).

7.4.2 Environment Variables

- Keyboard
- Mouse
- Screen
- File System

7.4.3 Assumptions

- The file system is able to read and provide the image file as specified by the user through an OS file-open dialog. Otherwise, if the file is not found, denied access, or cancelled, no changes should occur.
- The OS and WebBrowser are able to provide basic text or number input user controls with some basic built-in validation, and is able to handle events from Human Interface Devices (HIDs such as a mouse, keyboard, or touchscreen).

7.4.4 Access Routine Semantics

OnImageLoaded():

- **transition**: Sets up user control event handlers (e.g., mouse clicks or drag, button presses, text input change, ...) as needed for the user input modules (M3, M4 and M5), initializes the Display Control Module (M17) with the individual GUI draw controls/locations for each visualization and obtains an update function reference for re-draws or state changes. If another image is loaded (i.e. the image is changed), the Display Control is reinitialized with the new image.

[I am not sure what transition means, couldn't find it as a defined term in the slides. Defined what the function/method does. I hope this is right, continuing as so... — Author]

- **output**:

- `doUpdate()`: notifies the Display Control Module (M17) to update / redraw the visualization displays.
- `updateInfoDisplay()`: notifies the Information Display Module (M7) to update when needed (such as an input value change from the mentioned input modules).
- `doExport()`: Event handler for the "Export" button press, it calls the Image Export Module (M6).

7.4.5 Local Functions

`doUpdate()`, `updateInfoDisplay()`, and `doExport()` as described above.

8 MIS of Information and Metrics Display

8.1 Module

infoDisp (M7)

8.2 Uses

Metrics Calculation Module (M21)

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
updateInfo	textDisplayControl, ($\mathbb{I}_{w,h}$), resultImage ($\mathbb{I}_{w,h}$), imageMag (\mathbb{R}_+)	baseImage -	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

The decimal length for rounding the number for display as defined by Module Specification (7).

8.4.3 Assumptions

- A suitable display control (capable of displaying text and numbers) is constructed and displayed in the GUI for use by this module.

8.4.4 Access Routine Semantics

updateInfo(textDisplayControl, baseImage, resultImage, imageMag):

- `transition`: Calls the Metrics Calculation Module (M21) to compare the given images (`baseImage` and `resultImage`) to get metric value (\mathbb{R}). The magnification (`imageMag`) and metric values are then rounded and pushed as formatted descriptive text to `textDisplayControl`.

8.4.5 Local Functions

None

9 MIS of Image Export

9.1 Module

imgExport (M6)

9.2 Uses

None

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
export	resultImage $(\mathbb{I}_{w,h})$, outputPath (string)	ImageFile	InvalidPath

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

The File System.

9.4.3 Assumptions

The output location is a valid writable and accessible.

9.4.4 Access Routine Semantics

export(resultImage, outputPath):

- output: an image file representing **resultImage** at location **outputPath**.
- exception: **InvalidPath** meaning the location cannot be written to, either because the directory is nonexistent, the path contains invalid characters, or inadequate write permissions.

9.4.5 Local Functions

`GetSuggestedFileName()`: generates a suggested filename with a timestamp that is displayed in the save-file dialog where possible.

10 MIS of Display Control

10.1 Module

dispControl (M17)

10.2 Uses

1. Rendering Module (M20)
2. Ground Truth Visualization Module (M8)
3. Subregion Visualization Module (M9)
4. Spot Profile Visualization Module (M10)
5. Spot Content Visualization Module (M11)
6. Spot Signal Visualization Module (M12)
7. Spot Layout Visualization Module (M13)
8. Sampled Subregion Visualization Module (M14)
9. Resulting Subregion Visualization Module (M15)
10. Resulting Image Visualization Module (M16)

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
Init	gtImage drawControls: controls for 2 to 10 in section 10.2	$(\mathbb{I}_{w,h})$, a reference to the GUI doUpdate function for each	-
doUpdateAll	-	-	-

10.4 Semantics

10.4.1 State Variables

- References to drawing stages/canvases for all the visualization/display modules mentioned in section [10.2](#)
- ... and each corresponding update function references
- `gtImage`: a reference to the ground truth image data (as provided by [M3](#)).
- `subregionImage`: a reference to `imageDrawingObject` (as provided by [M9](#)).

10.4.2 Environment Variables

- Display/Canvas controls as provided by the GUI Module ([M18](#)).
- Mouse
- Keyboard

10.4.3 Assumptions

None

10.4.4 Access Routine Semantics

`Init(gtImage, drawControls...):`

- `transition`: Initializes the drawing stages/canvases in each of the draw-control locations (`drawControls`) as provided by the GUI Module ([M18](#)) and passes them to each corresponding visualization module.
- `output`: a `doUpdate` function reference for each of the visualization modules.

`doUpdateAll():`

- `transition`: updates all the visualization displays by calling all the corresponding `doUpdate` function references.

10.4.5 Local Functions

None

11 MIS of Drawing Stage / Canvas

11.1 Module

stage (M19)

11.2 Uses

None

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	container, width (\mathbb{Z}_+), height (\mathbb{Z}_+)	drawing stage	ContainerNotFound
getLayers	-	array of the layers	-
getContext	-	drawing context [7]	-
getContainer	-	display control / container	-
toCanvas	-	canvasAPI object [7]	-
toDataURL	-	a URL to an exported image [7]	-

11.4 Semantics

Currently, using the implementing by the Konva [5] javascript library. Largely wraps around the HTML Canvas API object with added functionality such as layering and “transformers” for node-editable shapes.

11.4.1 State Variables

- width/height: the width and height of the drawing stage in pixels.
- Layers: drawing layers
- Container: the display control / container where to “paint” the images as provided by the GUI Module (M18).

- Event handlers: all the Konva objects (layers, geometries, stage) may have event handlers for HID events.

11.4.2 Environment Variables

The HIDs (e.g., mouse, keyboard) and the screen.

11.4.3 Assumptions

Any drawing exceptions will result in throwing errors that may be caught as needed, but will simply result in blank (or black) images with no interruption in any drawings in progress or drawing loops.

11.4.4 Access Routine Semantics

`init()`:

- transition: Initializes a drawing stage object with the given options where `container` is the control or location given by the GUI Module (M18).
- output: the drawing stage object.
- exception: `ContainerNotFound` meaning the given control is nonexistent or could not be found.

`getLayers()`:

- output: an array of all the individual drawing layers within the stage.

`getContext()`:

- output: the drawing context as defined by the CanvasAPI [7].

`getContainer()`:

- output: the display container as defined/given by the GUI Module (M18) when the stage is initialized.

`toCanvas()`:

- output: the canvasAPI element / object [7].

`toDataURL()`:

- output: a URL pointing to an image exported in-memory within the WebBrowser that can “downloaded” and saved a location specified by the user.

12 MIS of Image Rendering

12.1 Module

renderUtils (M20)

12.2 Uses

Drawing Stage / Canvas Module (M19)

12.3 Syntax

12.3.1 Exported Constants

defaultLineColor: the default line color (RGBA) to use for drawing grids (255,255,255,204)
- types: $(\mathbb{Z}_+, \mathbb{Z}_+, \mathbb{Z}_+, \mathbb{Z}_+)$.

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawGrid	gridLayer, rows (\mathbb{Z}_+) , cols (\mathbb{Z}_+) , lineColor	rect, cell size (width/height in pixels, \mathbb{Z}_+)	badGridParams
repeatDrawOnGrid	layer, rect, rows (\mathbb{Z}_+) , cols (\mathbb{Z}_+) , shape	-	badGridParams
ComputeProbeValue_gs	image $(\mathbb{I}_{w,h})$, probe	grayscale value (\mathbb{U})	-
get_avg_pixel_gs	rawImageData $(\mathbb{I}_{w,h})$	grayscale value (\mathbb{U})	-

12.4 Semantics

12.4.1 State Variables

None.

12.4.2 Environment Variables

The graphics processing hardware. [or optional software rendering? —Author]

12.4.3 Assumptions

None

12.4.4 Access Routine Semantics

`drawGrid(gridLayer, rect, rows, cols, lineColor = defaultLineColor):`

- transition: Draws a line (optional colour `lineColor`) grid with the specified number of `rows` and `cols` (columns) on the given drawing layer (`gridLayer`) within the given grid rectangular bounds (`rect`).
- output: the computed size in pixel of a cell within the grid drawn.
- exception: `badGridParams` meaning non-integer or non-positive values were given for `rows` and `cols`.

`repeatDrawOnGrid(layer, rect, rows, cols, shape):`

- transition: Draw a given geometry (`shape`) or `imageDrawingObject` repeated over a grid pattern with the specified number of `rows` and `cols` (columns) on the given drawing layer (`layer`) within the given grid rectangular bounds (`rect`).
- exception: `badGridParams` meaning non-integer or non-positive values were given for `rows` and `cols`.

`ComputeProbeValue_gs(image, probe):`

- transition: internally uses `get_avg_pixel_gs()` to calculate the pixel value of a locally composited or “stenciled” or “clipped” image (for sampling the region defined by the shape or `probe`, like a cookie cutter). Pixels that have been “stenciled” out are set to blank pixels (where all RGBA components are equal to 0) and the image is cropped to small rectangular bounding box of the “stencil” shape (`probe`).
- output: Gives the average pixel value (grayscale intensity: \mathbb{U}) by sampling the given image (`image`) object with the given shape / geometry (`probe`).

`get_avg_pixel_gs(rawImageData):`

- output: Gives the average pixel value (grayscale intensity: \mathbb{U}) from a given `imageData` array (`rawImageData`) of the RGBA pixel values ignoring any blank pixels (where all RGBA components are equal to 0).

12.4.5 Local Functions

None

13 MIS of Image Metrics Calculation

13.1 Module

metrics (M21)

13.2 Uses

None

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
compare	image1 ($\mathbb{I}_{w,h}$), image2 ($\mathbb{I}_{w,h}$)	similarity ratio (\mathbb{A})	DifferentImageSizes

13.4 Semantics

See the SRS [2] and MG [1] for more information.

13.4.1 State Variables, Environment Variables, and Assumptions

None

13.4.2 Access Routine Semantics

compare(image1, image2):

- transition: Compares the two images and computes a value representing the similarity.
- output: Gives a value (\mathbb{A}) where 1.0 means a perfect match and 0 means zero similarity.
- exception: DifferentImageSizes meaning the size of image1 and image2 do not match.

13.4.3 Local Functions

None

14 MIS of Ground Truth Visualization

14.1 Module

drawGroundtruthImage (M8)

14.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawGroundtruthImage	stage, gtImage, subregionImage	imageDrawingObject, doUpdate	-

14.4 Semantics

14.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to drawing stage.
- **rect**: a reference to a rectangle geometry.
- **gtImage**: a reference to imageDrawingObject for the ground truth image.
- **subregionImage**: a reference to imageDrawingObject for the subregion image.

14.4.2 Environment Variables and Assumptions

None

14.4.3 Access Routine Semantics

`drawGroundtruthImage(stage, gtImage, subregionImage):`

- **transition**: Defines a drawing arrangement to fill the stage with the ground truth image (`gtImage` as provided by the Display Control M17) with a semi-transparent rectangle (`rect`) representing the bounds of the `subregionImage` (as provided by the Display Control M17).
- **output**: an object with a reference to the update function (`doUpdate`) and a reference to `rect`.

14.4.4 Local Functions

`doUpdate()`: Update the drawing based on the state variables.

15 MIS of Subregion Visualization

15.1 Module

drawSubregionImage (M9)

15.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

15.3 Syntax

15.3.1 Exported Constants

None

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSubregionImage	stage, gtImage, subregionImage updateCallback		-

15.4 Semantics

15.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to drawing stage.
- **gtImage**: a reference to imageDrawingObject for the ground truth image.
- **subregionImage**: a reference to imageDrawingObject for the subregion image.
- **updateCallback**: an optional reference to a function call when an update occurs (i.e. when view bounds change).
- mouse events (scroll and drag)

15.4.2 Environment Variables

The HIDs.

15.4.3 Assumptions

None

15.4.4 Access Routine Semantics

`drawSubregionImage(stage, gtImage, updateCallback = null):`

- transition: Draw a view displaying a copy of the ground truth image (`gtImage` as provided by the Display Control M17) representing the current subregion / ROI. This view can be panned and zoomed with mouse events. The `updateCallback` function reference is called when mouse events (drag or scroll) causes the of the view bounds to change.
- output: a reference to `subregionImage` `imageDrawingObject` (which can be used like `rect`) representing the bounds of the current view.

15.4.5 Local Functions

`doUpdate()`: Update the drawing based on the state variables that change on mouse event such as dragging or scrolling events (pan and zoom) and calls the `updateCallback`.

16 MIS of Spot Profile Visualization

16.1 Module

drawSpotProfile (M10)

16.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSpotProfile	stage	beam (drawingObject)	-

16.4 Semantics

16.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to the drawing stage.
- **beam**: a reference to the ellipse geometry.

16.4.2 Environment Variables

The HIDs.

16.4.3 Assumptions

None

16.4.4 Access Routine Semantics

`drawSpotProfile(stage)`:

- transition: On the given drawing **stage**, draws an editable ellipse shape representing the beam/spot shape.
- output: a reference to the ellipse geometry (**beam**).

16.4.5 Local Functions

Mouse events handled by Konva for shape node-editing / “transformers”.

17 MIS of Spot Content Visualization

17.1 Module

drawSpotContent (M11)

17.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

17.3 Syntax

17.3.1 Exported Constants

None

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSpotContent	stage, subregionImage, sImage sBeam (drawingObject), updateCallback		-

17.4 Semantics

17.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to the drawing stage.
- **sImage**: a reference to the subregion image clone (imageDrawingObject).
- **sBeam**: a local reference to the beam “stencil” geometry clone (not changed).
- **updateCallback**: an optional reference to a function call when an update occurs (i.e. when the **sImage** position and scaling to change).
- mouse events (scroll and drag)

17.4.2 Environment Variables

The HIDs.

17.4.3 Assumptions

None

17.4.4 Access Routine Semantics

`drawSpotContent(stage, subregionImage, sBeam, updateCallback = null):`

- **transition:** On a given drawing stage (**stage**), draws an image clone (based on **subregionImage**) of the subregion (**sImage**) that is “stenciled” or clipped by the **sBeam** geometry/shape. This image can be panned and zoomed by mouse events. The **updateCallback** function reference is called when mouse events (drag or scroll) causes the image (**sImage**) position and scaling to changes.
- **output:** a reference to the image (**sImage**) being moved and scaled.

17.4.5 Local Functions

`doUpdate():` Update the drawing based on the state variables that change on mouse event such as dragging or scrolling events (pan and zoom) and calls the **updateCallback**.

18 MIS of Spot Signal Visualization

18.1 Module

drawSpotSignal (M??)

18.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

18.3 Syntax

18.3.1 Exported Constants

None

18.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSpotSignal	sourceStage, destStage, sBeam (drawingObject)	doUpdate	-

18.4 Semantics

18.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **sourceStage**: a reference to the drawing stage from M11 given by the Display Control M17.
- **destStage**: a reference to the drawing stage to the spot signal representation draw on.
- **sBeam**: a local reference to the beam “stencil” geometry clone (not changed).
- **doUpdate**: an optional reference to a function call when an update occurs.

18.4.2 Environment Variables and Assumptions

None

18.4.3 Access Routine Semantics

`drawSpotSignal(sourceStage, destStage, sBeam):`

- **transition:** On a given drawing stage (`destStage`), draws the `sBeam` geometry/shape filled in by the computed average pixel value from the clipped / “stenciled” image content as displayed in Spot Content (M11).
- **output:** a reference to an update function (`doUpdate`) to call (by the Display Control M17) when a redraw is needed (such as a change in Spot Content (M11)).

18.4.4 Local Functions

`doUpdate()`: Update the drawing based on the state variables.

19 MIS of Spot Layout Visualization

19.1 Module

drawSpotLayout (M13)

19.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

19.3 Syntax

19.3.1 Exported Constants

None

19.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSpotLayout	drawStage, subregionImage, imgParams, beam (drawingObject)	doUpdate	-

19.4 Semantics

19.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **drawStage**: a reference to the drawing stage.
- **subregionImage**: a reference to the subregion image (imageDrawingObject).
- **beam**: a reference to the beam geometry.
- **imgParams**: a reference to function to call to get the rasterization grid parameters (number of rows **imageRows** and columns **imageCols**, and magnification (**imageMag**) provided by M4 through the Display Control M17).
- **doUpdate**: an optional reference to a function call when an update occurs.

19.4.2 Environment Variables and Assumptions

None

19.4.3 Access Routine Semantics

`drawSpotLayout(drawStage, subregionImage, imgParams, beam):`

- transition: On the given stage, draws a grid over the `subregionImage` with the `beam` geometry clone in the center of each cell in the drawn grid representing the individual location beam/spot sampling location and spot area coverage.
- output: a reference to the update function (`doUpdate`).

19.4.4 Local Functions

`doUpdate()`: Update the drawing based on the state variables.

20 MIS of Sample Subregion Visualization

20.1 Module

drawSampledSubregion (M14)

20.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

20.3 Syntax

20.3.1 Exported Constants

None

20.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSampledSubregion	drawStage, subregionImage, imgParams, beam (drawingObject)	doUpdate	-

20.4 Semantics

20.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **drawStage**: a reference to the drawing stage.
- **subregionImage**: a reference to the subregion image (imageDrawingObject).
- **beam**: a reference to the beam geometry.
- **imgParams**: a reference to function to call to get the rasterization grid parameters (number of rows **imageRows** and columns **imageCols**, and magnification (**imageMag**) provided by M4 through the Display Control M17).
- **doUpdate**: an optional reference to a function call when an update occurs.

20.4.2 Environment Variables and Assumptions

None

20.4.3 Access Routine Semantics

`drawSampledSubregion(drawStage, subregionImage, imgParams, beam):`

- **transition:** On the given stage, draws the `subregionImage` with “stenciled” or “clipped” the `beam` geometry clone at the center of each cell of the rasterization grid. This display represents the image content to be sampled by the beam/spot at discrete location and the area covered by the beam.
- **output:** a reference to the update function (`doUpdate`).

20.4.4 Local Functions

`doUpdate()`: Update the drawing based on the state variables.

21 MIS of Resulting Subregion Visualization

21.1 Module

drawResultingSubregion (M15)

21.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

21.3 Syntax

21.3.1 Exported Constants

None

21.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawResultingSubregion	stage, subregionRect, gtImage, imgParams, beam (drawingObject)	doUpdate	-

21.4 Semantics

21.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to the drawing stage.
- **subregionRect**: a reference to the subregion bounds.
- **gtImage**: a reference to the ground truth image (imageDrawingObject).
- **imgParams**: a reference to function to call to get the rasterization grid parameters (number of rows **imageRows** and columns **imageCols**, and magnification (**imageMag**) provided by M4 through the Display Control M17).
- **beam**: a reference to the beam geometry (drawingObject).
- **doUpdate**: an optional reference to a function call when an update occurs.

21.4.2 Environment Variables and Assumptions

None

21.4.3 Access Routine Semantics

`drawResultingSubregion(stage, subregionRect, gtImage, imgParams, beam):`

- **transition:** On the given stage (`stage`), draws the resampled subregion (using `gtImage` cropped to the bounds of `subregionRect`) meaning each cell in the rasterization grid (as defined by `imgParams`) is filled with the corresponding computed average pixel value using the “stenciled” or “clipped” image content with the `beam` geometry at the center of each cell as represented by the Sampled Subregion display (M14).
- **output:** a reference to the update function (`doUpdate`).

21.4.4 Local Functions

`doUpdate()`: Update the drawing based on the state variables.

22 MIS of Result Image Visualization

22.1 Module

drawResultingImage (M16)

22.2 Uses

Rendering Module (M20) and DrawingStage Module (M19)

22.3 Syntax

22.3.1 Exported Constants

None

22.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawResultingImage	stage, gtImage, subregionRect, imgParams	beam, updateConfigValues	-

22.4 Semantics

22.4.1 State Variables

These are kept for mutation, update calls, and performance reasons.

- **stage**: a reference to the drawing stage.
- **beam**: a reference to the beam geometry (**drawingObject**).
- **gtImage**: a reference to the ground truth image (**imageDrawingObject**).
- **subregionRect**: a reference to the subregion bounds.
- **imgParams**: a reference to function to call to get the rasterization grid parameters (number of rows **imageRows** and columns **imageCols**, and magnification (**imageMag**) provided by M4 through the Display Control M17).

22.4.2 Environment Variables

None

22.4.3 Assumptions

The `subregionRect` is smaller than the full extent of the ground truth image (`gtImage`).

22.4.4 Access Routine Semantics

`drawResultingImage(stage, beam, gtImage, subregionRect, imgParams):`

- transition: On the given stage (`stage`), continuously draws (row by row for responsiveness and performance) the resampled full image (`gtImage`) based on the beam shape (`beam`: Spot Profile M5), the rasterization grid as defined by `imgParams` for `subregionRect` (similar to Resulting Subregion M15) but extended to the full extent of the ground truth image, keeping the same relative cell size (meaning more cells - or rows and columns - that are “smaller” in the full image).
- output: returns a reference to `updateConfigValues` to call (by the Display Control M17) when there is a change in the magnification (`imageMag`), the rasterization grid (`imageRows` and `imageCols`) or the spot profile (beam shape, M5).

22.4.5 Local Functions

`updateConfigValues()`: Update the drawing based on the state variables.

References

- [1] J. de Fourestier. Module guide for imgbeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/Design/SoftArchitecture/MG.pdf.
- [2] J. de Fourestier. Software requirements specification for imgbeamer: Scanning electron microscope image formation, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/SRS/SRS.pdf.
- [3] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- [4] Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.
- [5] Anton Lavrenov. Konva.js - JavaScript 2d canvas library, December 2021. URL <https://konvajs.org/index.html>.
- [6] MDN. ImageData - Web APIs | MDN, February 2023. URL <https://developer.mozilla.org/en-US/docs/Web/API/ImageData>.
- [7] W3C. HTML living standard, the canvas element, Mar 2023. URL <https://html.spec.whatwg.org/multipage/canvas.html>.