

# Module Interface Specification for ImgBeamer

Joachim de Fourestier

March 19, 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

## 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

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

8.4	Semantics . . . . .	8
8.4.1	State Variables . . . . .	8
8.4.2	Environment Variables . . . . .	8
8.4.3	Assumptions . . . . .	8
8.4.4	Access Routine Semantics . . . . .	8
8.4.5	Local Functions . . . . .	9
<b>9</b>	<b>MIS of Image Export</b>	<b>10</b>
9.1	Module . . . . .	10
9.2	Uses . . . . .	10
9.3	Syntax . . . . .	10
9.3.1	Exported Constants . . . . .	10
9.3.2	Exported Access Programs . . . . .	10
9.4	Semantics . . . . .	10
9.4.1	State Variables . . . . .	10
9.4.2	Environment Variables . . . . .	10
9.4.3	Assumptions . . . . .	10
9.4.4	Access Routine Semantics . . . . .	10
9.4.5	Local Functions . . . . .	11
<b>10</b>	<b>MIS of Display Control</b>	<b>12</b>
10.1	Module . . . . .	12
10.2	Uses . . . . .	12
10.3	Syntax . . . . .	12
10.3.1	Exported Constants . . . . .	12
10.3.2	Exported Access Programs . . . . .	12
10.4	Semantics . . . . .	12
10.4.1	State Variables . . . . .	12
10.4.2	Environment Variables . . . . .	13
10.4.3	Assumptions . . . . .	13
10.4.4	Access Routine Semantics . . . . .	13
10.4.5	Local Functions . . . . .	13
<b>11</b>	<b>MIS of Drawing Stage / Canvas</b>	<b>14</b>
11.1	Module . . . . .	14
11.2	Uses . . . . .	14
11.3	Syntax . . . . .	14
11.3.1	Exported Constants . . . . .	14
11.3.2	Exported Access Programs . . . . .	14
11.4	Semantics . . . . .	14
11.4.1	State Variables . . . . .	14
11.4.2	Environment Variables . . . . .	15
11.4.3	Assumptions . . . . .	15

11.4.4	Access Routine Semantics . . . . .	15
<b>12</b>	<b>MIS of Image Rendering</b>	<b>16</b>
12.1	Module . . . . .	16
12.2	Uses . . . . .	16
12.3	Syntax . . . . .	16
12.3.1	Exported Constants . . . . .	16
12.3.2	Exported Access Programs . . . . .	16
12.4	Semantics . . . . .	16
12.4.1	State Variables . . . . .	16
12.4.2	Environment Variables . . . . .	16
12.4.3	Assumptions . . . . .	16
12.4.4	Access Routine Semantics . . . . .	17
12.4.5	Local Functions . . . . .	17
<b>13</b>	<b>Appendix</b>	<b>19</b>

### 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](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
character	char	a single symbol or digit
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 $(0, \infty)$
imageData [6]	$\mathbb{I}_{w,h}$	<b>data:</b> 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 <b>width:</b> $\mathbb{Z}_+$ width of $w$ and <b>height:</b> $\mathbb{Z}_+$ height of $h$ .

The specification of ImgBeamer uses some derived data types: sequences, strings, and tuples. 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. 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	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 as long the SRS value constrains 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 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  $\mathbb{R}$ .

#### 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 Module (M17) and obtains an update function reference for redraws or state changes. [I am not sure what transition means, couldn't find it as a defined term in the slides. I hope this is right... —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

None.

## 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, baseImage, resultImage, imageMag	-	DifferentImageSizes

### 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.

- exception: `DifferentImageSizes` meaning the size of `baseImage` and `resultImage` do not match.

#### **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, outputPath	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

None

## 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
<code>Init</code>	Display controls for 2 to 10 in section 10.2	a reference to the <code>doUpdate</code> for each	-
<code>doUpdateAll</code>	-	-	-

### 10.4 Semantics

#### 10.4.1 State Variables

References to drawing stages/canvases for all the visualization/display modules mentioned in section 10.2, along with each corresponding update function references.

### 10.4.2 Environment Variables

Display controls as provided by the GUI Module (M18).

### 10.4.3 Assumptions

None

### 10.4.4 Access Routine Semantics

`Init()`:

- `transition`: Initializes the drawing stages/canvases in each of the Display control (locations) 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, height	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 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

None

#### 12.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawGrid	gridLayer, rows, lineColor	rect, cell size (width/height cols, in pixels)	badGridParams
repeatDrawOnGrid	layer, rect, rows, cols, shape	-	badGridParams
ComputeProbeValue_gs	image, probe	grayscale value (U)	-
get_avg_pixel_gs	rawImageData	grayscale value (U)	-

### 12.4 Semantics

#### 12.4.1 State Variables

None.

#### 12.4.2 Environment Variables

The graphics processing hardware.

#### 12.4.3 Assumptions

None

#### 12.4.4 Access Routine Semantics

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

- 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 image object 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

## References

- [1] J. de Fourestier. Module guide for imgbeamer, 2023. URL [https://github.com/joedf/CAS741\\_w23/blob/main/docs/Design/SoftArchitecture/MG.pdf](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](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>.



## 13 Appendix

[Extra information if required —SS]