Verification and Validation Report: ImgBeamer

Joachim de Fourestier

April 6, 2023

1 Revision History

Date	Version	Notes
2023/03/26	0.1.0	Creation
2023/04/02	0.1.1	Start requirements testing sections
2023/04/03	0.2.0	Fill in the Functional Requirements Evaluation section, along with test images
2023/04/04	0.3.0	Complete NFR testing sections
	0.3.1	Start unit testing section
2023/04/05	0.4.0	Complete first revision
	0.4.1	Minor tweaks
	0.4.2	Use references for code libraries
	0.4.3	Added rank graphs
2023/04/06	0.4.4	Added another change/remark from testing

2 Symbols, Abbreviations and Acronyms

symbol	description
GUI	Graphical User Interface
L	Library
MG	Module Guide
MIS	Module Interface Specification
NaN	Not a Number
NFR	Non Functional Requirement
R	Requirement
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

See the SRS [3], VnV Plan [4], MG [1], and MIS [2] Documentation for additional items.

Contents

List of Tables

List of Figures

3 Report Purpose

This purpose of this report is to document the tasks accomplished and testing results as part the verification and validation process of ImgBeamer as laid out in the VnV Plan [4]. The code documentation along with notes on developer setup and testing is available at: https://joedf.github.io/ImgBeamer/jsdocs/index.html The software design documentation is available at: https://github.com/joedf/CAS741_w23. The source code is available at: https://github.com/joedf/ImgBeamer/tree/cas741

4 Functional Requirements Evaluation

In this section, we report the measures that were taken to evaluate whether the functional requirements (as listed in the SRS [3]) were met. Most of these tests were executed manually due to the complex nature of visual information, GUI testing, and the subjectivity of image quality.

4.1 Image Import and Export (R1 and R6)

This section is focused on testing the image import and export functionalities.

4.1.1 T1: Test import for PNG/JPG/BMP format (R1)

The software could successfully load or import valid (non-corrupt) images files in the PNG, JPG, and BMP formats. An example is shown in figure 1.

4.1.2 T2: Test PNG Image Export (R6)

The software could successfully export valid PNG image files of the resulting image (see figure 2).

4.2 Spot Profile and Imaging Parameters (R2, R3, R4, and R5)

This section focuses on testing the sampling and image rendering based on the given imaging parameters and subregion / ROI. The ground truth / input image used for these tests is depicted in figure 3 with the cyan overlay depicting the subregion area.

4.2.1 T3: Spot Width and Height - Exact-sampling (R2 and R5)

The test was passed as shown in figure 4.

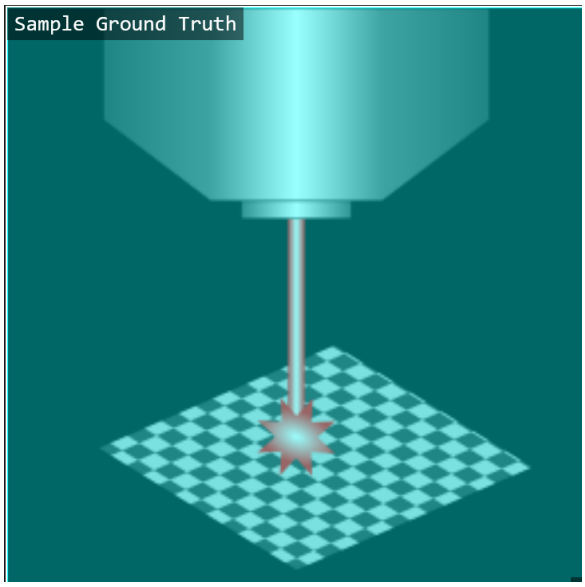

Figure 1: An example loaded test image.

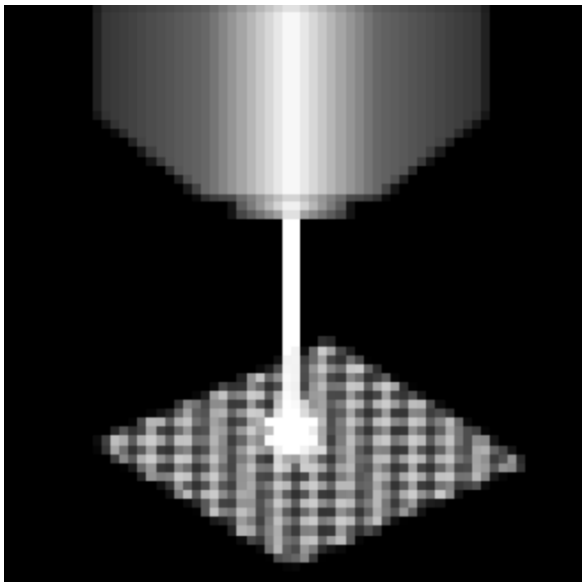

Figure 2: The image was faithfully reproduced (in grayscale with some imaging parameters change for clarity).

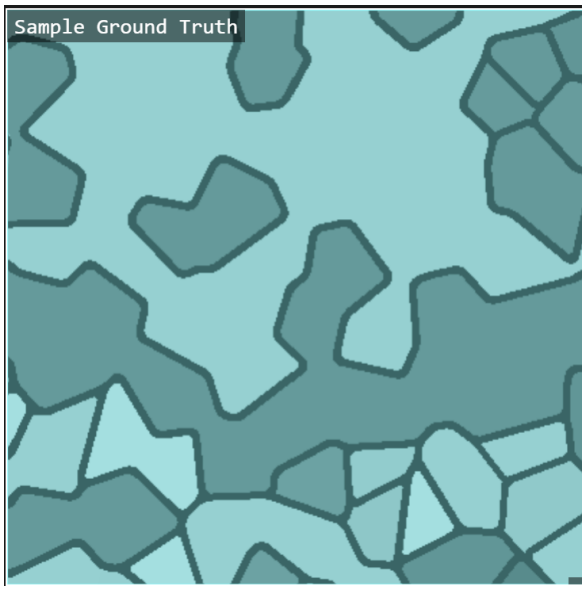

Figure 3: The ground truth image used for testing with the subregion set to cover the entire image.

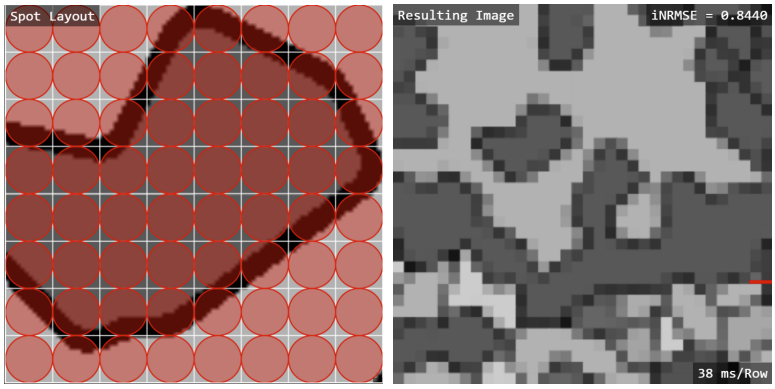

Figure 4: The spot layout and resulting image matching the expected test result: Spot size of 100 by 100%.

4.2.2 T4: Spot Width and Height - Under-sampling (R2 and R5)

The test was passed as shown in figure 5.

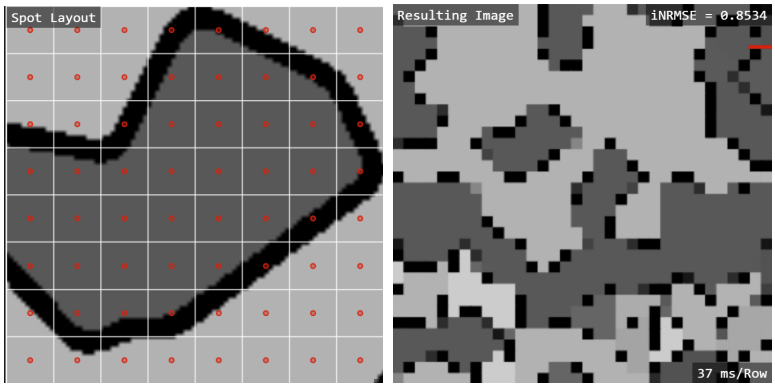

Figure 5: The spot layout and resulting image matching the expected test result: Spot size of 10 by 10%.

4.2.3 T5: Spot Width and Height - Over-sampling (R2 and R5)

The test was passed as shown in figure 6.

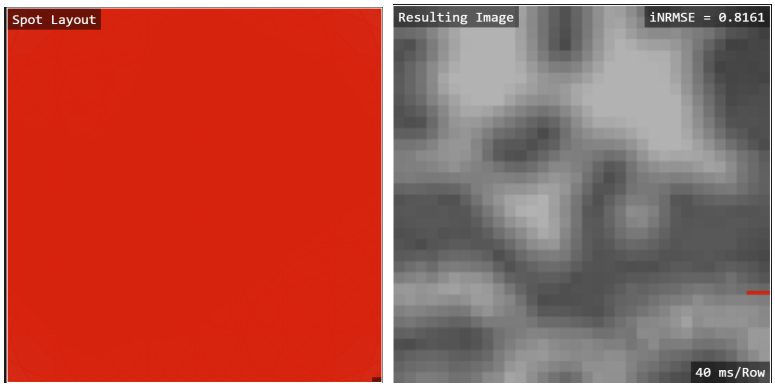

Figure 6: The spot layout and resulting image matching the expected test result: Spot size of 500 by 500%.

4.2.4 T6: Spot Rotation - Astigmatism (R2 and R5)

The test was passed as shown in figure 7.

4.2.5 T7: Raster Grid / Pixel Size (R3 and R5)

The test was passed as shown in figure 8.

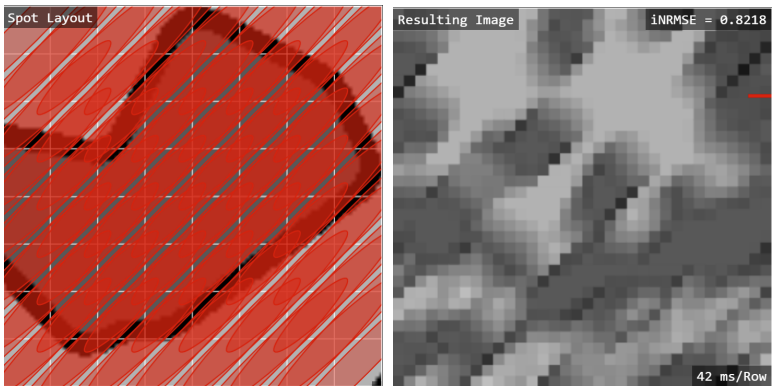

Figure 7: The spot layout and resulting image matching the expected test result: Spot size of 60 by 500% at 45 degrees rotation.

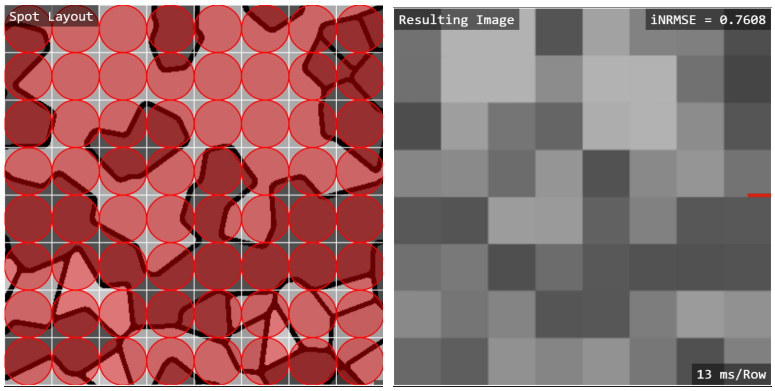

Figure 8: The spot layout and resulting image matching the expected test result: an 8 by 8 pixel image.

4.2.6 T8: Subregion / ROI (R4 and R5)

The test was passed as shown in figure 9.

4.2.7 T9: Ground Truth Reproduction (R1, R2, R3, and R6)

The test was passed as shown in figure 10.

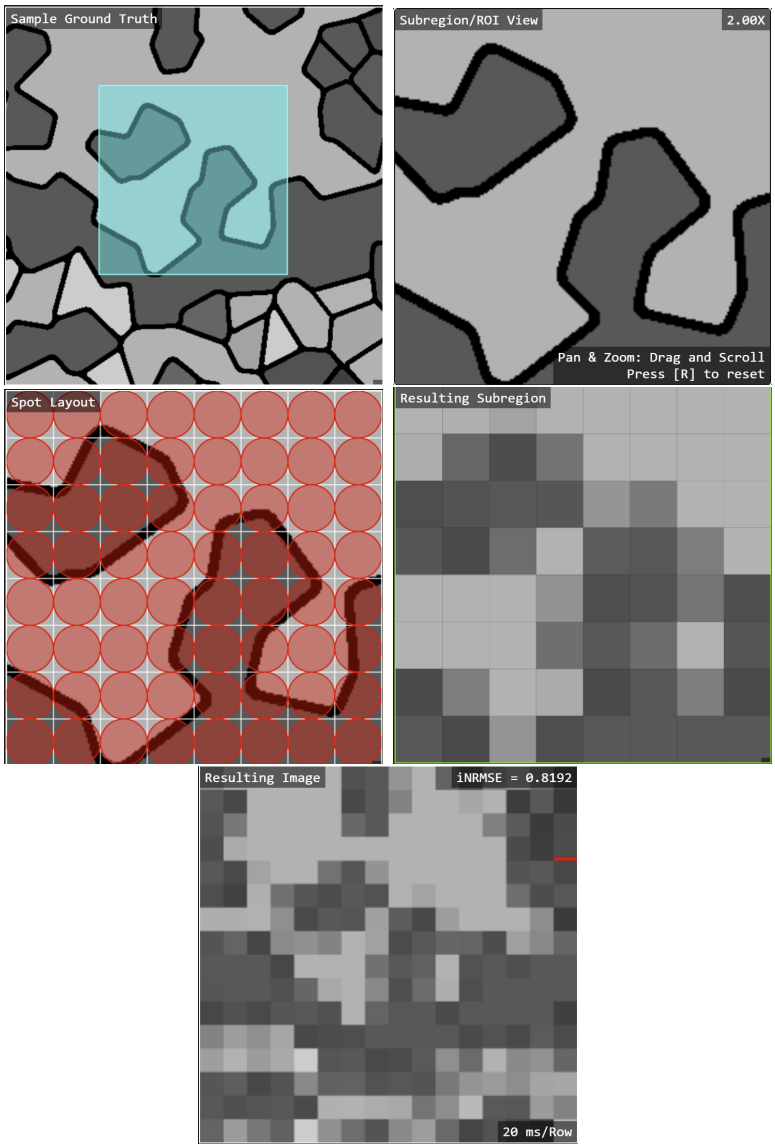

Figure 9: The spot layout, subregion, and resulting image matching the expected test result.

4.3 Image Quality Metric (R7)

This section focuses on testing the image quality metric general cases. Naturally, this is no flawless or foolproof image quality metric. Over 20 different image metrics have been reviewed and compared by Jagalingam and Hegde in a 2015 paper, each with their different strengths and weaknesses [6].

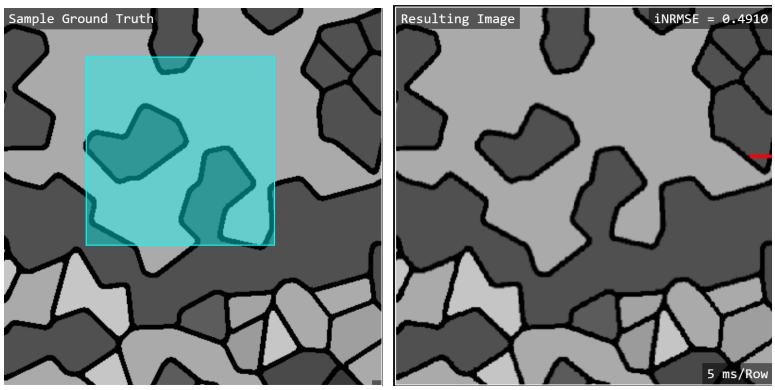

Figure 10: The expected test result: the ground image and resulting image are visually identical.

4.3.1 T10: High metric value (approximate to exact-sampling)

The test was passed as shown in figure 11.

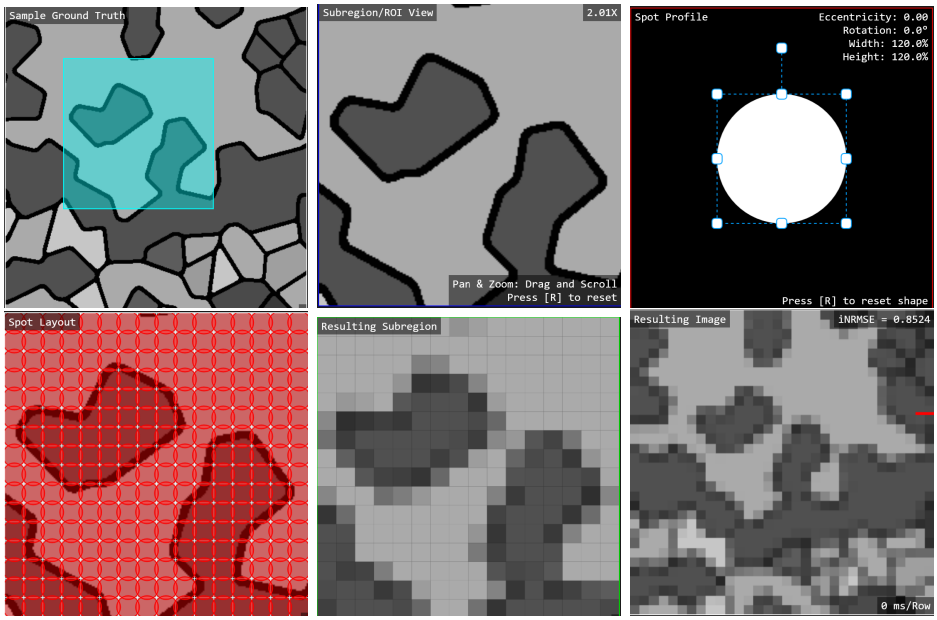

Figure 11: The spot layout, spot profile, subregion, and resulting image with a score equal or greater to the expected test result of a “0.8501” minimum.

4.3.2 T11: Low metric value (under-sampling)

The test was passed as shown in figure 12.

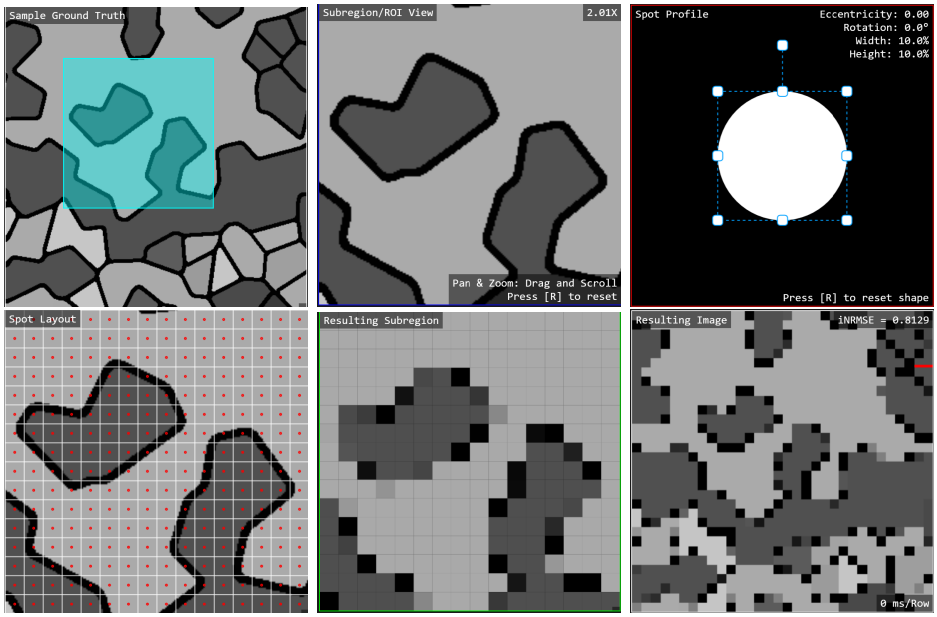

Figure 12: The spot layout, spot profile, subregion, and resulting image with a score less or equal to the expected test result of a “0.8501” maximum.

4.3.3 T12: Low metric value (over-sampling)

The test was passed as shown in figure 14.

4.3.4 T13: Control metric value

The test was passed as shown in figure 14.

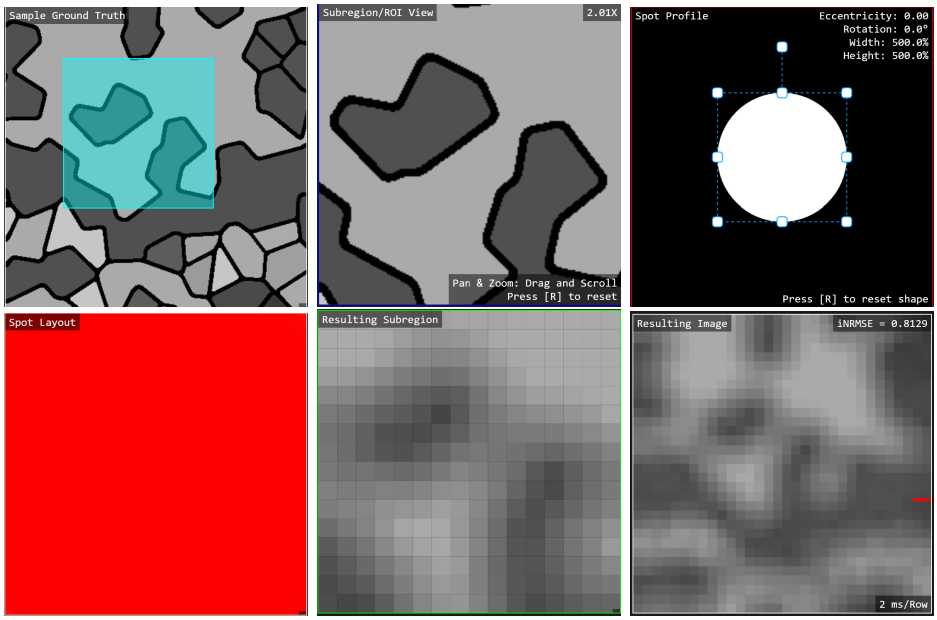

Figure 13: The spot layout, spot profile, subregion, and resulting image with a score less or equal to the expected test result of a “0.8501” maximum.

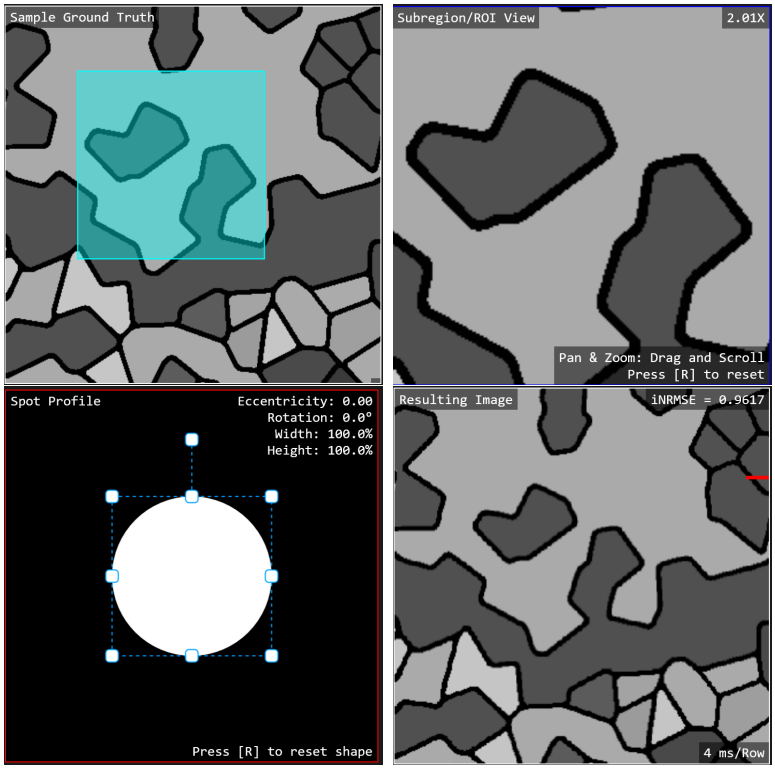

Figure 14: The spot layout, spot profile, subregion, and resulting image with a score greater or equal to the expected test result of a “0.9500” minimum.

5 Nonfunctional Requirements Evaluation

This section focuses on the testing results verifying whether the nonfunctional requirements (as defined in the SRS [3]) are met. Emphasis is put on the *usability* and by extension *portability*: The software should be easy to set up and use without having to worry about any technicalities that could otherwise hinder or completely prohibit non "tech-savvy" individuals from using the software.

5.1 Usability

5.1.1 T14: Usability Survey (NFR2)

The usability survey (see VnV Plan [4]) was not completed. The software was casually reviewed in verbal discussion with the expert consultants (as listed in the VnV Plan [4]) throughout the development of the software. Suggested features and minor issues has been implemented into the software such as: the ability to set a spot size by numeric input, draw the resulting row-by-row for responsiveness, and a display for spot shape eccentricity.

5.2 Accuracy

5.2.1 T15: Image Metric Survey (NFR1)

The image quality survey (see VnV Plan [4]) was not executed, but shall be some time in the future.

5.3 Maintainability

5.3.1 T16: Static code analysis (NFR3)

ESList was used to analyze the JavaScript code, and flake8 to analyze the Python code (only the test code involves Python). ESLint reported 186 errors and 20 warnings. Flake8 reported 10 issues. All of which have been resolved. The code linters now return zero issues.

5.3.2 T17: Code Review (NFR3)

A code review was executed by the author, but should be done by another individual some time in the future. Nearly all code that did not respect the code checklist (from the VnV Plan [4]) was rectified. In a few cases, some code has been marked for change with comments prefixed with `TODO:`. Otherwise, all functions except for a few non-design-dependent utility functions have been documented with JsDoc style

comments. As a result, code documentation has been generated and is published at: https://joedf.github.io/ImgBeamer/jsdocs/index.html

5.4 Portability

5.4.1 T18: Various Platforms and Environments (NFR4)

This test was executed by the author and was also indirectly tested successfully a number of times by the expert consultants [4]. Users simply need to navigate to the following link in any modern web browser to run and test the software: https://joedf.github.io/ImgBeamer/app/index.html

6 Comparison to Existing Implementation

This section is not applicable as an existing implementation for public use was not found or is not available for comparison.

7 Unit Testing

As stated, unit testing was limited to image metrics (as explained in the VnV Plan [4]). Unit tests for the image metrics are available under the `tests/image_metrics` folder.

Some statistics on the unit tests:

- A total of 247 tests (13 metric variants with 19 images comparisons each) were executed in 931 ms for the image metrics implemented in JavaScript.
- A total of 152 tests (8 metric variants with 19 images comparisons each) were executed in 38.3125 s for the image metrics implemented in Python.

7.1 Testing Setup

The following are general instructions (repeated here for convenience to the reader) to run the tests (each in their own subfolder).

- `js-tests/`: for javascript implementations
  - open the `index.html` page (using a local web server as described above) and look in the webconsole.
  - Optionally, an online hosted version exists here.

  - mainly in the webconsole, you can use `run_all(fant)` where if `fant` is true, all the image comparison tests will be run using the "fant-sampled" image [5] as the ground truth. Otherwise (false), it will use the "original" image as the ground truth instead.

- `py-tests/`: for python implementations
  - Install Python v3.10.6 or better (has not been tested on other versions)
  - You'll likely need to run `pip install sewar` **once** to get the required image metrics module/library.
  - run `imgquality.py`

7.2 Results for Functional Requirements

7.2.1 Image Metric Control Testing

All these tests passed within the margin of error.

7.3 Results for Nonfunctional Requirements

The tests are mostly experimental in nature help determine if there is a more fitting metric to use. Failures here do mean a failure for the software. Failures found here may represent a misunderstanding by the author or a limitation of the metrics.

7.3.1 Image Metric Control Testing

Please refer to figures 15 and 16 for the results.

- All metrics passed U10 variants U1 to U3, except for "psnr (darosh)" (from L4) and "mssim (sewar)" (from L1) which both failed U10 (variant U3).
- Nearly all metrics failed U10 variants U4 and U5, except for "(i)nrmse" (from L2) and "mssim (darosh)" (from L6). That said, they all scored consistently (about equal). The MSE-based metric that failed were due to a different normalization step.
- Nearly all metrics failed U10 variants U6 and U7, except for the metric used by ImgBeamer (2). MSE-based metrics all scored consistently. The SSIM-based and UQI metrics were not consistent at all, and seems to representing a limitation, design flaw, or an inappropriate test (likely was designed for different image types).

- For U10 variants U8, same as above.
- Other than the first two u10 test variants (U1 and U2), the SCC and VIFP metrics scores were all failures: all 0 or NaN (invalid, calculation could not be completed).

7.3.2 Image Metric Experimental Testing

The values (figures 15 and 16) were sorted into rank orders which be can viewer in figures 17, 18, 19, and 20. None of the metrics fit the expected rank order perfectly. That said, The MSE-based metrics, followed closely by UIQ then SSIM-based, fit closely; the discrepancies were all based on values that were within $\pm$ 0.02 of each other. These are otherwise considered relatively good fits. All other metrics did not fit the expected order at all.

#	testname	Expected Rank	ssim (obartra)	ssim (darosh)	mse (jdf)	psnr (jdf)	rmse (jdf)	nrmse (jdf)	inrmse (jdf)	nmse (jdf)	inmse (jdf)	mse (darosh)	psnr (darosh)	msssim (darosh)	ssim (darosh msssim)
1	og v og	1	1.0000	1.0000	0.0000	Infinity	0.0000	0.0000	1.0000	0.0000	1.0000	0.0000	Infinity	1.0000	1.0000
3	og v mDef	2	0.9947	0.9950	1.2946	47.0094	1.1378	0.0045	0.9955	0.0000	1.0000	0.1819	55.5336	0.9956	0.9950
2	og v fant	3	0.6219	0.6314	1136.7175	17.5743	33.7152	0.1322	0.8678	0.0175	0.9825	159.6693	26.0986	0.6953	0.6314
7	og v c120	4	0.6048	0.6145	1143.9479	17.5467	33.8223	0.1326	0.8674	0.0176	0.9824	160.6742	26.0713	0.6904	0.6145
8	og v c130	5	0.5869	0.5970	1166.8326	17.4607	34.1589	0.1340	0.8660	0.0179	0.9821	163.8791	25.9856	0.6754	0.5970
6	og v c100	6	0.6321	0.6418	1142.2711	17.5531	33.7975	0.1325	0.8675	0.0176	0.9824	160.4544	26.0773	0.7017	0.6418
5	og v c060	7	0.6612	0.6720	1267.5625	17.1011	35.6028	0.1396	0.8604	0.0195	0.9805	178.0619	25.6251	0.6872	0.6720
4	og v c010	8	0.6571	0.6667	1910.3629	15.3196	43.7077	0.1714	0.8286	0.0294	0.9706	268.3506	23.8438	0.5860	0.6667
9	og v c6a5	9	0.4257	0.4389	1736.9918	15.7328	41.6772	0.1634	0.8366	0.0267	0.9733	243.8960	24.2588	0.4698	0.4389
10	og v c500	10	0.3734	0.3849	2087.9348	14.9336	45.6939	0.1792	0.8208	0.0321	0.9679	293.1565	23.4598	0.3077	0.3849
	what is better	low	high	high	low	high	low	low	high	low	high	low	high	high	high

	Extra Tests	Expected Rank	ssim (obartra)	ssim (darosh)	mse (jdf)	psnr (jdf)	rmse (jdf)	nrmse (jdf)	inrmse (jdf)	nmse (jdf)	inmse (jdf)	mse (darosh)	psnr (darosh)	msssim (darosh)	ssim (darosh msssim)
11	og v blak	-	0.0102	0.0102	15824.9738	6.1374	125.7974	0.4933	0.5067	0.2434	0.7566	2222.9502	14.6615	0.0001	0.0102
12	og v wite	-	0.4042	0.4107	24510.4191	4.2373	156.5580	0.6140	0.3860	0.3769	0.6231	3443.0036	12.7614	0.0034	0.4107
13	og v half	-	0.2016	0.2074	19105.8071	5.3191	138.2238	0.5421	0.4579	0.2938	0.7062	2683.8122	13.8433	0.0022	0.2074
	Control Tests														
14	blak v wite	-	0.0079	0.0079	64516.0000	0.0341	254.0000	0.9961	0.0039	0.9922	0.0078	9062.6283	8.5583	0.0075	0.0079
15	blak v half	-	0.4802	0.4781	32258.0000	3.0444	179.6051	0.7043	0.2957	0.4961	0.5039	4531.3141	11.5686	0.1831	0.4781
16	wite v half	-	0.4804	0.4776	32258.0000	3.0444	179.6051	0.7043	0.2957	0.4961	0.5039	4531.3141	11.5686	0.1847	0.4776
17	blak v gray	-	0.0174	0.0174	13689.0000	6.7671	117.0000	0.4588	0.5412	0.2105	0.7895	1922.9078	15.2912	0.0167	0.0174
18	wite v gray	-	0.7623	0.7623	18769.0000	5.3964	137.0000	0.5373	0.4627	0.2886	0.7114	2636.5006	13.9205	0.7538	0.7623
19	half v gray	-	0.3716	0.3696	16229.0000	6.0279	127.3931	0.4996	0.5004	0.2496	0.7504	2279.7042	14.5520	0.1414	0.3696

Execution time		
(obartra) ssim_test	244	ms
darosh_ssim_test	65	ms
(jdf) NRMSE_test	5	ms
darosh mse_test	13	ms
darosh_msssim_test	604	ms
total time (ms)	931	
total tests	247	

Figure 15: The results from the JavaScript tests. As indicated in the VnV Plan [4]: "ssim (obartra)" refers to L3, "ssim (darosh)" refers to L5, columns with "jdf" are metrics implemented by the author as L2, "mse and psnr (darosh)" refer to L4, and "msssim and ssim (darosh)" refer to L6.

#	testname	Expected rank	mse	rmse	psnr	uqi	ssim	msssim	scc	vifp
1	og v og	1	0.0000	0.0000	inf	1.0000	1.0000	1.0000	0.4161	1.0000
3	og v mDef	2	1.4586	1.2077	46.4915	0.9999	0.9948	0.9985	0.4158	0.9994
2	og v fant	3	1447.8925	38.0512	16.5234	0.8951	0.5964	0.7487	0.0000	0.0629
7	og v c120	4	1457.3952	38.1758	16.4950	0.8942	0.5793	0.7360	-0.0009	0.0605
8	og v c130	5	1485.0299	38.5361	16.4135	0.8925	0.5624	0.7214	-0.0009	0.0579
6	og v c100	6	1453.2852	38.1220	16.5073	0.8949	0.6069	0.7553	0.0000	0.0638
5	og v c060	7	1606.2846	40.0785	16.0726	0.8852	0.6402	0.7668	-0.0003	0.0639
4	og v c010	8	2326.2763	48.2315	14.4642	0.8459	0.6516	0.7233	0.0006	0.0564
9	og v c6a5	9	2112.4697	45.9616	14.8829	0.8660	0.4375	0.5201	0.0016	0.0313
10	og v c500	10	2491.8421	49.9184	14.1656	0.8499	0.4109	0.3949	0.0018	0.0209
	what is better		low	low	high	high	high	high	high	high
	Extra tests									
11	og v blak		17994.7421	134.1445	5.5793	0.0001	0.0003	0.0845	0.0000	0.0000
12	og v wite		23094.8339	151.9698	4.4957	0.5343	0.4556	0.2207	0.0000	0.0000
13	og v half		19453.8052	139.4769	5.2408	0.2834	0.2232	0.1556	0.0002	0.0013
	Control tests									
14	blak v wite		65025.0000	255.0000	0.0000	0.0000	0.0001	0.2930	0.0000	nan
15	blak v half		32512.5000	180.3122	3.0103	0.4859	0.4798	0.6784	0.0000	nan
16	wite v half		32512.5000	180.3122	3.0103	0.5003	0.4800	0.6812	0.0000	nan
17	blak v gray		16384.0000	128.0000	5.9866	0.0000	0.0004	0.3520	0.0000	nan
18	wite v gray		16129.0000	127.0000	6.0547	0.6430	0.8019	0.9710	0.0000	nan
19	half v gray		16256.5000	127.5010	6.0205	0.3280	0.3851	0.6622	0.0000	0.0000
	Total time (s)	38.3125								
	Total tests	152								

Figure 16: The results from the Python tests.

javascript tests													
Expected Rank	SSIM based metrics		MSE based metrics									SSIM based metrics	
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	6	6	3	3	3	3	3	3	3	3	3	4	6
4	7	7	5	5	5	5	5	5	5	5	5	5	7
5	8	8	6	6	6	6	6	6	6	6	6	7	8
6	5	5	4	4	4	4	4	4	4	4	4	3	5
7	3	3	7	7	7	7	7	7	7	7	7	6	3
8	4	4	9	9	9	9	9	9	9	9	9	8	4
9	9	9	8	8	8	8	8	8	8	8	8	9	9
10	10	10	10	10	10	10	10	10	10	10	10	10	10

Figure 17: The ranked results from the JavaScript tests.

Python (sewar) tests								
Expected rank	mse	rmse	psnr	uqi	ssim	ms-ssim	scc	vifp
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	6	5	7	5
4	5	5	5	5	7	6	10	6
5	6	6	6	6	8	8	9	7
6	4	4	4	4	5	4	6	4
7	7	7	7	7	4	3	8	3
8	9	9	9	10	3	7	5	8
9	8	8	8	8	9	9	4	9
10	10	10	10	9	10	10	3	10

Figure 18: The ranked results from the Python tests.

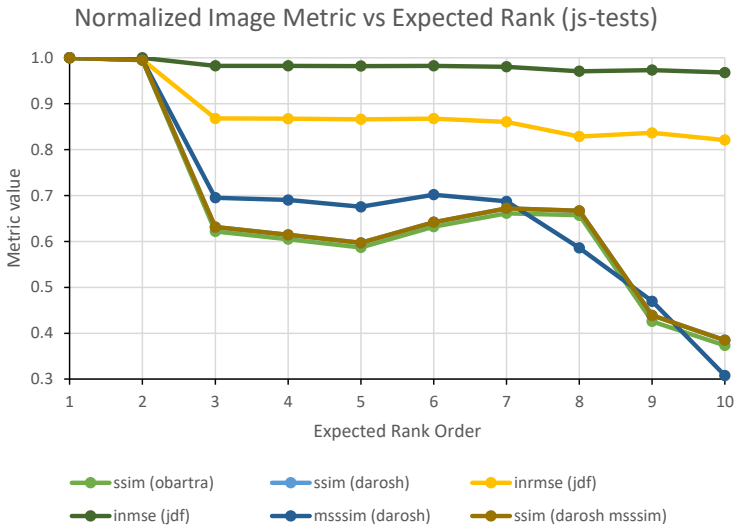

Figure 19: A graph of the JavaScript test results of the normalized image metrics (from L2, L3, L5 and L6) versus the expected rank order.

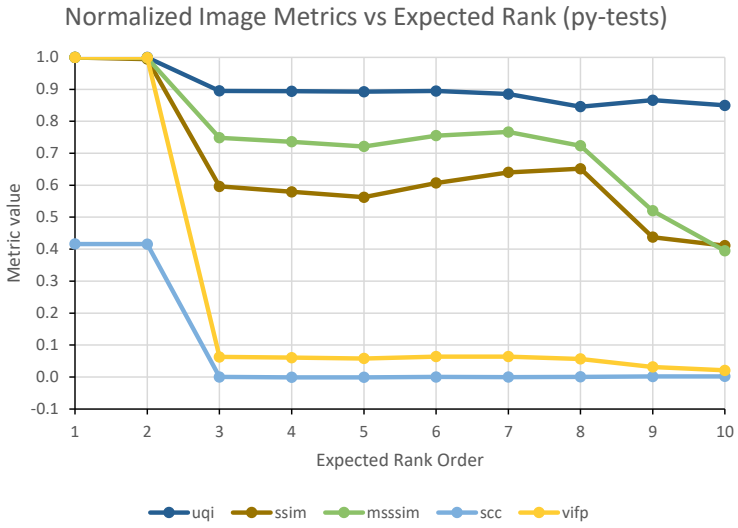

Figure 20: A graph of the Python test results of the normalized image metrics (from L1) versus the expected rank order.

8 Remarks and Changes Due to Testing

1. The passing value for T13 (see 4.3.4) had to be changed from "0.9800" to '0.9500' since the resulting images look identical. The resulting value of '0.9617' seemed satisfactory and the original expected test value was perhaps too constrained.

2. Through manual testing (mainly section 4.3) and watching for runtime warnings and errors, a small discrepancy was found in the sizes of the compared images with the image metrics: one image was 1-2 pixels taller depending on the size of the web browser window. This was due to sub-pixel position calculations of the canvas when the UI is first drawn. This has been rectified by ensuring all values are rounded up to the nearest integers.

3. Manual testing (mainly section 4.3) also revealed a limitation in the MSE-based metrics (as used by ImgBeamer): two images with very similar average gray values overall will erroneously score a "good" score (e.g., $\geq 80\%$). This is a design flaw or limitation likely documented in literature (but did not check), stemming from the use of averaging.

4. For similar reasons, the error margin for the unit tests was adjusted from $\pm$ 2% to $\pm$ 5%.

5. Code is cleaner/clearer and better commentated/documented after running code linting and doing code review. Many unnecessary global variables have been eliminated.

6. Switching over to the UQI/UIQ or SSIM metrics may be worth considering.

7. Noticed from unit testing that some image metrics in javascript (js-tests) were somewhat inconsistent with their python counterpart (py-tests) because of inaccurate pixel values being read. For example, black pixels (0x000000) where read as (0x010101) or gray pixels (0x808080) being read as (0x767676)... As a result, the output metric values were very liekly off because of this. An issue has been opened on the GitHub code repository to explore any possible mitigations.

9 Automated Testing

Currently, this is no automated execution of the tests every time there is change. Automated testing using pytest and QUnit to display and calculate the test pass percentages and execution times would be helpful and is desired (perhaps sometime in the future), but have not been set up at this time due to a lack of time / resources.

That said, the tests are easy to run (just executing two files) and could be set up with continuous integration some time in the future.

10 Trace to Requirements

In Table 1, the connections between the test sections and the requirements are shown.

Section	R1	R2	R3	R4	R5	R6	R7	NFR1	NFR2	NFR3	NFR4
4.1	X					X					
4.2		X	X	X	X	X					
4.3							X				
7							X				
5.2								X			
5.1									X		
5.3										X	
5.4											X

Table 1: Traceability Matrix Showing the Connections Between the Test Sections and the Requirements

11 Trace to Modules

In Tables 2 and 3, the connections between the test sections and the modules are shown.

Section	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
4.1	X		X			X					X	X
4.2	X			X	X			X	X	X	X	X
4.3							X					
7							X					

Table 2: Traceability Matrix Showing the Connections Between the Test Sections and the Modules (Part 1)

Sections	M13	M14	M15	M16	M17	M18	M19	M20	M21
4.1		X	X	X	X	X	X	X	
4.2	X	X	X	X	X	X	X	X	
4.3						X	X	X	X
7						X	X	X	X

Table 3: Traceability Matrix Showing the Connections Between the Test Sections and the Modules (Part 2)

12 Code Coverage Metrics

Code coverage was used as part of this project. However, callgraphs have been generated using Code2Flow to verify the flow application with respect to the design (as per the MG [1] and MIS [2]). An up-to-date but simplified (with limited search depth) callgraph is shown in figure 21. An older (somewhat out-dated) callgraph but with greater detail is shown in figure 22.

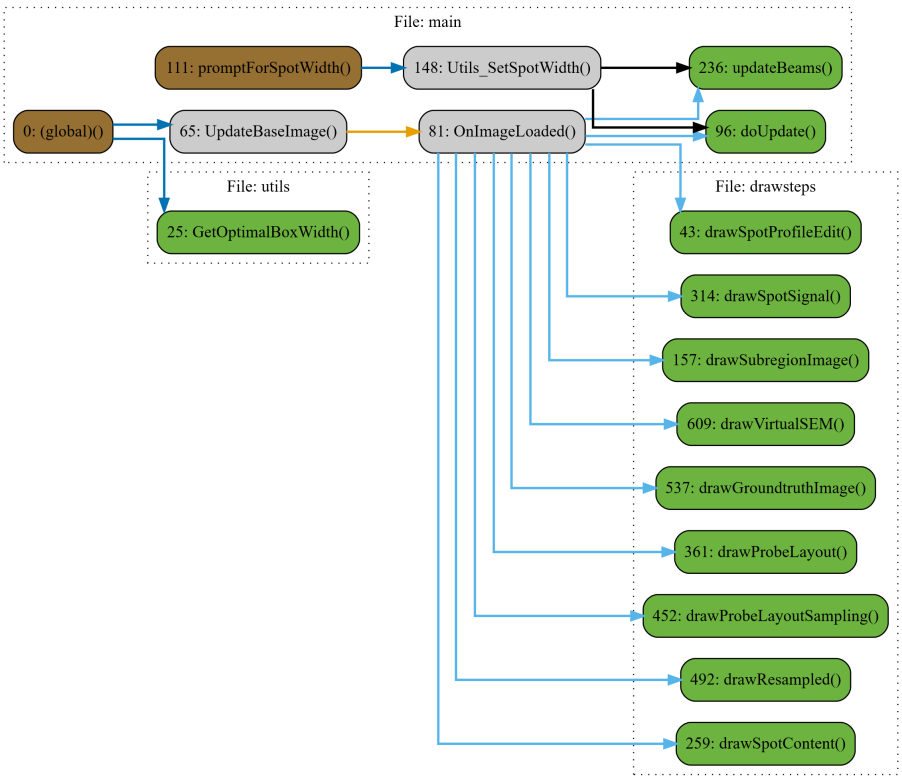

Figure 21: An up-to-date callgraph generated using Code2Flow

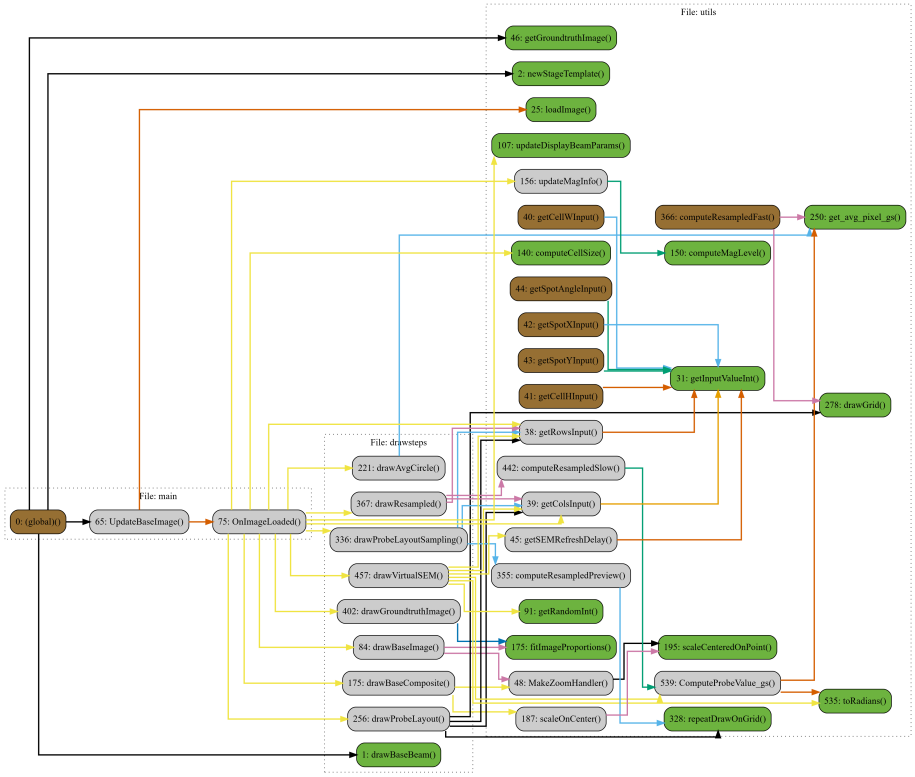

Figure 22: A somewhat out-dated but more detailed callgraph generated using Code2Flow

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. Module interface specification for ImgBeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/Design/SoftDetailedDes/MIS.pdf.

[3] 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.

[4] J. de Fourestier. Verification and validation plan for ImgBeamer, 2023. URL https://github.com/joedf/CAS741_w23/blob/main/docs/VnVPlan/VnVPlan.pdf.

[5] Karl M. Fant. A Nonaliasing, Real-Time Spatial Transform Technique. *IEEE Computer Graphics and Applications*, 6(1):71–80, January 1986. ISSN 1558-1756. doi: 10.1109/MCG.1986.276613. Conference Name: IEEE Computer Graphics and Applications.

[6] P. Jagalingam and Arkal Vittal Hegde. A review of quality metrics for fused image. *Aquatic Procedia*, 4:133–142, 2015. ISSN 2214-241X. doi: https://doi.org/10.1016/j.aqpro.2015.02.019. URL https://www.sciencedirect.com/science/article/pii/S2214241X15000206.