



# **Test Report**

University of Maryland Global Campus  
SWEN 670 – Team B  
Spring Semester  
Version 1.0

April 1, 2023

| Version | Issue Date | Changes                |
|---------|------------|------------------------|
| 0.1     | 3/27/2023  | Initial Version        |
| 1.0     | 4/01/2023  | Milestone 4 Submission |

|  |                                     |
|--|-------------------------------------|
| 1. Executive Summary.....                                    | 4                                   |
| 2. Introduction .....  | 4                                   |
| 2.1 Purpose .....  | 4                                   |
| 2.2 Scope .....  | 4                                   |
| 3. Testing .....   | 4                                   |
| 3.1 Test Strategy.....                                       | 4                                   |
| 3.2 Tests Conducted.....                                     | 5                                   |
| 4. Requirements Traceability Matrix.....                     | 6                                   |
| 5. Functional Tests Execution Findings .....                 | 7                                   |
| 5.1 CORE: UPLOAD IMAGES TO THE SERVER.....                   | 9                                   |
| 5.2 CORE: PANORAMIC IMAGE CREATED .....                      | 11                                  |
| 5.3 CORE: DETERMINE TRANSITION HOTSPOTS.....                 | 13                                  |
| 5.4 EDITING: EXTRACT TEXT FROM IMAGES .....                  | 15                                  |
| 5.5 EDITING: DETECT AND BLUR HUMAN FACES .....               | 20                                  |
| 5.6 EDITING: APPLY GLOW FILTER .....                         | 23                                  |
| 5.7 EDITING: ADD/EDIT/DELETE INFORMATIONAL HOTSPOT .....     | 26                                  |
| 5.8 SEARCH: SEARCH TEXT .....                                | <b>Error! Bookmark not defined.</b> |
| 6. Test Results and Summary .....                            | 28                                  |
| 6.1 Defect Summary (Fail tests).....                         | 28                                  |
| 6.1.1 Test Case 5.2.2 – Panoramic Image .....                | 28                                  |
| 6.1.2 Test Case 5.3.1 – Neighbor Detection Limitations ..... | 28                                  |
| 6.1.3 Test Case 5.6.1 – Glow Filter Limitations.....         | 30                                  |
| 6.1.4 Test Case 5.4.1 – Text Extraction Limitations .....    | 30                                  |

## **1. Executive Summary**

This report presents the results of testing performed on Team B's contribution to the ViroTour application for the Spring 2023 UMGC Software Engineering Capstone course. The tests were conducted to ensure the software meets the specified requirements and to identify and document any limitations, defects, or issues.

The testing approach was a combination of automated and manual methods, covering all the main functional areas of the back-end of ViroTour, and includes both positive and negative results. The results of the testing indicate that the software features delivered according to the specified requirements, however, not all features were implemented, and there are known limitations that need further enhancement. These limitations are documented in 6.1 Defect Summary.

## **2. Introduction**

### **2.1 Purpose**

The purpose of this test report document is to provide a summary of the testing activities performed for Team B's efforts to develop the server-side of the application, ViroTour. It is to communicate the results of the testing process to stakeholders, including the developers, our front-end counterpart, Team A, project managers, and the client.

This document will provide an overview of the testing activities, demonstrate the quality of the software, communicate the status of the project, provide recommendations on how to improve the software's quality, identify any issues that need to be addressed, and serve as a record of tests performed and their results which can be used for future reference and to support decision making.

This Test Report documents the results of the ViroTour tests results as of 04/01/2023.

### **2.2 Scope**

The tests in this document are for the server-side features of the ViroTour application and includes testing of functional and non-functional features for the core functions, editing functions, and search functions. The types of tests conducted and our approach is outlined in section 3.1, Testing Strategy.

What is not considered in scope for Team B's testing purposes includes, penetration testing, system testing, performance testing, security testing, and testing of any front-end features for ViroTour.

## **3. Testing**

### **3.1 Test Strategy**

The purpose of testing is to make Sure It Works! There are many ways to achieve that, but it depends on the goals of the project and the expectations from the stakeholders.

The strategy is to build an automated suite of test cases which will help in the development of the software. ViroTour's Server Processing Team strives to use testing as a tool to build modular, maintainable, resilient, and fault-tolerant code.

For this project, we will develop unit tests, component tests, integration tests, and manual tests, finally, we will perform User Acceptance Tests (UAT). These are sessions with the client where the engineering team will demonstrate the working product. The product owner and project manager will collect feedback from the client during these sessions and incorporate it into subsequent development efforts.

### 3.2 Tests Conducted

| Feature Tested                | Description   | Status          |
|-------------------------------|---|-----------------|
| Upload Images to the Server   | The process of uploading photos from the client system to the program server is tested.   | Pass            |
| Panoramic Image Created       | The process of creating panoramic images based on the user input images will be evaluated.  | Pass            |
| Determine Transition Hotspots | The process of determining transition hotspots in a virtual tour will be evaluated.   | Pass            |
| Extract Text from Images      | This test will verify that the text from every hotspot/location object (that can be viewed in a panorama) is extracted from the to-be-stored object.    | Pass            |
| Detect and Blur Human Faces   | The capability of applying a blurring effect to anonymize individuals' faces that appear in the panoramas will be tested.                               | Pass            |
| Apply Filters                 | This test will verify that the user can apply a filter to an existing panoramic image, and the resulting image be saved on the server under a new name. | Pass            |
| Add Informational Hotspot     | This test will verify that the user can add a hotspot to a virtual tour   | Not Implemented |
| Edit Informational Hotspot    | This test will verify that data for the informational hotspot was updated within the database   | Not Implemented |

|                                 |  |                 |
|---------------------------------|--|-----------------|
| Delete Informational Hotspot    | This test will verify that the user can delete a hotspot   | Not Implemented |
| Search Text                     | This test will verify that the user can search images by providing a search text.                                    | Pass            |
| Multiple Tours                  | This test will verify that the application can support multiple virtual tours.                                       | Non-Functional  |
| Image Processing Volume Testing | This test will verify that the application can process approximately 60 images within one hour during tour creation. | Non-Functional  |
| Image Stitching Performance     | This test will verify that the application can stitch a minimum of four images within 10 seconds.                    | Non-Functional  |

**Table 3.2 – Descriptions about All Covered Test Cases.**

#### **4. Requirements Traceability Matrix**

| <b>Test ID</b>                   | <b>Test Description</b>       | <b>SRS Requirement #</b> | <b>Requirement Description</b>        | <b>Status</b> |
|----------------------------------|-------------------------------|--------------------------|---------------------------------------|---------------|
| 5.1.1                            | Upload single image file      | 3.1.1.1                  | Core: Upload Images to the server     | Pass          |
| 5.2.1                            | Panoramic image created       | 3.1.1.2                  | Core: Panoramic image creation        | Pass          |
| 5.3.1<br>5.3.2                   | Determine transition hotspots | 3.1.1.3                  | Core: Determine transitional hotspots | Pass          |
| 5.4.1<br>5.4.2<br>5.4.3<br>5.4.4 | Extract text from images      | 3.1.2.1                  | Editing: Extract text from images     | Pass          |
| 5.5.1<br>5.5.2<br>5.5.3          | Detect and blur human faces   | 3.1.2.2                  | Editing: Detect and blur human faces  | Pass          |

|       |                              |         |                         |                         |
|-------|------------------------------|---------|-------------------------|-------------------------|
| 5.6.1 | Apply glow filter            | 3.1.2.3 | Editing: Apply filters  | Pass                    |
| 5.7.1 | Add informational hotspot    | 3.1.2.4 | Editing: Add hotspot    | Feature not implemented |
| 5.7.2 | Edit informational hotspot   | 3.1.2.5 | Editing: Edit hotspot   | Feature not implemented |
| 5.7.3 | Delete informational hotspot | 3.1.2.6 | Editing: Delete hotspot | Feature not implemented |
| 5.8.1 | Search text                  | 3.1.3.1 | Search: Search text     | Feature not implemented |

**Table 4 – Correlation between Different Tests and Requirements, Including the Current Results of Each Test.**

## **5. Functional Tests Execution Findings**

This is a view of the ViroTour tests cases executing programmatically. Notice that some tests have smaller parts that are tested and that our longest test takes about a minute to execute.

|   |   |               |
|---|---|---------------|
| ✓ | compute                                   | 5 min 3 sec   |
| ✓ | test_compute_tour                         | 3 min 59 sec  |
| ✓ | test_compute_tour                         | 47 sec 752 ms |
| ✓ | test_compute_tour_full                    | 3 min 11 sec  |
| ✓ | test_image_blur_faces                     | 1 sec 603 ms  |
| ✓ | test_image_blur_faces                     | 1 sec 471 ms  |
| ✓ | test_getPathToDetectorFile                | 0 ms          |
| ✓ | test_grayscaleImage                       | 2 ms          |
| ✓ | test_detectFaces                          | 130 ms        |
| ✓ | test_image_extract_text                   | 24 sec 823 ms |
| ✓ | test_extract_text_from_image_with_text    | 21 sec 311 ms |
| ✓ | test_extract_text_from_image_with_no_text | 3 sec 512 ms  |
| ✓ | test_image_filter                         | 6 sec 674 ms  |
| ✓ | test_detect_brightness                    | 11 ms         |
| ✓ | test_adjust_contrast_brightness           | 25 ms         |
| ✓ | test_adjust_hue_saturation_value          | 45 ms         |
| ✓ | test_apply_gaussian_filter                | 41 ms         |
| ✓ | test_apply_glow_effect                    | 6 sec 552 ms  |
| ✓ | test_panoramic                            | 31 sec 47 ms  |
| ✓ | test_compute_panoramic                    | 896 ms        |
| ✓ | test_compute_panoramic_full               | 30 sec 151 ms |
| ✓ | images                                    | 378 ms        |
| ✓ | test_add_images                           | 378 ms        |
| ✓ | test_add_images_to_tour                   | 108 ms        |
| ✓ | test_add_images_multiple_locations        | 270 ms        |
| ✓ | tour                                      | 160 ms        |
| > | test_hello                                | 1 ms          |
| > | test_tour_add                             | 27 ms         |
| > | test_tour_delete                          | 41 ms         |

**Figure 5 – Unit Tests of ViroTour Features.**




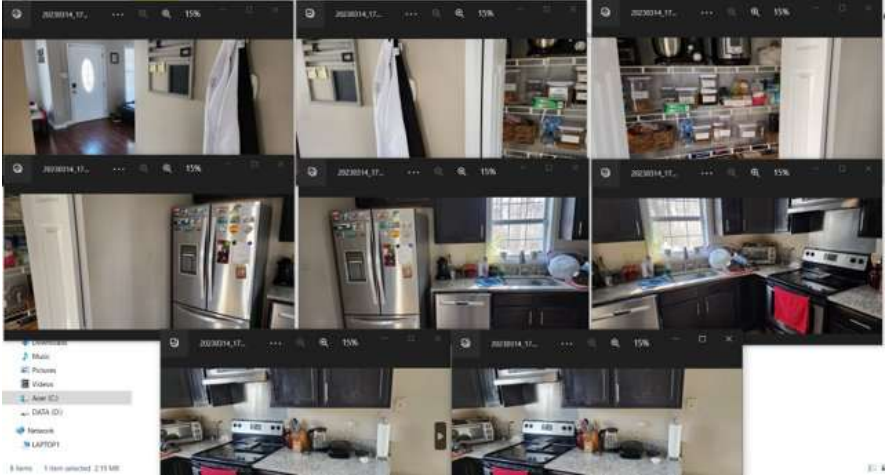

## 5.1 CORE: UPLOAD IMAGES TO THE SERVER

|                      |  |
|----------------------|--|
| Test Number          | 5.1.1  |
| Feature Being Tested | Upload single image file   |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. Initialize the test by setting up the necessary testing environment.</li><li>2. Create a tour by calling the "add_tour" function with the tour name and description as arguments.</li><li>3. Retrieve the image paths for a given location using the "get_image_paths" function.</li><li>4. The returned data should contain the correct tour ID and server file paths as expected.</li><li>5. The raw images for the locations should be retrieved successfully.</li><li>6. Assert that returned data contains the correct image count and server file path by comparing them to the expected values.</li></ol>  |
| Expected Results     | <p>The expected result of the test case is that it should verify the functionality of adding images to a tour. The test should confirm that the tour can be created successfully and that the images can be uploaded to the tour without any issues. Additionally, the test should validate that the returned data contains the expected tour ID and server file paths.</p> <p>Furthermore, the test should verify the functionality of retrieving the raw images for a tour location. It should confirm that the raw images can be retrieved successfully and that the returned data contains the expected image count and server file paths. The test should pass without any errors or exceptions, indicating that the functionality for adding and retrieving images to a tour is working as expected.</p> |
| Assumptions          | <p>The assumptions for the test are that the client variable is properly initialized and configured to communicate with the server, and that the functions being tested (add_tour, get_image_paths, upload_images, and get_raw_images) are functioning correctly. Additionally, it is assumed that the input images for the test case exist and are correctly formatted in the expected directory, and that the test is being executed in a controlled testing environment rather than in a production environment. Finally, it is assumed that the</p>  |

|               |  |
|---------------|--|
|               | returned data from the functions being tested contains the expected information in the expected format.  |
| Actual Result | Tour name: "Tour 1"<br>Tour description: "Tour Description Example"<br>Image location:<br>Input_image/sample_tour/location1/20230314_171226.jpg<br>and Input_image/sample_tour2/location01/1.jpg |
| Pass/Fail     | Pass   |


## 5.2 CORE: PANORAMIC IMAGE CREATED

|                      |   |
|----------------------|---|
| Test Number          | 5.2.1   |
| Feature Being Tested | Panoramic Image Created   |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. User initializes a tour.</li><li>2. User uploads images.</li></ol>   |
| Expected Results     | The images are placed inside a directory and are then associated to a tour of a specific location. The set of images represents a single location within the tour. The images within the location folder will be stitched together as a single panoramic image. |
| Assumptions          | The user has initialized a tour.  |
| Actual Result        |   |
| Pass/Fail            | Pass  |

|                      |  |
|----------------------|--|
| Test Number          | 5.2.2  |
| Feature Being Tested | Panoramic Image Created  |
| Test Process/Steps   |  |
| Expected Results     | The images are correctly stitched.   |
| Assumptions          | The user has initialized a tour.   |
| Actual Result        |  |
| Pass/Fail            | Fail   |

### 5.3 CORE: DETERMINE TRANSITION HOTSPOTS

|                      |  |
|----------------------|--|
| Test Number          | 5.3.1  |
| Feature Being Tested | Determine transition hotspots  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. User initializes a tour.</li><li>2. User uploads panoramic image list. (input example: ["st_1.jpg", "st_2.jpg", "st_3.jpg", "st_4.jpg", "st_5.jpg"])(<i>Figure 5.3.1</i>)</li><li>3. The algorithm use image list as input to detect transition hotspots.</li></ol>   |
| Expected Results     | For each image, the algorithm will compare it with its adjacent images and output a corresponding hotspot coordinates. This hotspot contains x-axis and y-axis coordinate and let user interact with the generated tours.  |
| Assumptions          | The user has completed the initialization of a tour, and the route of this tour has been determined to be linear. In order for the algorithm to operate correctly, the user needs to input the linear tour into the system in the order specified. In other words, the user must provide the path information of the tour in the predetermined order for the algorithm to process. |
| Actual Result        | The transition hotspot for each input image [[(4683, 1791)], [(4683, 1791)], [(4683, 1791)], [(4767, 2118)], [(4740, 2082)]]   |
| Pass/Fail            | Pass   |

|                      |  |
|----------------------|--|
| Test Number          | 5.3.2  |
| Feature Being Tested | Determine transition hotspots  |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>2. User initializes a tour.</li> <li>4. User uploads panoramic image list.</li> <li>5. The algorithm use image list as input to detect transition hotspots.</li> </ol>  |
| Expected Results     | For each image, the algorithm will compare it with its adjacent images and output a corresponding hotspot coordinates. This hotspot contains x-axis and y-axis coordinate and let user interact with the generated tours.  |
| Assumptions          | The user has completed the initialization of a tour, and the route of this tour has been determined to be linear. In order for the algorithm to operate correctly, the user needs to input the linear tour into the system in the order specified. In other words, the user must provide the path information of the tour in the predetermined order for the algorithm to process. |
| Actual Result        | <p>Blue Marking of Neighbor in Current Location.</p>   |
| Pass/Fail            | Pass   |

## 5.4 EDITING: EXTRACT TEXT FROM IMAGES

|                      |  |                     |                       |                            |
|----------------------|--|---------------------|-----------------------|----------------------------|
| Test Number          | 5.4.1  |                     |                       |                            |
| Feature Being Tested | Extract Text from Images   |                     |                       |                            |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>1. The user creates a new tour or wants to add a new location to a tour.</li> <li>2. The user selects the images they want to compute a panoramic image from.</li> <li>3. The user clicks on compute tour or add location to tour.</li> <li>4. The panoramic image gets created and text is extracted, stored into the database.</li> </ol> <p>(Figure 4.4.1 below is used for this test case).</p> |                     |                       |                            |
| Expected Results     | Text from the computed panoramic image will be extracted and stored in the database (for searching text purposes).   |                     |                       |                            |
| Assumptions          | The panoramic image <b>has</b> text to extract.  |                     |                       |                            |
| Actual Result        | Extracted Text   | XY Location of Text | Computed Accuracy (%) | Is Text Visually Accurate? |
|                      | Mr: and Mrs  ! ectarg  | [2852, 1886]        | 2.66%                 | somewhat                   |
|                      | OME CENTER   | [5956, 1886]        | 99.98%                | yes                        |
|                      | WELCOME  | [7959, 1907]        | 99.84%                | yes                        |
|                      | Wekcome Center   | [2934, 2043]        | 92.89%                | somewhat                   |
|                      | 01 Arbus IMAX Theater  | [7959, 2077]        | 49.89%                | somewhat                   |
|                      | "edSbic 8  | [2958, 2114]        | 2.43%                 | no                         |
|                      | ~LOWER LEVEL   | [7959, 2152]        | 28.02%                | yes                        |
|                      | Airbus IMAX? Theater @1  | [2940, 2162]        | 29.08%                | yes                        |
|                      | :4 Space Hangar  | [7932, 2264]        | 20.29%                | yes                        |
|                      | 7 FRestoration Hanear  | [7942, 2302]        | 48.85%                | somewhat                   |
|                      | and Walkway  | [2954, 1792]        | 97.73%                | yes                        |
|                      | 0 Simulator Rides  | [2919, 1933]        | 48.24%                | yes                        |
|                      | The Comm   | [2802, 1959]        | 99.83%                | yes                        |
|                      | UPPER LEVEL  | [7959, 1966]        | 94.93%                | yes                        |
|                      | ~Baby Care Room  | [2931, 2219]        | 53.08%                | yes                        |
| Pass/Fail            | Pass   |                     |                       |                            |



**Figure 4.4.1 - Panoramic Image of Air and Space Museum with Text Extraction Overlay.**

|                      |  |                     |                       |                            |
|----------------------|--|---------------------|-----------------------|----------------------------|
| Test Number          | 5.4.2  |                     |                       |                            |
| Feature Being Tested | Extract Text from Images   |                     |                       |                            |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>1. The user creates a new tour or wants to add a new location to a tour.</li> <li>2. The user selects the images they want the panoramic image to be computed from.</li> <li>3. The user clicks on compute tour or add location to tour.</li> <li>4. The panoramic image gets created and text is extracted, stored into the database.</li> </ol> <p>(Figure 4.4.2 below is used for this test case).</p> |                     |                       |                            |
| Expected Results     | Text from the computed panoramic image will be extracted and stored in the database (for searching text purposes).   |                     |                       |                            |
| Assumptions          | The panoramic image <b>has</b> text to extract.  |                     |                       |                            |
| Actual Result        | Extracted Text   | XY Location of Text | Computed Accuracy (%) | Is Text Visually Accurate? |
|                      | 481  | [2672, 1704]        | 22.50%                | yes                        |
|                      | Ryan PT-22A Recrue   | [3126, 2620]        | 41.04%                | somewhat                   |
|                      | IIIA   | [4450, 2726]        | 82.37%                | yes                        |
|                      | IA   | [5616, 2736]        | 85.07%                | yes                        |
|                      | Westland [   | [4138, 2638]        | 46.20%                | somewhat                   |
|                      | Lysan  | [4274, 2682]        | 99.99%                | yes                        |
|                      | Blackbird  | [5754, 2727]        | 73.06%                | yes                        |
|                      | Lockl  | [5515, 2776]        | 66.81%                | yes                        |
| Pass/Fail            | Pass   |                     |                       |                            |





**Figure 4.4.2 - Panoramic Image of Air and Space Museum with Text Extraction Overlay.**

|                      |   |                     |                       |                            |
|----------------------|---|---------------------|-----------------------|----------------------------|
| Test Number          | 5.4.3   |                     |                       |                            |
| Feature Being Tested | Extract Text from Images  |                     |                       |                            |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>1. The user creates a new tour or wants to add a new location to a tour.</li> <li>2. The user selects the images they want the panoramic image to be computed from.</li> <li>3. The user clicks on compute tour or add location to tour.</li> <li>4. The panoramic image gets created and text is extracted, stored into the database.</li> </ol> <p><i>(Figure 4.4.3 below is used for this test case).</i></p> |                     |                       |                            |
| Expected Results     | Text from the computed panoramic image will be extracted and stored in the database (for searching text purposes).  |                     |                       |                            |
| Assumptions          | The panoramic image <b>has</b> text to extract.   |                     |                       |                            |
| Actual Result        | Extracted Text  | XY Location of Text | Computed Accuracy (%) | Is Text Visually Accurate? |
|                      | SONY  | [5495, 1250]        | 99.61%                | yes                        |
|                      | Zioez   | [5952, 1602]        | <b>4.84%</b>          | no                         |
|                      | H   | [6581, 1613]        | 48.06%                | no                         |
| Pass/Fail            | Pass  |                     |                       |                            |



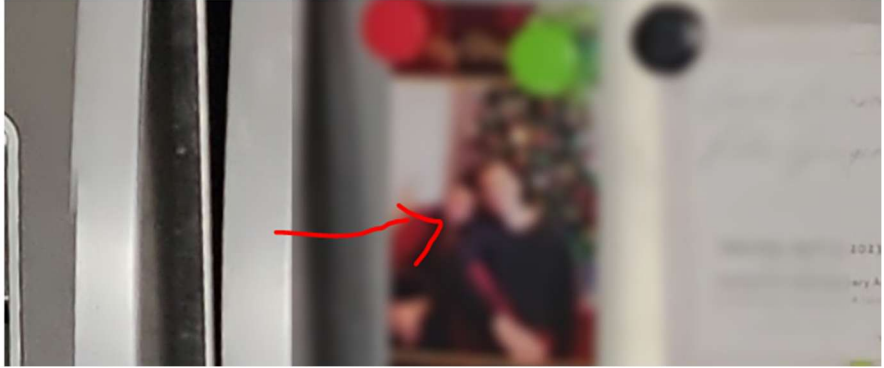
**Figure 4.4.3 - Panoramic Image Computed from ViroTour App with Text Extraction Overlay.**

|                      |  |                     |                       |                            |
|----------------------|--|---------------------|-----------------------|----------------------------|
| Test Number          | 5.4.4  |                     |                       |                            |
| Feature Being Tested | Extract Text from Images   |                     |                       |                            |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>1. The user creates a new tour or wants to add a new location to a tour.</li> <li>2. The user selects the images they want the panoramic image to be computed from.</li> <li>3. The user clicks on compute tour or add location to tour.</li> <li>4. The panoramic image gets created and because no text exists in the image, no text is extracted (and empty array object returned to database).</li> </ol> <p>(Figure 4.4.4 below is used for this test case).</p> |                     |                       |                            |
| Expected Results     | For the computed transitional hotspot/location, there will be no extracted text that is saved in the location object's text property.  |                     |                       |                            |
| Assumptions          | The panoramic image <b>does not have</b> text to extract.  |                     |                       |                            |
| Actual Result        | Extracted Text   | XY Location of Text | Computed Accuracy (%) | Is Text Visually Accurate? |
|                      | N/A  | N/A                 | N/A                   | N/A                        |
|                      | (No text objects were retrieved; therefore, the table above is empty).   |                     |                       |                            |
| Pass/Fail            | Pass   |                     |                       |                            |




**Figure 4.4.4 - Image That Returned No Extracted Text Objects from EasyOCR Library.**


## 5.5 EDITING: DETECT AND BLUR HUMAN FACES

|                      |  |
|----------------------|--|
| Test Number          | 5.5.1  |
| Feature Being Tested | Detect and Blur Human Faces  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. The user creates a tour from multiple images.</li><li>2. The user requests a panoramic image at a specified location.</li></ol>   |
| Expected Results     | The panoramic image and verifies that the human faces are blurred.   |
| Assumptions          | The panoramic image has a face to blur.  |
| Actual Result        | <p>The panoramic with the blurred faces is returned.</p> <p>► Get Image File After ✎</p> <div><div>GET ▼</div><div>http://127.0.0.1:8081/api/tour/images/panoramic-image-file/Virtual Tour 1/14</div></div> <div>Params Authorization Headers (9) Body Pre-request Script Tests Setting</div> <div>Body Cookies (1) Headers (11) Test Results</div> <div></div> |
| Pass/Fail            | Pass   |

|                      |  |
|----------------------|--|
| Test Number          | 5.5.2  |
| Feature Being Tested | Detect and Blur Human Faces  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. The user creates a tour from multiple images.</li><li>2. The user requests a panoramic image at a specified location.</li></ol> |
| Expected Results     | The panoramic image and verifies that the human faces are blurred.   |
| Assumptions          | The panoramic image has a face to blur.  |


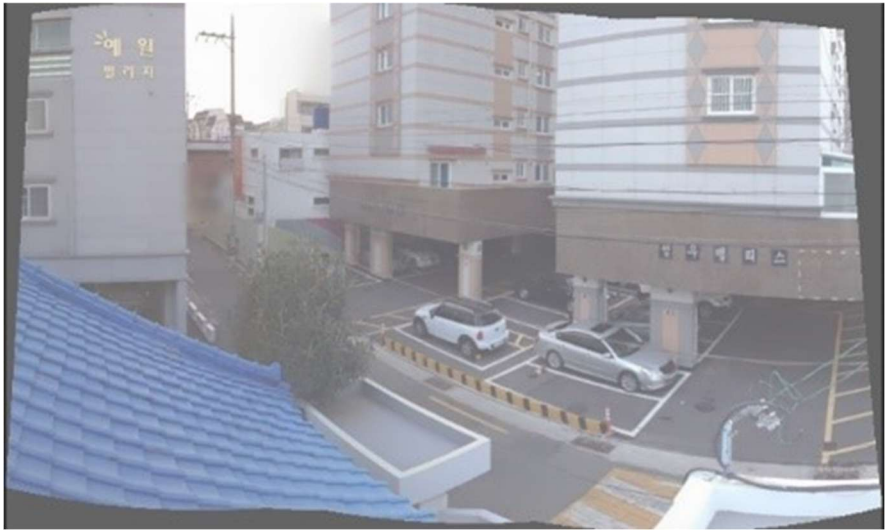
|               |  |
|---------------|--|
| Actual Result | <p>The panoramic with the blurred faces is returned.</p>  |
| Pass/Fail     | Pass   |

|                      |  |
|----------------------|--|
| Test Number          | 5.5.3  |
| Feature Being Tested | Detect and Blur Human Faces  |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>3. The user creates a tour from messi.png test image.</li> <li>4. The user requests a panoramic image at a specified location.</li> </ol> |
| Expected Results     | The panoramic image and verifies that the human faces are blurred.   |
| Assumptions          | The panoramic image has a face to blur.  |

|  |   |
|--|---|
| <p data-bbox="203 195 381 226">Actual Result</p> | <p data-bbox="537 195 1162 226">The panoramic with the blurred faces is returned.</p>  A photograph of a soccer match in progress. Two players are in the foreground: one in a white jersey with the number 10 and the Swiss flag, and another in a dark blue jersey with the number 10 and the Argentine flag. Both players' faces are blurred. A soccer ball is on the grass in front of them. The background shows a blurred crowd in a stadium. <p data-bbox="203 1058 321 1089">Pass/Fail</p> |
|  | <p data-bbox="537 1058 594 1089">Pass</p>   |



## 5.6 EDITING: APPLY GLOW FILTER

|                      |  |
|----------------------|--|
| Test Number          | 5.6.1  |
| Feature Being Tested | Apply glow filter  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. The user creates a tour from multiple images.</li><li>2. The user requests a panoramic image at a specified location.</li><li>3. The user sends a brightness value.</li></ol> |
| Expected Results     | A new panoramic image is returned with a glow filter applied based on brightness value.  |
| Assumptions          | A tour was created.  |
| Input                | <p>The panoramic image before processing</p>    |
| Actual Result        | <p>The panoramic image with the glow filter is returned.</p>   |
| Pass/Fail            | Pass   |

The following is a python test script to validate the functionality of the glow filter.

```
def test_apply_glow_effect(client):
    tour_name = "tour-1"
    brightness = 100
    location_id = 1
    add_tour(client, tour_name, "Tour Description Example")
    upload_images(client, tour_name, get_image_paths('input_images/location1'))
    compute_tour(client, tour_name)

    # Test get relative image path
    pano_image_path = get_panoramic_image(client, tour_name, 1)['server_file_path']
    assert pano_image_path == 'panoramic_images/T_1_L_1_pano_blurred.jpg'
    # Get absolute file path of panoramic image
    input_file = api_upload_resolve_path(pano_image_path).replace("\\", "/")
    # Set the output directory
    output_directory = get_image_path('input_images\\location4\\result').replace("\\", "/")
    # Set the absolute file path of processed panoramic image
    output_file = get_image_path('input_images\\location4\\result\\result.png').replace("\\", "/")
    # Clear all files from output directory
    clear_files_from_folder(get_image_path(output_directory))
    # Detect dark image
    before = detect_brightness(api_upload_resolve_path(pano_image_path))
    # Calls image_filter.adjust_contrast_brightness()
    file = adjust_contrast_brightness(input_file, brightness, output_file)
    # Validate that output file path was returned
    assert file == output_file
    # Detect brighter image
    after = detect_brightness(file)
    assert before != after

    # Save glow filter values
    apply_glow_effect(location_id, brightness)
    # Verify filter setting for glow effect was saved to the database
    location = db.session.query(Location).filter(Location.location_id == location_id).first()
    filter_ = db.session.query(Filter).filter(Filter.filter_id == location.filter_id).first()
    assert filter_.filter_name == 'glow'
    assert filter_.setting == brightness
```

**Figure 5.6.1 - Glow Effect Test Script.**

The glow filter produced the following images with setting the brightness value at 255, 100, -100 and -255. The glow filter produced expected results. When the brightness value was closest to the maximum number of 255, it produced an image closest to white; whereas, a brightness value closest to the minimum of -255 produced a black image.





original



Brightness value = 255



Brightness value = 100



Brightness value = -100



Brightness value = -255

**Figure 5.6.2 - Varying Brightness Value Output.**

## 5.7 EDITING: ADD/EDIT/DELETE INFORMATIONAL HOTSPOT

|                      |  |
|----------------------|--|
| Test Number          | 5.7.1  |
| Feature Being Tested | Add Informational Hotspot (NOT IMPLEMENTED)  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. The user enters a tour and is loaded into a location.</li><li>2. The user clicks “add informational hotspot.”</li><li>3. The user manually selects the location they want the hotspot to be at.</li><li>4. The user confirms the location.</li><li>5. The user adds text they want the hotspot to contain, and confirms it.</li><li>6. The user clicks add informational hotspot, and is added to the view.</li></ol> |
| Expected Results     | An informational hotspot will be clickable and viewable when users enter the tour.   |
| Assumptions          | There are no existing informational hotspots at the location the new one was created at.   |
| Actual Result        | N/A  |
| Pass/Fail            | NOT IMPLEMENTED  |

|                      |   |
|----------------------|---|
| Test Number          | 5.7.2   |
| Feature Being Tested | Edit Informational Hotspot (NOT IMPLEMENTED)  |
| Test Process/Steps   | <ol style="list-style-type: none"><li>1. The user enters a tour and is loaded into a location.</li><li>2. The user clicks on an existing informational hotspot.</li><li>3. The user can either manually move the hotspot, change the previous existing text, or both, clicking save to save their edits.</li><li>4. The informational hotspot is then updated to the new location and the text is changed to the new value.</li></ol> |
| Expected Results     | An already existing informational hotspot is updated in its location, content, or both.   |
| Assumptions          | The new location the user chooses to move the current informational hotspot is not on top of another informational hotspot (will not allow to save on top of each other).   |

|               |                 |
|---------------|-----------------|
| Actual Result | N/A             |
| Pass/Fail     | NOT IMPLEMENTED |

|                      |  |
|----------------------|--|
| Test Number          | 5.7.3  |
| Feature Being Tested | Delete Informational Hotspot (NOT IMPLEMENTED)   |
| Test Process/Steps   | <ol style="list-style-type: none"> <li>1. The user enters a tour and is loaded into a location.</li> <li>2. The user clicks on an existing informational hotspot.</li> <li>3. The user clicks “delete hotspot.”</li> <li>4. The user confirms their delete action to delete said hotspot.</li> <li>5. The hotspot is removed from the tour.</li> </ol> |
| Expected Results     | There is an already existing informational hotspot for a user to delete.   |
| Assumptions          | If the tour has none, this feature is not executable. There needs to be at least one informational hotspot available in the tour.  |
| Actual Result        | N/A  |
| Pass/Fail            | NOT IMPLEMENTED  |

## 6. Test Results and Summary

### 6.1 Defect Summary (Fail tests)

#### 6.1.1 Test Case 5.2.2 – Panoramic Image

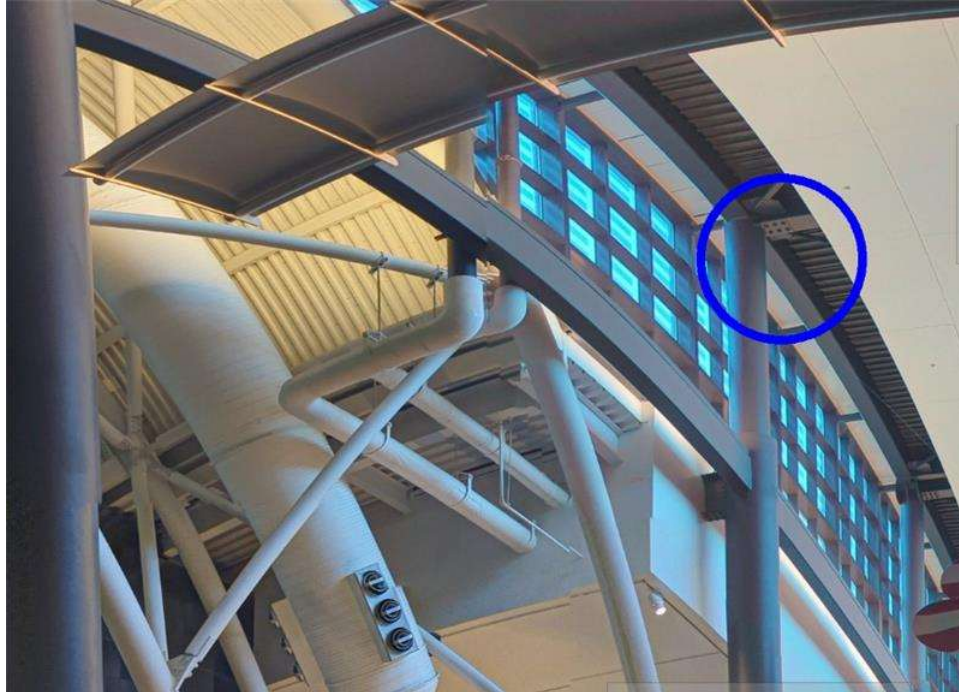
Not all of the panoramic images are stitched. Some of the images are not present in the result. This seems to be a limitation of the stitching algorithm provided by OpenCV. This causes data loss, since it fails to create the 360-degree image. One side effect of this is that the neighbor detection algorithm may not work correctly.

#### 6.1.2 Test Case 5.3.1 – Neighbor Detection Limitations

The limitations of Neighbor Detection can sometimes manifest in the following two situations:

Case 1: As shown in the figure, when performing hotspot detection, the algorithm identifies hotspot regions by recognizing the feature points in the image. However, in some cases, even though the feature points in two photos are similar, they do not necessarily represent the correct transition point.

For example, in the (*figure 6.1.2.1*) and (*6.1.2.2*), two adjacent hotspot regions may appear, and their feature points in both photos match the ceiling. Although the algorithm can recognize these feature points and they are similar in both photos, they do not represent the correct transition point. Therefore, this is one of the limitations of Neighbor Detection.



**Figure 6.1.2.1 – Blue Marking of Neighbor in Current Location.**



**Figure 6.1.2.2 – Transition Expected Via Red Arrow but Was Not Received.**

Case 2: In addition to the limitations of uneven distribution and insufficient number of feature points, the second limitation of Neighbor Detection is the error in OpenCV feature point matching. Sometimes, the information contained in the matched feature points is not sufficient to be a candidate for a hotspot.

For example, the image (*figure 6.1.2.3*) shows a photo of a kitchen cabinet. Although the algorithm is able to recognize some feature points on the cabinet, the information contained in these feature points is limited and cannot be used as a basis for determining hotspot areas. In this range, there is only one color, which can cause the algorithm to produce misjudgments.

To overcome this limitation, further optimization and filtering of feature point quality is required. This can be achieved by selecting appropriate feature point detection algorithms or using filtering techniques such as RANSAC and LMedS to reduce matching errors and improve matching accuracy. In addition, machine learning techniques such as deep learning can also effectively improve the results of hotspot area detection.



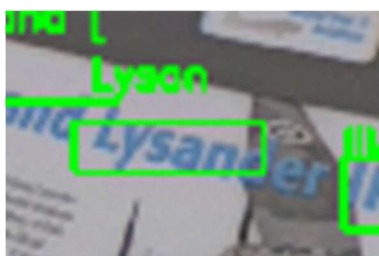
**Figure 6.1.2.3 – Blue Marking of Neighbor in Current Location.**

### 6.1.3 Test Case 5.6.1 – Glow Filter Limitations

At this time the glow filter limitations are that contrast and brightness is adjusted across the entire image and across all pixels. The glow filter currently does not target a specific area of interest on the image to apply a glow filter. The glow filter does not detect underexposed areas of the image for brightening. A possible suggestion for future implementation is to segment the image into smaller components so that each pixel can be evaluated and adjusted. To further enhance the glow filter, other methods to illuminate an image can be explored such as adjusting the hue, saturation and value or applying a Gaussian blur. Initial implementations of these are included in the code base but are not being used. Another limitation of the glow filter is as images are brightened; the quality of the image may reduce by inadvertently introducing noise.

### 6.1.4 Test Case 5.4.1 – Text Extraction Limitations

The contrast between the text and sudden shift in background color can confuse the text extraction to work properly, as displayed below.

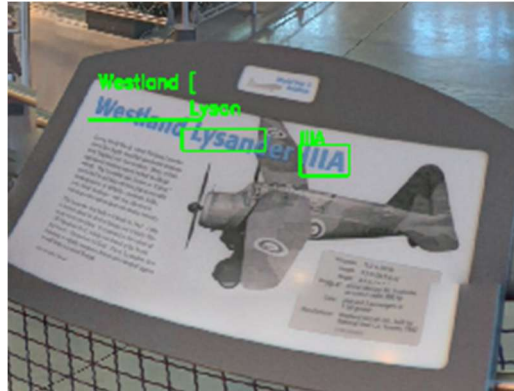


**Figure 6.1.4.1 - The Actual Text Extracted. Closeup.**



**Figure 6.1.4.2 - Original**

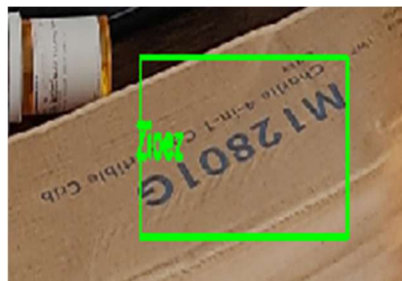
As a result, potential text (or the full context; "Lysander" instead of just "Lysan") can be lost. Text extraction can also fail to extract the small text, only extracting the headings in the image below:



**Figure 6.1.4.3 - Small Body Text Failed to Extract.**

A possible solution to this is to possibly edit the panoramic image before extracting text, e.g., adjusting the contrast or brightness of the image to have the text stand out more visually. However, this approach has not been tested, and we have not utilized our own glow filter with the text extraction to see any increase or decrease in the amount of extracted text objects.

Another issue noticed was text recognition can be faulty when considering text that is upside down, as displayed below:



**Figure 6.1.4.4 - Small Body Text Failed to Extract.**

The text in the image is upside down but the algorithm used by EasyOCR believes the lettering to be just enough to look like actual text, as if it was right-side up. According to the test case in section 5.4, the extracted object in the above image's content was computed as "Zioez," which from viewing the image is not what the text was supposed to be. There is a good cause to conduct further image processing or algorithm enhancement in order to limit the amount of false text extractions, although it does work well in many cases (refer to section 5.4 for results).