Lost Stars

G. H.
Course: Space Engineering 2024
Czech Technical University in Prague
Technická 2, Prague, Czech Republic
X

M. Š.
Course: Space Engineering 2024
Czech Technical University in Prague
Technická 2, Prague, Czech Republic
X

R. H.
Course: Space Engineering 2024
Czech Technical University in Prague
Technická 2, Prague, Czech Republic
X

***Abstract*—This project aims to conclude a possible solution suggestion for finding the abnormal phenomenon "lost stars" in images of fields in the sky. It entails the strategic method of finding lost stars using modern technology and software. The method is summarized by comparing old images against new ones, containing a field in the sky, and looking for differences in brightness of stars. By doing so, it can be determined if a star has been lost. From implementing this on real images of the sky, it was concluded that stars can be determined as lost or not. Due to the noisy characteristics of images of the sky, it is not so trivial determining if a star is lost or not. This projects advances the understanding of lost stars by creating a more efficient way of determining if a star is lost.**

I. Introduction

Observing the sky and detecting new stars have been an areas of great interest for astronomers for centuries. It grants astronomers and astrophysicists a greater understanding of the universe by allowing them to study the behavior of celestial bodies and astronomical objects.

Recently, a phenomenon was discovered while observing stars [1], where it appeared that the stars had become dimmer than expected or had disappeared altogether. Upon further investigation by comparing pictures of the same field in the sky, it was discovered that lost stars were a reoccurring phenomenon.

A. Background

The interest in searching for lost stars has been growing due to the abnormality of the phenomenon. This search is of great interest for multiple observatories all over the world, one of which is the "Sonneberg observatory" [2]. The observatory has been operational since 1925 and has a large inventory of old and new pictures of the same fields in the sky. This enables a deep search for lost stars. In the past, this search has been done manually by hand and, although this is a reliable technique, it requires a lot of effort and time. To be able to find lost stars without excessive work will enable future astronomers to investigate this matter further. This report conducts research on how to find lost stars more efficiently using modern tools. Using Sonneberg observatories data of the sky and a new detection technique will enable the findings of potential lost stars.

B. Contribution

The main contributions to this study of finding lost stars are:

- Identification of an efficient strategy.
- Implementation of strategy on real images.
- Validation of strategy.
- Evaluation of complete work.

Collectively, these contributions provide an efficient way of finding lost stars. In addition, this work lays the foundation for any further work that could be made.

C. Organization

The following sections present and explain the implementation process for finding lost stars. The report is organised as follows:

- **Method**: Establishes a robust solution and strategy for finding lost stars.
- **Results**: Demonstrates the implementation of the solution on images from the Sonneberg Observatory.
- **Discussion**: Explores the results, highlights the challenges encountered, and identifies opportunities for future work.

II. Methodology

In order to analyse fields in the sky and determine if there are any potential lost stars in an image, a strategy must be established. As mentioned in section I-A, previously, detection and determination were performed by humans by manually comparing images. This method is not feasible if a large investigation is to be performed due to the large amount of time required. Thus, to increase the time efficiency when comparing images, the usage of more advanced tools is necessary.

The method for efficiently determining lost stars follows the schematic presented in Fig. 1.

Fig. 1. Schematic of finding lost stars

A. Image Cropping - Tiles Creation

In general, images of fields in the sky contain huge amounts of information, so capturing it all requires a very high resolution of the image. Processing an image of this size requires large amounts of processing capacity and is not an efficient way to handle this problem. Therefore, we decided to opt for the following approach, where each image can be processed as multiple smaller images by dividing the image into smaller sections, i.e. tiles of the full image. This enables each tile of the old and new image to be compared without the use of massive processing power. This process is illustrated in Fig. 2.

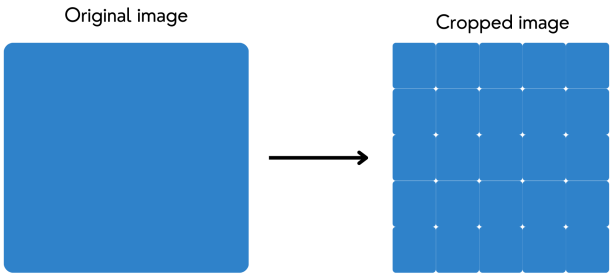

Fig. 2. Image transformation into tiles. The color blue indicating the night sky of an image

In practice, the original sky images provided by Sonneberg observatory needed to be rotated by 180 degrees first. Since the file format is .tif, this task turned out to be not as trivial as one might have thought. However, we used a Linux library *libvips* [3], which allowed for a quick and most importantly efficient way to rotate images, simply by using the command line. Once this step was completed, it was important to take into account the size of the original image and the limitations that arise from it in relation to tile creation and cropping itself. Furthermore, a plausible approach was needed to ensure that the original images were cropped correctly. The original size of the images was 13825×13825 pixels and it was decided to create tiles of size 2048×2048 pixels. The reasoning behind this was to make sure that there is a reasonable number of tiles created, while preserving the maximum possible information within them. To be able to do that, it was also needed to upscale the image. The whole process is handled within a function, which corresponds to a pseudocode, that is depicted in Algorithm 1.

Algorithm 1 Tiles creation - cropping

1: **Input:** `image_path`, `tile_size`
2: Open the image file from `image_path`
3: Get the image width and height
4: Calculate the number of tiles needed in the horizontal direction:

$$x\_tiles = \lceil \frac{\text{width}}{\texttt{tile\_size}} \rceil$$

5: Calculate the number of tiles needed in the vertical direction:

$$y\_tiles = \lceil \frac{\text{height}}{\texttt{tile\_size}} \rceil$$

6: Calculate the new image width and height:

$$\text{new\_width} = x\_tiles \times \texttt{tile\_size}$$

$$\text{new\_height} = y\_tiles \times \texttt{tile\_size}$$

7: Resize the image to the new dimensions
8: Create a folder for tiles with the same name as the image file
9: **Output:** Resized image and tiles folder created.

B. Tiles Transformation and Alignment

New and old images of the same field in the sky can have different properties. They can, for example, be misaligned, have different light levels, different amounts of noise, etc. This is a problem when comparing images using software. Due to the original images being inverted, i.e. stars appear as dark spots and space appears as a bright whitish field, the images had to be inverted back (i.e. more human natural way, where a star is white and the background is black) to enable the detection of stars or bright regions in the field. By identifying the brightest stars in each tile pair and aligning those tiles accordingly, the alignment happens in a robust manner. To form a complete picture, the tiles must be transformed back to a complete image. This process is illustrated in Fig.3

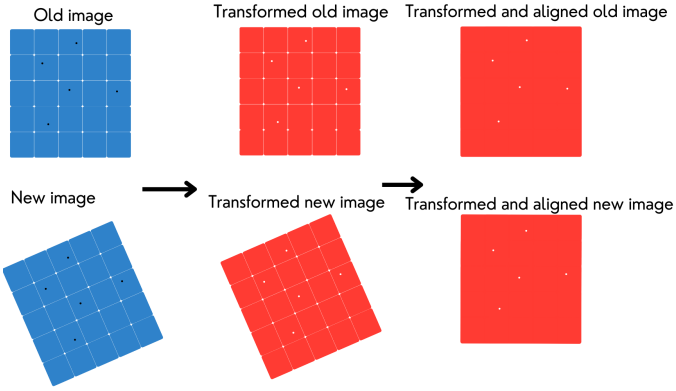

Fig. 3. Images are color inverted (indicated in red), aligned and transformed to a full image

Now, in concrete terms, to first invert individual tiles, utilisation of the *numpy* library [4] and its function *invert* was made. Thus providing a straightforward solution to the problem. However normalization of the tiles was needed. This was solved using the following method, as can be seen in Algorithm 2.

Algorithm 2 Normalize Image

Input: `image_path`
Open the image at `image_path`
Convert the image to a numpy array
Normalize the image using the formula:

$$\text{normalized image} = \frac{\text{image} - \text{image.min()}}{\text{image.max()} - \text{image.min()}} \times 255$$

Output: Normalized image as numpy array.

Once these operations are completed, a search for objects in the tiles is made, i.e. stars, that are above a certain threshold. Put differently, the identification of the brightest stars in each tile is made. The pseudocode for this follows, see Algorithm 3.

Algorithm 3 Find Brightest Spots in an Image

Input: Image (`image`), Brightness threshold (`threshold`), Minimum blob size (`min_blob_size`), Maximum stars to detect (`max_stars`)
Create a binary mask for pixels brighter than the threshold.
Label connected components in the binary mask.
Filter out blobs smaller than `min_blob_size`.
Calculate centroids of the connected components.
Filter the centroids to retain only the brightest:
if number of centroids exceeds `max_stars` **then**
    Sort centroids by brightness and retain the top `max_stars`.
end if
Output: The list of star coordinates.

An important step here is the usage of so-called *blobs*. These represent the actual star, since otherwise we would end up with a huge number of individual pixels located close to each other. Those would correspond to a single object to a human eye, but computers see it differently.
At this point, we have the coordinates of the brightest stars in each tile formatted as python arrays. However, prior to moving forward, we need to sort them in a very particular way. Since the stars coordinates in the arrays are unsorted, we first calculate the distance between each of the points. We then keep only the `num_matches` best matches in terms of minimal distance between them. This comes down to the assumption that the transformation between two tiles is not major, and thus the position of the corresponding stars has not shifted much. See Algorithm 4.

Algorithm 4 Sort Corresponding Stars Based on Minimal Distances

Input: Stars coords in array1 `array1`,
Stars coords in array2 `array2`,
Number of stars to keep `num_of_stars`
distances $\leftarrow$ Empty list.
for all points p_1 in **array1 do**
    for all points p_2 in **array2 do**
        Compute **dist** $\leftarrow \|p_1 - p_2\|_2$.
        Append $(\textbf{dist}, i, j)$ to **distances**.
    end for
end for
Sort **distances** based minimal distance.
best_matches $\leftarrow$ First **num_matches** elements of **distances**.
reordered_array1 $\leftarrow$ [**array1**$[i]$ for $(\_, i, \_)$ in **best_matches**].
reordered_array2 $\leftarrow$ [**array2**$[j]$ for $(\_, \_, j)$ in **best_matches**].
Output: Reordered array1 and array2.

Once we have the sorted coordinates of the stars at hand, we try to find the full transformation using the *astroalign* library [5].

C. Difference Map Creation

In the next step, the aligned and transformed tiles are compared to create a difference map where differences in light level are an indication of something that could possibly be identified as a lost star, see Fig. 4. In terms of code, see Algorithm 5.

D. Identification of lost star

It also worth mentioning, that we needed to ensure that there is a lost star indicated and not some interference from noise or interference of disturbance with the image. That is why we tried to perform multiple iterations of the identification process. Consequently, if the difference is consistent across multiple difference maps, it can be concluded that a possible lost-star candidate has been found.

Algorithm 5 Compute Difference Map Between Two Images

1: **Input:** Two images `image1` and `image2`
2: Calculate the absolute pixel-wise difference between the two images:

```
difference_map = abs(image1 - image2)
```

3: **Output:** Absolute difference map `difference_map`.

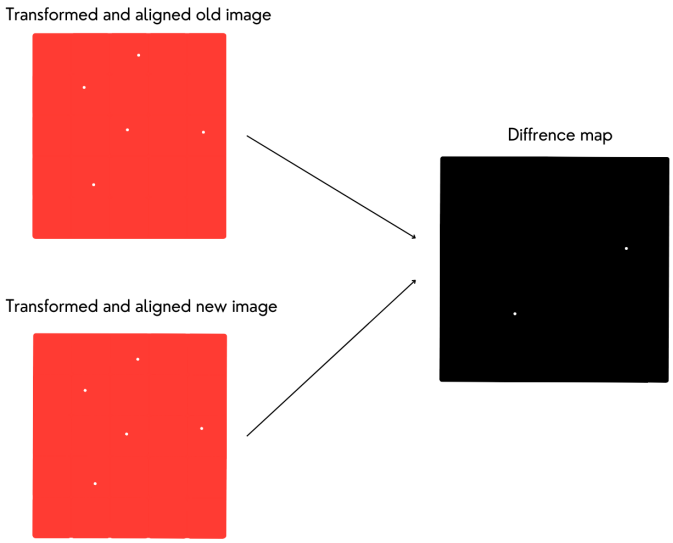

Fig. 4. Theory behind the creation of a difference map.

III. Results

By using the images from Sonneberg observatory in the established method, we can determine if there is a lost star in a certain field in the sky. An example of the chosen fields of the sky is as follows:

- **Old Images:** 602940, 602943 & 602942
- **Recent Images:** 603169, 603172 & 603171

Presented in this section is the analysis between images 602940 and 603169. Other examples of analyses can be seen in the appendix A. The final result, that is, the identified stars, the footprint of the transformation, and the transformed image, is shown in Fig. 5.

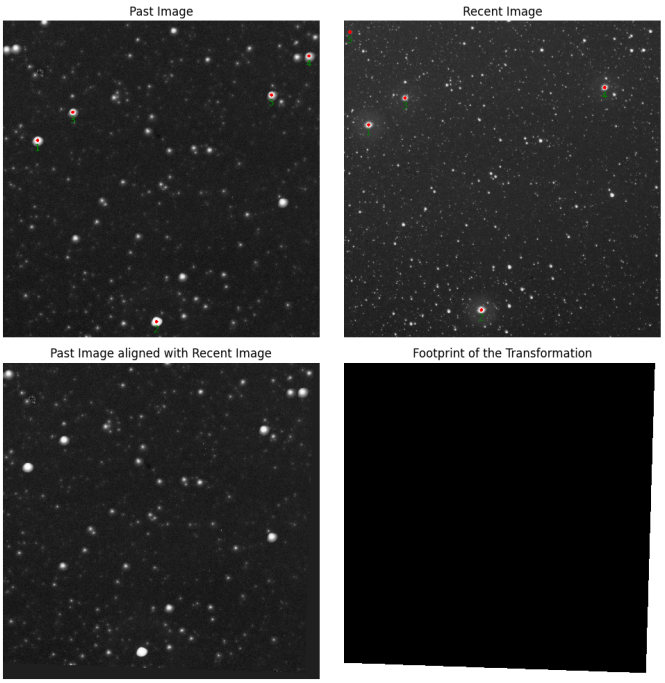

Fig. 5. Transformation showcase. Images 602940 and 603169.

The final result of the whole process is depicted in Fig. 6.

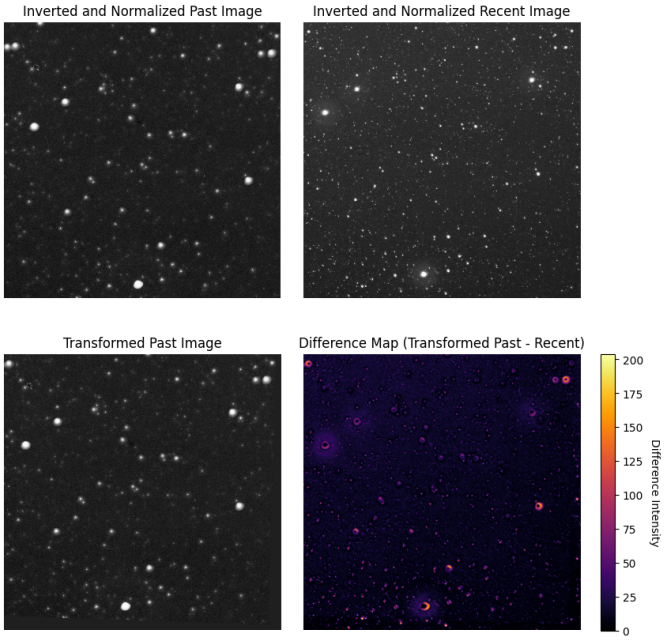

Fig. 6. Difference map example. Images 602940 and 603169

The creation of a difference map explained in II-C is depicted in Fig. 7. As one can see, the brighter a particular spot is, the higher the chance that a potential candidate has been found. Although in most cases, further inspection is required.

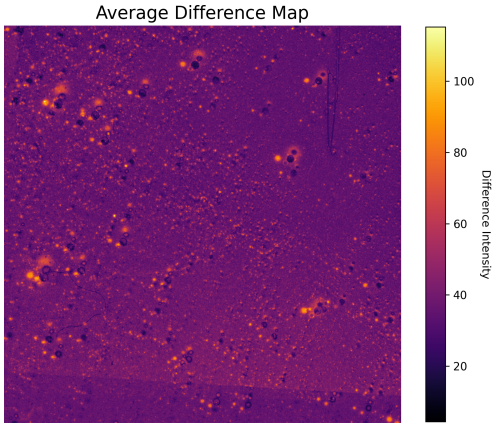

Fig. 7. Average difference map example for a particular tile from image pair set of four images.

The full code can be found on the following Github repository.

IV. Conclusion and Discussion

We have presented our approach to the problem and the results achieved. Therefore, it is now appropriate to reflect on the results achieved, discuss, and identify potential improvements in our methodology.

It goes without saying that the complete process of finding stars can be extremely difficult, especially due to the size and the resolution of the images that are considered, i.e. determining if a star is missing when there are only ten of them is trivial, on the other hand, in case there are thousands of stars to consider, it becomes significantly harder.

In addition, images are often noisy, unbalanced, or corrupted in a particular way. We tried to deploy various techniques to combat this, but in the end the full-scale problem of image processing turned out to be much more difficult than we originally imagined it to be and therefore only reduced amount of attention has been paid to it. We believe that by improving the quality of original images, e.g. through further preprocessing or by e.g. applying correct denoising techniques, the whole pipeline and subsequently the results could be improved significantly.

As seen in Fig. 5, as long as the algorithm was able to detect a sufficient number of stars and the assumption about their minor shift did hold, the transformation found is adequate. That holds for the corresponding difference map as well, refer to Fig. 6 again. As we can see, apart for a certain glow around the stars, the difference map tells us that not much, in terms of stars vanishing or appearing again, has changed. Certain bright spots appear only due to small errors in the found transformation, but nothing that would confuse a human reader. Refer to Fig. 8, Fig. 10 and Fig. 9, Fig. 11 for additional examples.

It is worth mentioning the multiple improvement options that arise when determining and applying the transformation:

- Firstly, a strictly set threshold might not suit all images equally. While it may be good enough to detect ten stars in one image, at the same time it might prove to be powerless when applied to a different image. This could be solved by allowing the user to recognise the star position correspondence themselves or by applying a different technique altogether. However, we did our best to streamline the process as much as possible, which in some cases might have resulted in incorrect transformations and thus faulty results (plots).
- Secondly, it would be favourable to consider bigger tiles, since that would allow more stars to be included in the picture, eliminating situations where a portion of a star gets cropped into multiple tiles at the same time. However, that would most likely turn out to be more computationally demanding.
- Another further improvement could be using AI or machine learning. A neural network could be trained to either denoise the images, to identify the lost stars by analysing given images, or even both.

Regarding the average difference maps, there are also possible improvements. One problem we came across was when a particular tile pair did not produce a difference map, for one reason or another or the transformation found, and therefore the difference map for that particular pair was incorrect or corrupted. This would result in a skewed average map or no average map at all. However, there were some promising results, see Fig. 13 or Fig. 12.

References

[1] S. Gills, "Some of the universe's stars have gone missing. but where did they go?" *space.com*, May 2021, accessed: Nov. 30, 2024. [Online]. Available: https://www.space.com/hunt-for-universe-missing-stars-space-mysteries

[2] S. observatory. Die geschichte der sternwarte sonneberg. Accessed: Nov. 30, 2024. [Online]. Available: https://www.astronomiemuseum.de/sternwarte

[3] libvips - a fast image processing library with low memory needs. https://www.libvips.org/. Accessed: 21.12.2024.

[4] Numpy - the fundamental package for scientific computing with python. https://numpy.org/. Accessed: 21.12.2024.

[5] Astroalign. https://astroalign.quatrope.org/en/latest/. Accessed: 21.12.2024.

APPENDIX

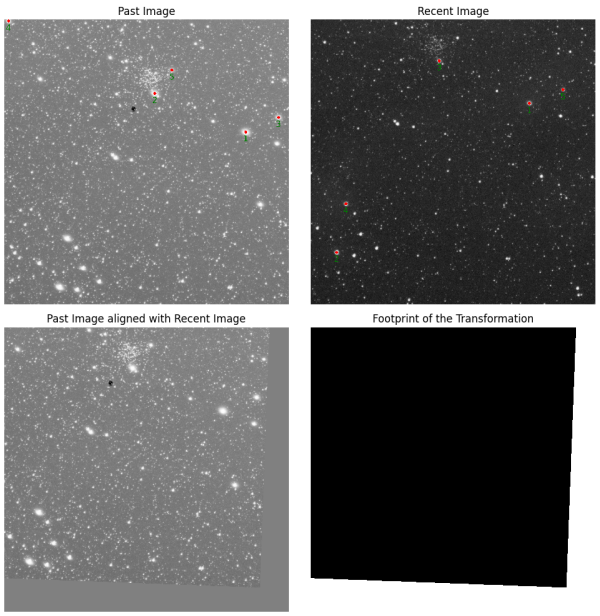

Fig. 8. Transformation example. Image 602943 and 603172.

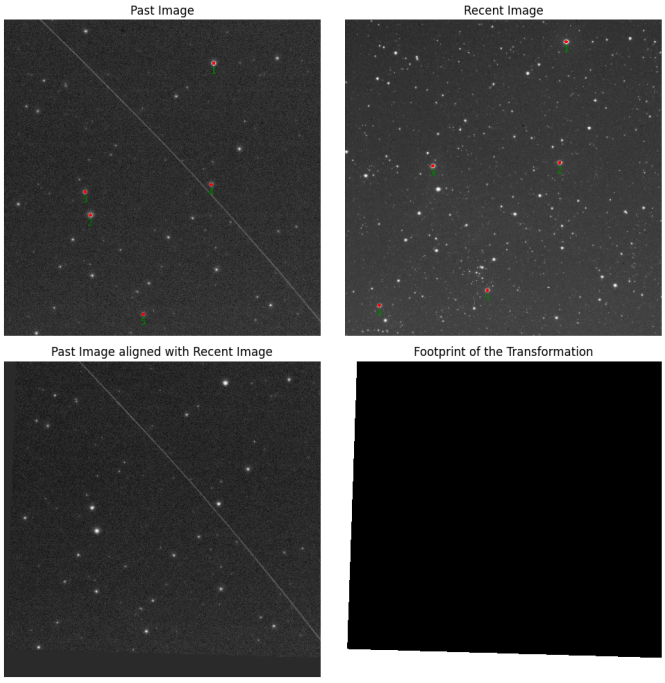

Fig. 9. Transformation example. Image 602942 and 603171.

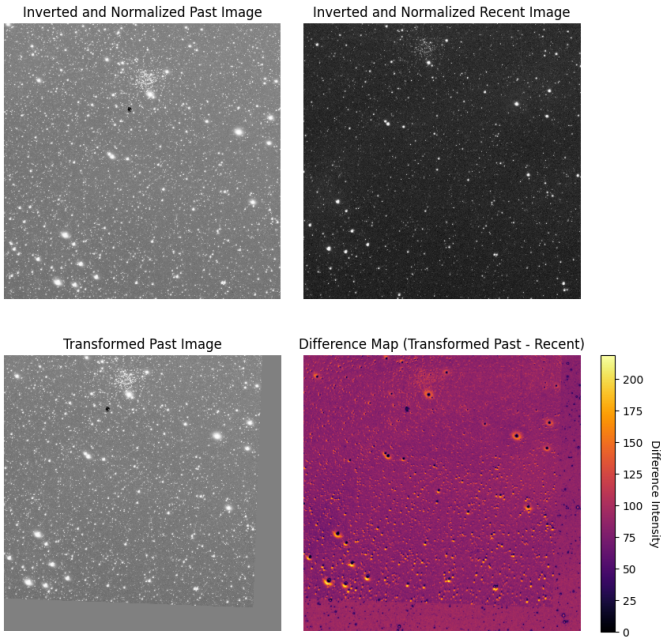

Fig. 10. Difference map example. Image 602943 and 603172.

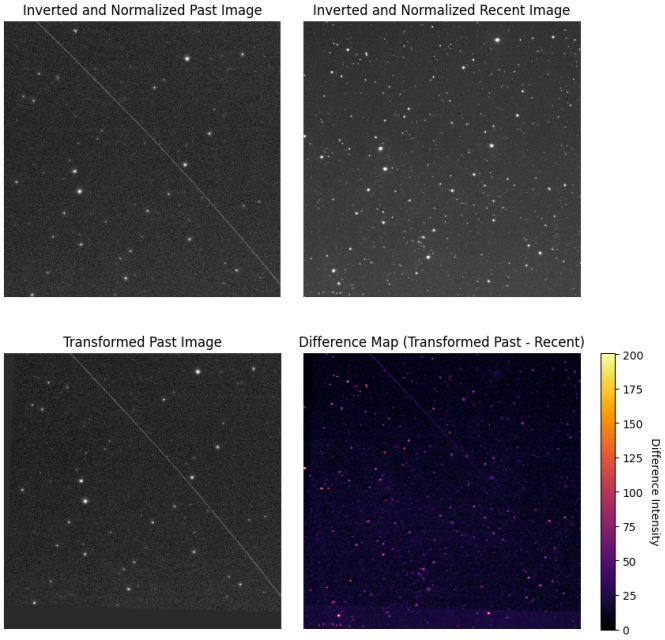

Fig. 11. Difference map example. Image 602942 and 603171.

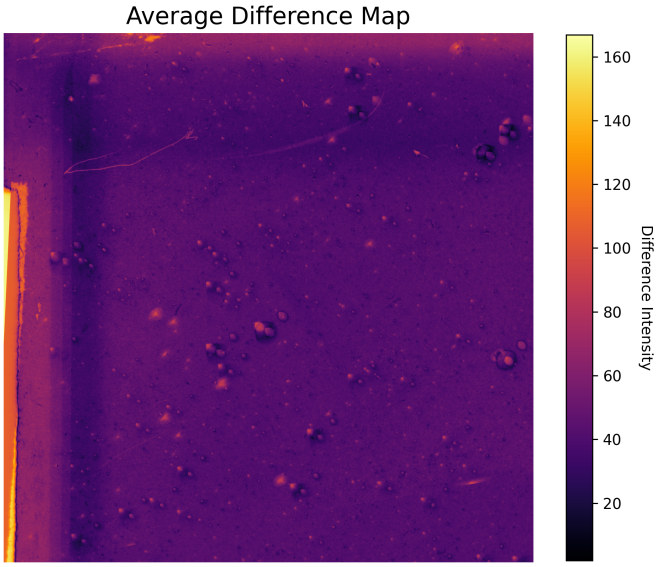

Fig. 12. Average difference map example for a particular tile from image pair set of four images.

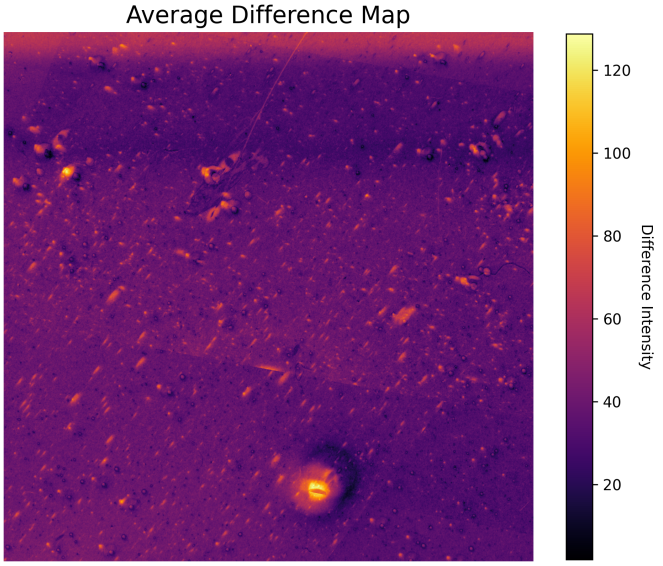

Fig. 13. Average difference map example for a particular tile from image pair set of four images.