Appendix

A. Lens Distortion Correction

To correct the radial distortion of the overhead Canon DSLR camera, which was installed for flow mapping purposes, we used Adobe Photoshop's Auto Lens Correction feature, which uses the camera-specific EXIF metadata to apply the appropriate distortion profile (see Figure 1). Complete code and details for this step can be found in the script `Step01_DistortionCorrection.py` on our GitHubrepository. For additional guidance, please refer to How to Use Photoshop Lens Correction Filter and How to Use Lens Correction in Adobe Photoshop Tutorial.

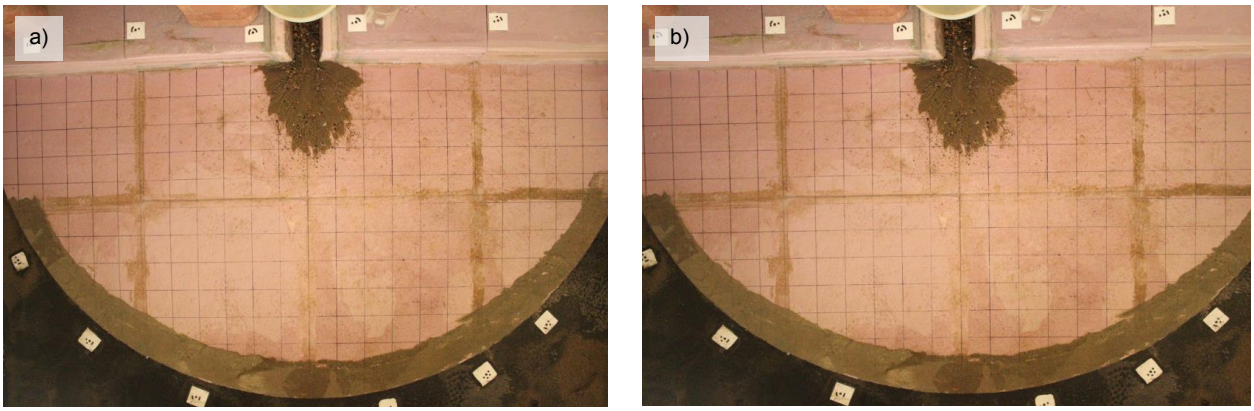

Figure 1: Correction of radial distortion in overhead imagery. (a) The raw image captured by the overhead camera, and (b) The corrected image after applying lens distortion correction using Adobe Photoshop's Auto Lens Correction tool.

B. Homography Matrix Calculation

After correcting for lens distortion, we imported both the adjusted images and the reference orthophoto into Python for alignment. We identified key reference points in each corrected image and matched them to corresponding points in the orthophoto. These point pairs were then used to calculate the homography matrix, which defines the transformation needed to align the images (see Figure 2). The code for this procedure can be found in the `Step02_Image_Matchingpoints_Selection.py` script in the same GitHub repository.

To compute this matrix, we used OpenCV's `cv2.findHomography` function (Luong & Faugeras, 1996), which leverages the RANSAC algorithm. This method helps estimate the transformation while minimizing the impact of outliers, ensuring a more accurate alignment. Once the homography matrix was determined, we applied `cv2.warpPerspective` to each corrected image. This transformation aligned the images with the orthophoto's coordinate system while also compensating for slight shifts in camera position over time. As a result, the images remained spatially consistent throughout the experiment. The relevant code is provided in `Step03_HomographyMatrix.py` in the same GitHub repository.

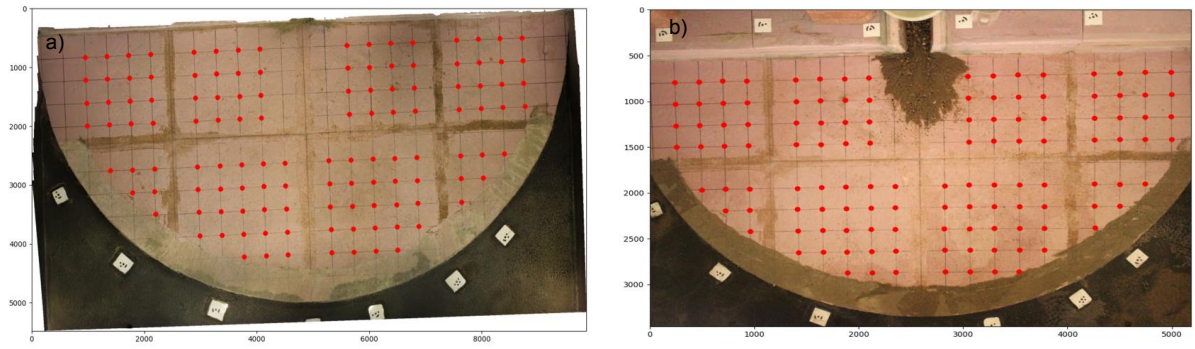

Figure 2: Matching key reference points between (a)the ortho image and (b) corrected images.

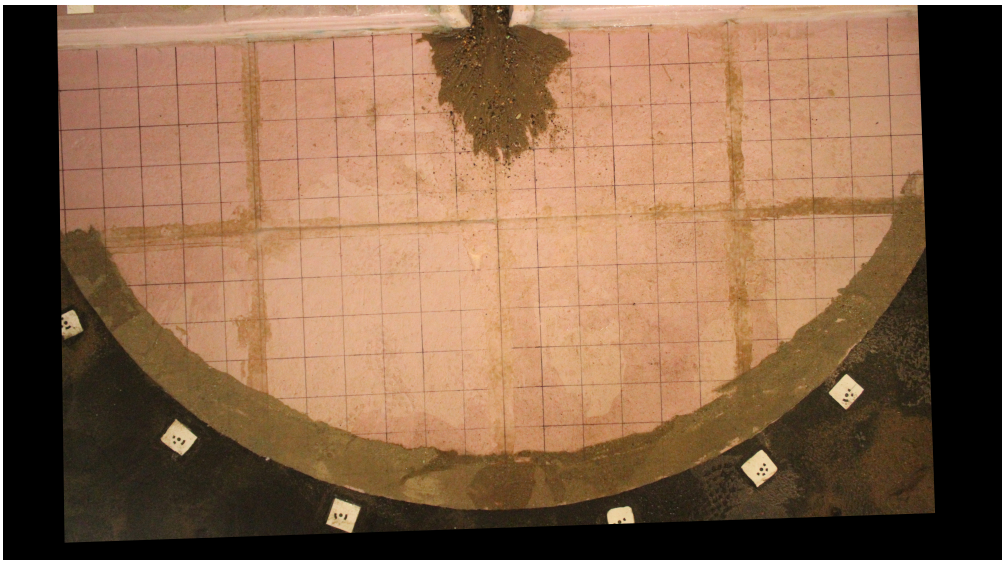

Figure 3: Corrected image after applying transformation through homography matrix.

C. Re-Projection Error and Validation

To assess alignment accuracy, we calculated the re-projection error, defined as the Euclidean distance between the transformed points in the corrected image and their corresponding points in the orthophoto. These errors were visualized using a histogram (Figure 4 (a)). The peak at lower values indicates that the transformation was generally successful, although some points may have been affected by mismatches or remaining distortions. The mean re-projection error was 8.65 pixels, which corresponds to approximately 4 mm given an image resolution of 0.0005 meters per pixel. This level of misalignment is considered acceptable for the purposes of this study. As illustrated in Figure 4 (b), points with the highest reprojection errors tend to be located in areas beyond the fan's extent. The overlay of the transformed image onto the reference image is also shown in Figure 5.

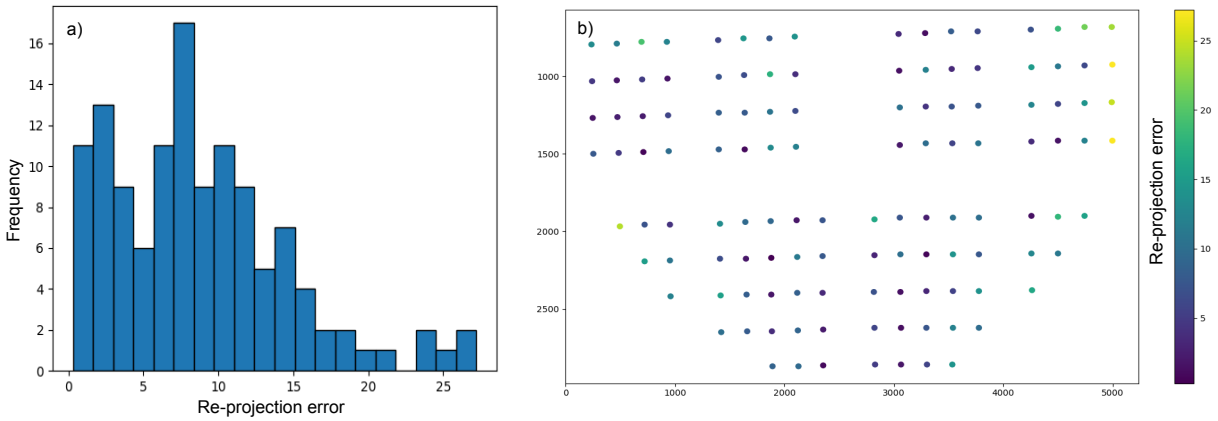

Figure 4: Distribution of re-projection errors following homography-based image transformation.

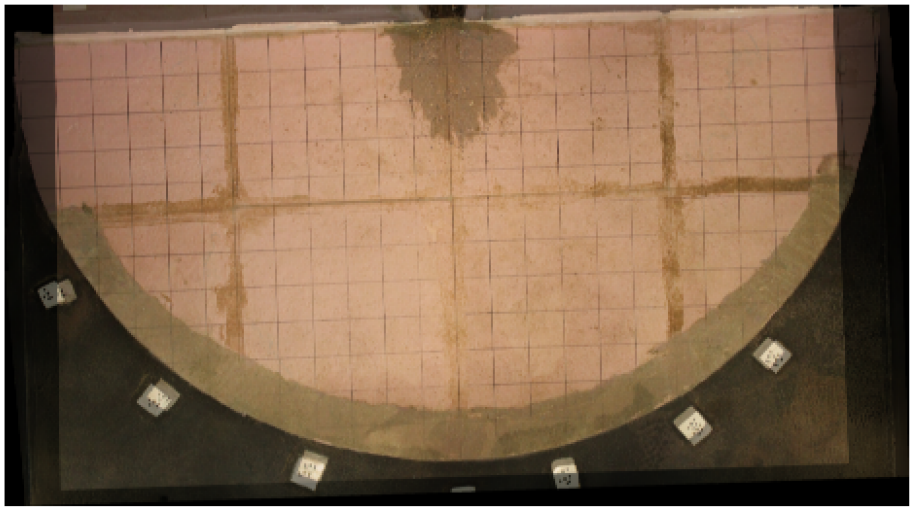

Figure 5: Overlay of orthoimage and transformed image post homography application.