MATEC Web of Conferences **370**, 07012 (2022) https://doi.org/10.1051/matecconf/202237007012
2022 RAPDASA-RobMech-PRASA-CoSAAMI Conference

Markerless monocular vision-based localisation for autonomous inspection drones

Gert Nel[1] , *Jacobus Adriaan Albertus* Engelbrecht[2], and *Herman Arnold* Engelbrecht[3]

Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa, 21567042@sun.ac.za[1], jengelbr@sun.ac.za[2], hebrecht@sun.ac.za[3]

Abstract. This paper presents a markerless vision-based localisation algorithm to enable an autonomous inspection drone to determine its pose relative to a known inspection target. During an offline mapping phase, a 3D catalogue of the inspection target's persistent image features is created and stored. A neural network is trained on regions of interest of the images for fast segmentation. During the online localisation phase, the images are first segmented and the detected features in the segmented areas are matched with the stored features in the target's 3D catalogue. A pose estimation algorithm is applied to the matched features to determine the pose of the drone relative to the target. Practical experiments show promising results with small position and attitude errors.

1 Introduction

Drones are becoming an increasingly popular tool to perform aerial inspections. The autonomous navigation of drones is highly dependent on the sensors available for localisation. The most basic localisation sensors all drones possess are an Inertial Measurement Unit (IMU) and the Global Positioning System (GPS). The IMU onboard a drone usually exhibits significant bias error and depends on GPS for more accurate localisation. In an environment where GPS is not readily available, or not available at all, the localisation accuracy depends solely on the IMU and results in large errors caused by drifting.

Simultaneous Localisation and Mapping (SLAM) [1] offers a solution where additional sensors are used to obtain a more accurate localisation, typically with cameras and Light Detection and Ranging (LIDAR). Vision-based SLAM makes use of only images to provide the algorithm with information for localisation. The algorithm extracts points of interest, known as features, from an image that can be identified in other images of the same object. The most common feature detection algorithm used in SLAM is known as the Oriented FAST and rotated BRIEF (ORB) algorithm as it is very efficient and can execute in real-time [2]. For computer vision tasks, such as structure from motion (SFM), the scale-invariant feature transform (SIFT) or speeded up robust features (SURF) algorithm is typically used. This is due to the algorithms' robustness to different scales and orientations.

SLAM provides a good solution when the drone is placed in an unknown environment. Inspection drones, however, typically inspect the same object in a mostly static environment. Therefore, creating a new map for every inspection is redundant. Making use of the same point cloud map containing features of the environment for every inspection reduces the frequency of problems arising from SLAM such as loop closing and relocalisation. More processing can also be accomplished when the point cloud is not created simultaneously with the localisation, allowing for better features and more robust methods.

The main contribution of this paper is to propose a novel pose estimation solution for inspection drones in GPS-denied areas. The method is similar to SLAM except that the mapping and localisation phases happen at different times. The algorithm requires more post-processing computation during the mapping phase, but in return reduces the computational time required during the localisation phase. The unique approach taken for data processing decreases the computation time as mentioned while maintaining the equivalent — and in some cases better — results in terms of accuracy. This can be seen in the results of the preliminary experiments. The algorithm proposes markerless localisation which implies that it will not make use of predetermined markers such as arUco markers or any form of generated marker that is added additionally to the inspected object to aid in the localisation process. The algorithm makes use of a single RGB camera, monocular camera, for all phases throughout the process.

This paper aims to show a proof of concept that the algorithm can be used and can be scaled to larger and more complex problems. Section 2 provides an overview of the related work. Section 3 describes the methodology and looks at both the mapping phase and the localisation phase. Section 4 presents and discusses the results of the preliminary experiments. Section 5 summarises the conclusions and main findings.

2 Related Work

The most widely used vision-based localisation method is SLAM. The SLAM algorithm allows for localisation of the camera with no knowledge about the environment. Barros et al. [3] published a paper on the performance and properties of the most common SLAM algorithms. The article compares the different results which indicates that monocular SLAM does not perform as well compared to the algorithms with additional sensors. Monocular SLAM has the limitation that it can only do sparse reconstruction due to only reconstructing tracked features. This limits the number of features stored in the point cloud and is known as the sparse cloud problem. The paper aims to improve the results from monocular SLAM by proposing an algorithm that splits the localisation and mapping phase. This will allow for an improved mapping algorithm as there is more time for processing which in turn will increase the accuracy in the localisation phase.

The concept of pre-mapping a known area before visual localisation exists in some work that is published. Lee et al. [4] presented an efficient visual localization method for a monocular camera, which suggests making use of two phases. In the offline phase, a database of images from the mapped area is created. Global features and local features are extracted from the images to create a model used for place detection and pose estimation. The online phase consists of two sides, namely the server-side and client-side. The server uses GPUs to run the model. This allows for place detection and localisation from the live video received. An online map is constructed as the environment is explored. The client determines its pose by comparing the constructed online map, to the offline map. However, this method does require

a dedicated server to run the model while the client is busy using the localisation functionality.

Xiao et al. [5] presented a monocular vehicle self-localization method which makes use of known information about the road landmarks to aid with the location estimation for the vehicle. A semantic map of the road landmarks is first constructed. This is done by extracting the landmarks' point cloud data and forming a semantic map. For the localisation of the vehicle, the video frames are passed through semantic segmentation. The information returned from the algorithm is inserted into a neural network to calculate the first iteration of the pose estimate. The landmarks stored in the database are projected to the estimated orientation and the reprojection error is iteratively reduced until the optimal pose is estimated. However, this method requires human input to label the road landmarks and does not contribute many benefits in areas where these landmarks cannot be observed. It does however show promising results for the use of segmentation in localisation problems.

3 Methodology

3.1 Overview

In our proposed method, the mapping and localisation of the static environment happens in two stages, which are visually illustrated in Figure 1. During an offline mapping phase, a 3D catalogue of the inspection target's persistent image features is created and segmented into objects. An object detection neural network is trained to detect and segment these feature clusters. During the online localisation phase, the neural network detects and segments the feature clusters in the drone's field of view that correspond to the feature clusters in the target's 3D catalogue. The detected features in the segmented areas are then matched with the corresponding stored features in the target's 3D catalogue. The purpose of the segmentation is to reduce the number of features that need to be matched by identifying the relevant window of stored features in the 3D catalogue. A pose estimation algorithm is then applied to the matched features to determine the pose of the drone relative to the target.

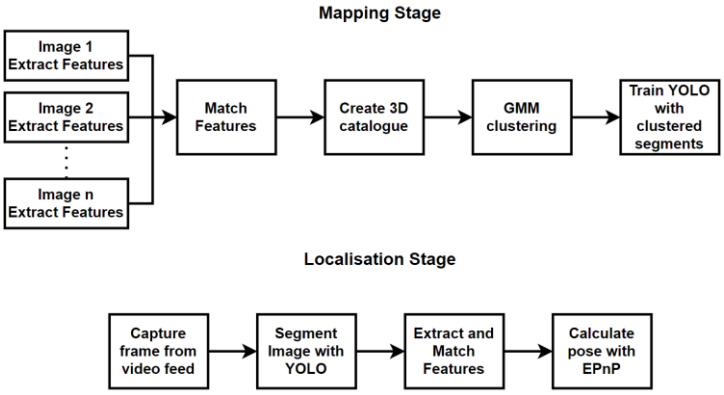

Fig. 1. An overview of the proposed algorithm.

3.2 Mapping

The offline mapping is performed by taking images of the target from known locations and orientations and extracting the SIFT features from the images. The common features between

these images are matched. The matched features are passed through Lowes' ratio test [6] to remove bad matches. This allows for at least two perspectives of a target. The corresponding points, $\mathbf{x_i}$ and $\mathbf{x_i'}^{\mathrm{T}}$ from the corresponding images can be used to form linear equations, as seen in Equation 1. These points are used to calculate the fundamental matrix, **F**.

$$\mathbf{x_i'}^{\mathrm{T}}\mathbf{F}\mathbf{x_i} = 0 \tag{1}$$

The random sample consensus (RANSAC) algorithm is used to remove outliers from the dataset. This is done by picking random samples from the dataset and calculating the fundamental matrix from the smaller, sampled dataset. The inlier points are then determined by ensuring that the left side of Equation 1 evaluates to a value which is sufficiently close to zero. The fundamental matrix with the most inliers is then used [7]. Thereafter, the outlier points are treated as noise and discarded from the dataset.

The camera matrix of each image is used for the triangulation of the inlier points. Matrix A is calculated — such that it satisfies $\mathrm{A}\mathbf{X} = \mathbf{0}$ — using Equation 2, where $\mathbf{p^{i\mathrm{T}}}$ represents the rows of the camera matrix.

$$\mathrm{A} = \begin{bmatrix} x(\mathbf{p}^{3T}) - \mathbf{p}^{1T} \\ y(\mathbf{p}^{3T}) - \mathbf{p}^{2T} \\ x'(\mathbf{p'}^{3T}) - \mathbf{p'}^{1T} \\ y'(\mathbf{p'}^{3T}) - \mathbf{p'}^{2T} \end{bmatrix} \tag{2}$$

Taking the singular value decomposition of A ($\mathrm{A}=\mathrm{USV}^T$), the unit singular vector corresponding to the smallest singular value of A is equal to $\mathbf{X}/a = \begin{bmatrix} \frac{X}{a} & \frac{Y}{a} & \frac{Z}{a} & \frac{1}{a} \end{bmatrix}^T$. This process is repeated for every corresponding point. The process used is known as triangulation and is described in the book *Multiple View Geometry* [8]. An example of features represented where they can be found in 3D space can be seen in Figure 2, as well as the example inspected object. The purpose of the 3D representation is not to create a dense reconstruction of the image, but rather a catalogue of features that can easily be identified with a high level of repeatability.

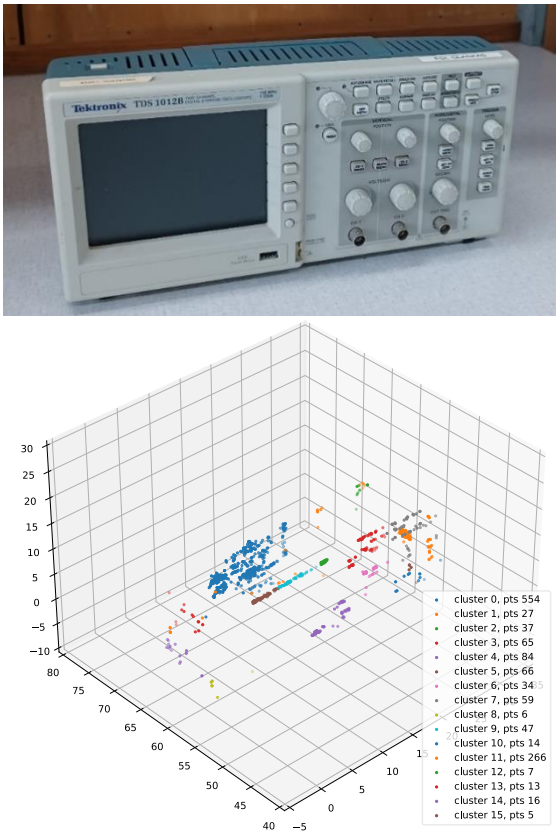

Fig. 2. The inspected object (top) and the point cloud of interesting features (bottom).

After a point cloud of descriptors is created, the data is divided into segments for faster feature extraction and matching during the localisation phase. The point cloud is first divided into planes with the RANSAC algorithm. This is done to create surfaces that can easily be identified in images. The algorithm creates a plane from three random data points. The distance from the plane to the data points is calculated with the following equation:

$$\text{Distance} = \frac{|ax+by+cz+d|}{\sqrt{a^2+b^2+c^2}} \quad (3)$$

Where x, y and z represent a coordinate in three-dimensional space and the variables a, b, c and d represent the coefficients that describe a planar equation ($ax + by + cz + d = 0$). If the distance is larger than the specified threshold, the point is classified as an outlier. This process is repeated to find the optimal plane. The classified inlier points are then removed from the dataset to find a new plane. The algorithm is repeated iteratively until no new plane can be detected. The identified plane is then further segmented. To accomplish this, a clustering algorithm is applied to the planes. The Gaussian mixture model (GMM) is used to accommodate for varying dimensions and clusters not parallel with the axis or plane. The optimal number of clusters is determined by the silhouette score as described by the paper published on validation of clusters [9]. The score *S* is calculated using the following equation:

$$s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))} \quad (4)$$

Where i represents any point in a cluster and $a(i)$ is the average distance between point i and the points within the same cluster. The value $b(i)$ is calculated by first calculating the average distance between point i and the points from neighbouring clusters, and determining the average distance between i and the neighbouring cluster that yields the smallest average distance. The average silhouette score for all points, known as the silhouette coefficient, describes how well the clusters are grouped. The number of clusters with the largest silhouette coefficient represents the optimal number of clusters.

The video frames or images from the mapping process are used to train a convolutional neural network. The segments are identified in the images or frames by using feature recognition. Bounding boxes are constructed for the identified segments to create labelled data to train the You Only Look Once (YOLO) model [10]. This is done by projecting the segmented points back onto images of the inspected object and constructing a rectangular box around the cluster of points. The training data undergoes data augmentation to synthetically increase the size of the training set and reduce overfitting. The augmentation steps consist of adding a Gaussian blur, changing the brightness, and cropping the image.

The additional augmented images are generated by using a randomly picked combination of the mentioned augmentation steps. The Gaussian blur on images is achieved by convolving a 5x5 Gaussian filter kernel with the image. The brightness adjustment involves converting the image from RGB format to HSV format and increasing or decreasing the value by a random amount. The windows for the cropping of images are chosen at random. Three different versions of the original image are used, one being the original and two being augmented images.

3.3 Localisation

During the localisation phase, the video frames from the drone's onboard camera are processed. The YOLO neural network detects and segments the feature clusters in the video frame. The SIFT algorithm is executed on the segmented parts of the video frames. This allows for high-resolution images to be processed with the SIFT algorithm and results in more robust features. The features extracted from the segmented areas are matched to the clustered points in the database. This process reduces the time needed to match features to many data points. An example image of where the YOLO algorithm is applied is shown in Figure 3. The example shows the different clusters of the segmentation algorithm as well as the certainty of the allocation.

Fig. 3. The inspected object, segments and identified key features.

The Efficient Perspective-n-Point (EPnP) algorithm [11] is used to calculate the rotation and translation vectors of the camera by mapping the 3D database points with the corresponding 2D image points. The RANSAC algorithm is used to remove outliers by choosing random samples, running the algorithm, and checking the number of samples that fall within the specified threshold. The iteration with the most inlier points is chosen.

4 Results

A differential GPS can provide an accuracy of up to 1.5 cm and can provide localisation data up to 20 times per second. In an ideal situation, this will be the desired result, but in practice, this requires ground stations to help with the localisation process. In GPS scarce areas the visual localisation algorithm performs well and is in line with the GPS in terms of absolute positional and attitude accuracy. However, the localisation rate of the algorithm is much lower than 20Hz. This means that the algorithm cannot be used as a standalone solution in real-time scenarios. The experiments are designed to test the performance of the algorithm in different conditions.

An oscilloscope is used as the inspected object. Pictures are taken at known orientations to recreate a 360˚ view of the inspection object. Two pictures per side are used, which allows for two-view reconstruction. The data processing algorithms are applied to the point cloud at which the algorithm identifies four different planes with a total of fourteen clusters. Fifty additional images are taken at different orientations to use as training data for the YOLO algorithm. The labelled bounding boxes to train the YOLO algorithm are drawn on the images around the features detected from each cluster. To test the localisation algorithm, six new images at different views are taken, as can be seen in Figure 4.

The algorithm's performance is evaluated against the algorithm without the segmentation pre-processing. The performance criteria consist of the time used for localisation, and the total error in orientation compared to the measured orientation called the ground truth. The absolute positional error is calculated by taking the absolute difference between the estimated position and the measured position, as can be seen in Equation 5. The absolute attitude error is calculated by taking the difference between the estimated and measured extrinsic rotations, as can be seen in Equation 6. Comparison is made between the localisation of the segmentation algorithm and the algorithm that does not make use of the proposed segmentation method.

$$\text{pos error}: \ \|\mathbf{X_{est}} - \mathbf{X_{meas}}\| \tag{5}$$

$$\text{ang error}: \ \|\boldsymbol{\theta}_{\mathbf{est}} - \boldsymbol{\theta}_{\mathbf{meas}}\| \tag{6}$$

Figure 4 shows the inspected model's features in 3D space, with the correct locations of the camera indicated by the ground truth. The proposed segmentation algorithm and the algorithm without segmentation can be compared relative to the ground truth.

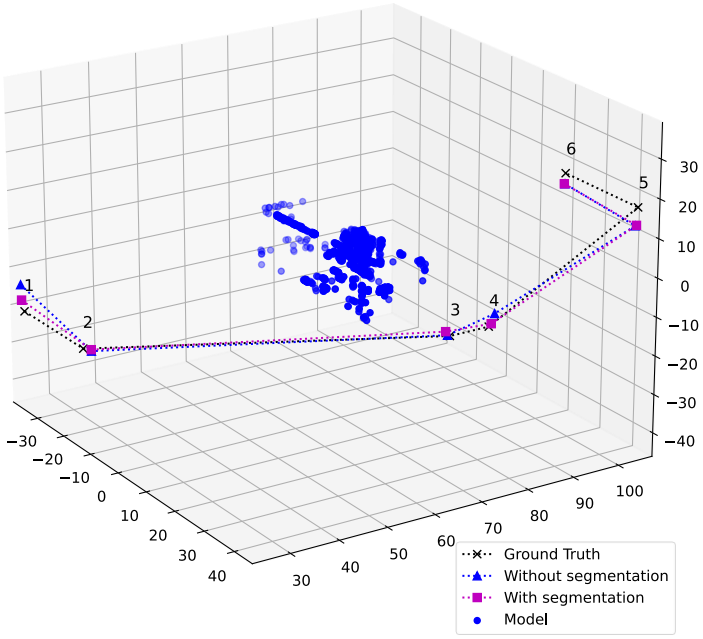

Fig. 4. The path of the camera around the inspected object.

The proposed method shows an average positional error of 2.3 cm, which is less than the algorithm without the segmentation, which shows an average positional error of 3.3 cm. This is due to fewer mismatches between the extracted and stored features. The algorithm with segmentation also shows a slight improvement in the attitude error, with an average of 3.7˚, in comparison with the algorithm without segmentation, which has an average attitude error of 4.3˚. The test results for the six different time instances shown in Figure 4 can be seen in Figure 5.

The most noticeable difference is in the time taken for the algorithm to execute. The proposed method has an average processing time of 0.282 seconds in comparison with the algorithm without segmentation, which has an average processing time of 1.584 seconds. This reflects a time improvement greater than 5.

Fig. 5. The execution time (bottom), the attitude error (middle) and the positional error (top) of the different approaches.

The algorithm is evaluated against a video of the inspection object. A cell phone camera is used as a portable camera. This introduces noise and motion blur to the video feed. The algorithm is tested against the video to investigate the effect of these irregularities on the

algorithms. Each frame from the video feed is used to generate a complete overview of the noise the algorithm experiences.

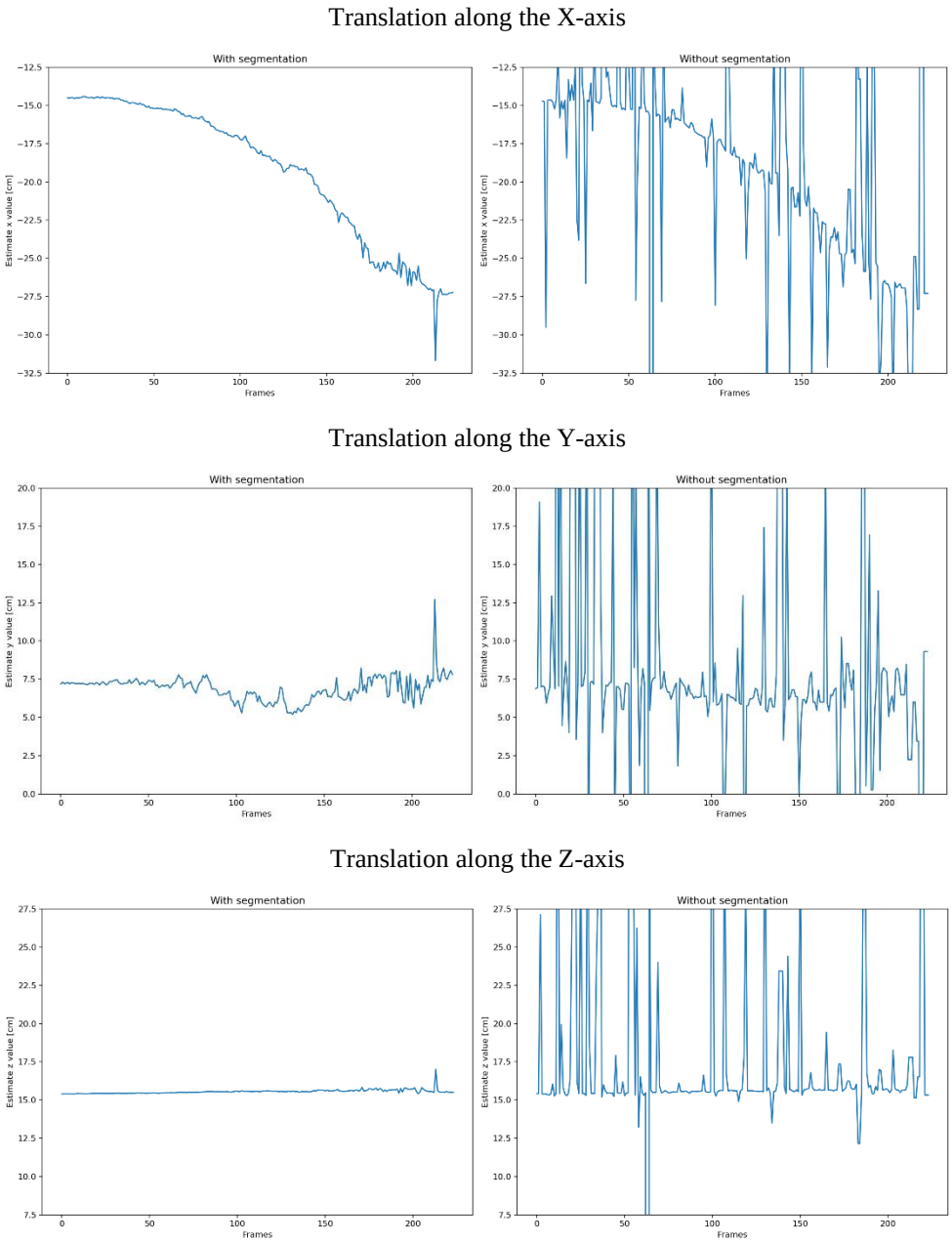

Fig. 6. The estimated position of the camera obtained from the video frames. The translation results from the algorithm with segmentation (left) and the results for the algorithm without the proposed segmentation step (right).

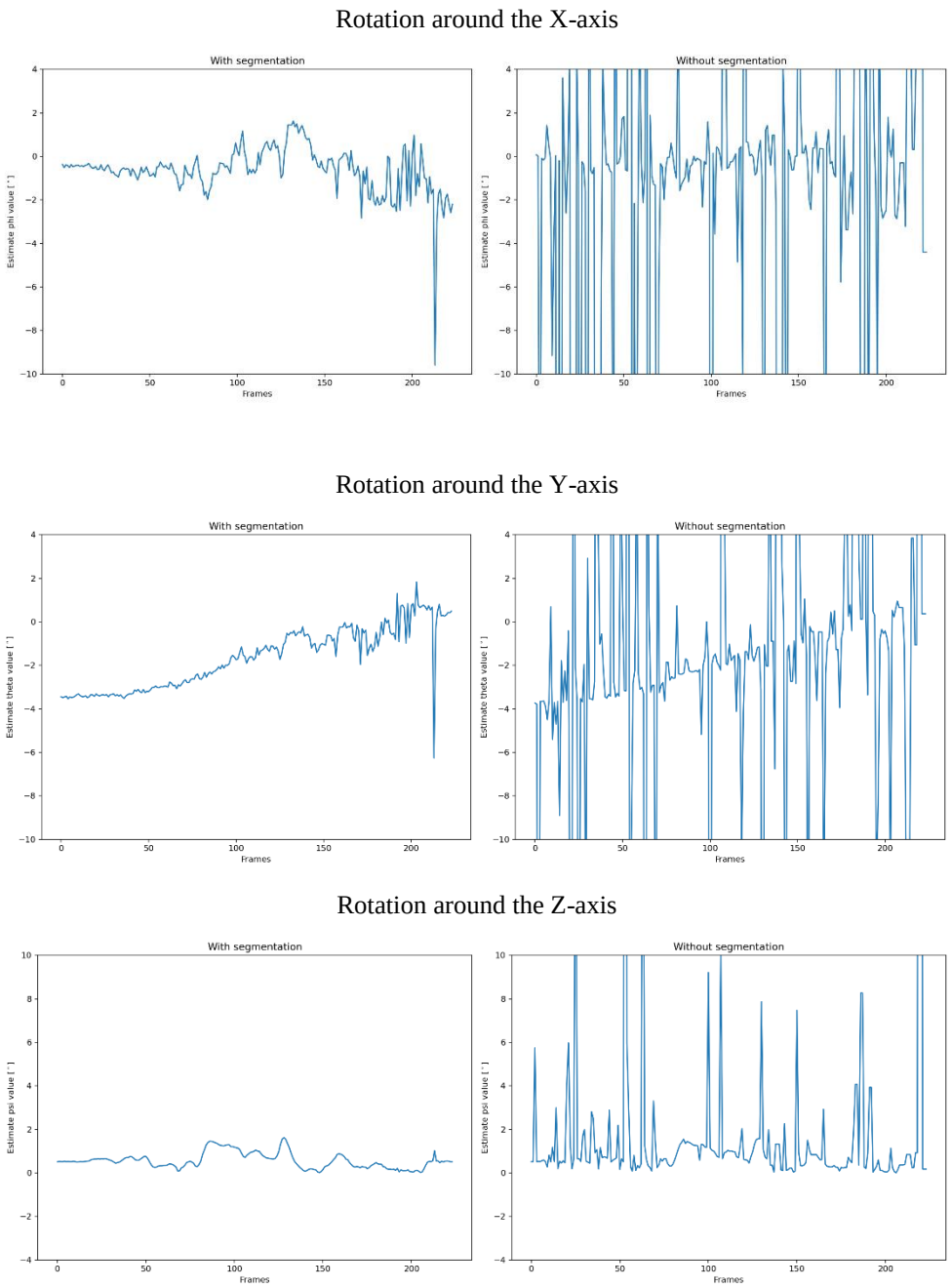

Fig. 7. The estimated attitude of the camera obtained from the video frames. The rotation results from the algorithm with segmentation (left) and the results for the algorithm without the proposed segmentation step (right).

The algorithm without the segmentation performs poorly in these conditions and the absolute positional and attitude precision decreases significantly. The algorithm with segmentation shows much more resistance toward these irregularities and the precision for this algorithm

stayed relatively constant, with small spikes at frames where the irregularities are present. Figure 6 and Figure 7 show the unfiltered data received from the algorithm.

The camera is moved 13 cm along the x-axis as can be seen in Figure 6 where the camera's translation in the x-direction can be seen to change from -14.5 cm to -27.5 cm. Non-linearities can be observed along the x- and y-axis. This is due to noisy movement and environment as a handheld camera is used. The z-axis does not experience any change as this is the axis that is used as a constant to investigate drift. Special care is taken to ensure this is kept at a constant distance. This was done by laying out the path on the surface of the object before the experiment and making sure that the path was clearly marked and at a known distance from the inspected object.

The camera experiences a slight rotation around the y-axis as can be seen in Figure 7. The rotation around the z-axis is kept at a constant angle to inspect the drift accumulated by the algorithm. The orientation output does not experience drift around the z-axis for the duration of the experiment but does experience up to 2° noise. The results from the translation and rotation estimation show that the segmentation part of the algorithm is essential. The segmentation is performed before the features are extracted and matched, and does not only improve the speed at which it executes, but also increase the robustness of the algorithm significantly.

5 Conclusion

A markerless vision-based localisation algorithm is shown where it is successfully implemented and tested. The algorithm does not make use of previous information about orientation to produce results. The algorithm is tested on images of known orientations to calculate the absolute error experienced by the algorithm. The tests show promising results with an absolute positional accuracy of 2.3 cm on average and an average attitude accuracy of 3.7˚. The accuracy performs well compared to DGPS, which has an absolute accuracy of up to 1.5 cm. The execution time for the algorithm can be sped up by making use of segmentation and can produce a result up to 4 times a second. The results from the handheld camera showed that making use of the segmentation step before localisation is done, is essential to ensure that the data received from the algorithm is useful. The noise levels of the camera can be seen by observing the translation on the Z-axis as this was constant throughout the experiment. The results showed minimal noise for the segmentation algorithm on the Z-axis with an error of less than 1 cm, with the exception of one frame. A higher-end camera is needed to ensure that the results from the algorithm are accurate. This algorithm in combination with an IMU, which typically has an accuracy of up to 30cm after 10 seconds, shows potential for accurate navigation in areas where GPS is not available.

The SIFT algorithm provides features that are very robust against scale and viewing angle. The algorithm, however, is not commonly used in localisation problems due to the execution time of the algorithm. The method proposed by this paper provides a possible solution to the problem. This is achieved by reducing the area the algorithm should extract features from and the number of features to match. The preliminary results show that the process can be sped up without compromising accuracy and could even result in small improvements to accuracy. This solution works well when an object is repeatedly inspected, and where simultaneous mapping and localisation are not necessary.

The proof of concept for this approach is shown in the paper. In future work, the algorithm will be tested on larger inspection objects to see if the same results can be duplicated. A structure-of-motion algorithm will also be investigated to reduce the human input and allow for a more autonomous approach to calculating the camera orientation in the mapping phase.

By making use of a structure-from-motion algorithm the number of features to be used during the localisation phase will increase due to more images per view.

References

1. Y. Wu, F. Tang and H. Li, *Image Based Camera Localization: An Overview*, (2018).
2. S. A. K. Tareen and Z. Saleem, *A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,* International Conference on computing, iCoMET, (2018).
3. A. M. Barros, M. Michel. Y. Moline, G. Corre, and F. Carrel, A *Comprehensive Survey of Visual SLAM Algorithms,* Robotics, **11**, 24 (2022)
4. S. J. Lee, D. Kim, S. S. Hwang and D. Lee, *Local to Global: Efficient Visual Localization for a Monocular Camera*, 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), 2230-2239, (2021).
5. Z. Xiao, K. Jian, S. Xie, T. Wen, C. Yu and D. Yang, *Monocular Vehicle Self localization method based on Compact Semantic Map*, (2018).
6. D. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, **60,** 91-110, (2004).
7. N. Todd, *Camera Calibration and Fundamental Matrix Estimation*, https://medium.com/build-more/camera-calibration-and-fundamental-matrix-estimation-with-ransac-2bee8ea18930, (2019).
8. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, **2,** (2004).
9. P. Rousseeuw, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, Journal of Computational and Applied Mathematics, **20**, 53-65, (1987).
10. J. Redmon, S. Divvala and R. Girshick, *You Only Look Once: Unified, Real-Time Object Detection*, IEEE Conference on Computer Vision and Pattern Recognition, 779-788, (2016).
11. V. Lepetit, F. Moreno-Noguer and P. Fua, *EPnP: An accurate O(n) solution to the PnP problem*, International Journal of Computer Vision, **81**, 155-166, (2009).