IMAV2016-41
http://www.imavs.org/pdf/imav.2016.41

Towards a real-time aerial image mosaicing solution

Alexander Kern[*], Markus Bobbe and Ulf Bestmann
TU Braunschweig, Germany

ABSTRACT

Real-time aerial image mosaicing is a crucial task for future search and rescue missions. Solving the correspondence problem, estimating a valid transformation and visualizing the result is computational intensive. It becomes more challenging if the flying platform is a small micro air vehicle (MAV), which limits the available margin for payload significantly.

This paper proposes a robust algorithm that is able to create and update photo maps in a fixed period of time. The approach uses a high number of features and strict match filtering to allow robust matching without additional sensor information. Subsequently the ability of todays most common single board computers (SBC) to run the presented algorithm is examined. Together with the selection of a lightweight board camera the setup is less than 100 grams allowing even small MAVs to generate maps in real-time.

1 Introduction

In recent years the development of unmanned aircrafts considerably reduced the costs and effort required to generate aerial images. As a result aerial mapping is getting more common in various fields such as agriculture or construction side documentation. Common mapping results are 2D pseudo-orthophotos and 3D surface meshes. Both are commonly generated using structure from motion or photogrammetry workflows which are started after the survey flight and usually take several hours.

The capability for cost and time efficient aerial mapping is also needed by various emergency response units. In civil catastrophe scenarios for instance after earthquakes or tsunamis these maps can be used by rescue workers to get an overview of the situation. While a 3D mesh of the area generated by structure from motion or photogrammetry can fulfill the need for an all around perspective at best, it is computationally intensive and therefore too time consuming. Certainly time is of critical importance to maintain an effective coordination of the emergency aid to save human lifes.

This paper aims to provide a solution for a 2D real-time mapping implementation. In its first half a suitable mosaicing algorithm is discussed and presented. Afterwards the hardware requirements are determined and compared to hardware available today. The ultimate goal is to create a lightweight standalone package capable of performing the image acquisition and mapping task independently from the aerial vehicle it is attached to.

[*]contact: al.kern@tu-braunschweig.de

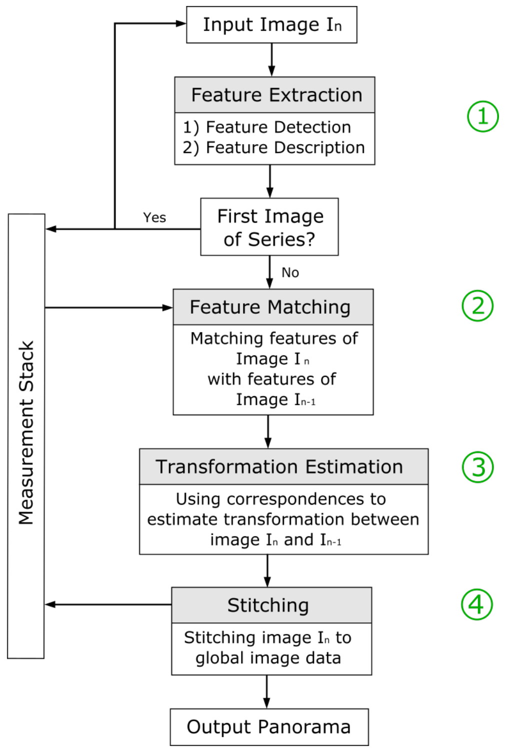
Input Image I_n
Feature Extraction
1) Feature Detection
2) Feature Description
First Image of Series?
Yes
No
Feature Matching
Matching features of Image I_n with features of Image I_{n-1}
Transformation Estimation
Using correspondences to estimate transformation between image I_n and I_{n-1}
Stitching
Stitching image I_n to global image data
Output Panorama
Measurement Stack
1
2
3
4

Figure 1: General approach to image mosaicing.

2 Real-time capable mosaicing algorithm

In this section the focus lies on the algorithms used for image mosaicing. The choice is highly influenced by the requirements emerging from real-time performance. While classic approaches use bundle adjustment to minimize the reprojection error over all images in a single, iterative step to get a global consistent solution, it is not feasible for real-time applications. The computational time is too high and will increase with every image taken as the complexity increases with $O((m+n)^3)$, where m is the number of images and n is the number of structure points [1]. The real-time performance imposes the requirement of a constant maximum time frame for every stitching iteration. This time frame has to be independent from the number of images that have already been taken. To allow such performance the extensive OpenCV 3.0 C++ library is well suited and is therefore used for this task.

2.1 Feature Extraction

In figure 1 the general approach for image mosaicing is displayed. As soon as a new image is acquired and sent to the pipeline, the first step is initiated. This step combines detection of markable points and description of the pixels near environment. For this task the ORB algorithm is used. It is based on the FAST keypoint detector and the BRIEF description of these keypoints [2]. The fact ORB uses a binary descriptor allows very efficient logical operations and comparisons, which fits the need for a real-time capable system excellently.

2.2 Feature Matching

If the input image is the first of the series it is then saved with all its features, since there is no other data it can be stitched to. After the acquisition and feature extraction of the next image the search for corresponding points is initiated. To ensure a fixed time frame the current input image is only stitched to the previous, thereby matches between these two sets of features have to be found. It is the most crucial part as it influences the final transformation notably. After generating a list of matches by brute force comparing their descriptor it is therefore beneficial to filter these results. A common method to identify good matches is the ratio test [3]. It compares the best and the second best match for each feature by their euclidean distance. A good match has a high chance of not having a second best match too close, which would be an indicator for a homogenous image section.

Additionally the previously introduced characteristic of ORB as a binary descriptor is used. It describes the features near environment with 1 and 0. Comparing two descriptors bit by bit and quantifying the difference by the hamming distance a proposition about the likeliness of two features can be done. That way the set of raw matches can be reduced to a set of best matches. Figure 2 shows the movement of the matches from two consecutive frames. On the top the movement of the raw matches is displayed, whereas on the bottom the filtered matches can be seen. In figure 3 the resulting stitched scene based on 43 aerial images is displayed. It should be noted, that reducing the matches drastically can result in decreasing transformation stability despite their quality, as they may not be uniformly spread across the image leaving more degrees of freedom. A good compromise between quality and quantity of matches should therefore be intended and can be achieved by selecting proper parameters for the pipeline depending mainly on the processing power and the overall performance requirements.

To improve the overall stability of the algorithm by minimizing the total drifting error it is furthermore beneficial to use an initial guess of the position to compute image neighboorhood relations. This can be done by kalman or particle filtering, additional sensor data like GPS or simple linear extrapolation of image centroids. If neighbours can be found within a defined range, matched features with the previous frame can be reprojected into the neighboring images to increase the number of observations and therefore ultimately improve transformations. At this step advanced state-of-the-art algorithms perform local bundle adjustment to optimize camera pose and 3D world points for even higher accuracy [4]. Considering the increasing complexity accompanied with numeric optimization a simpler approach for our system was chosen. The neighbourhood relations are used to compute additional matches between more than two frames resulting in reduced error propagation, which is evaluated in chapter 4.

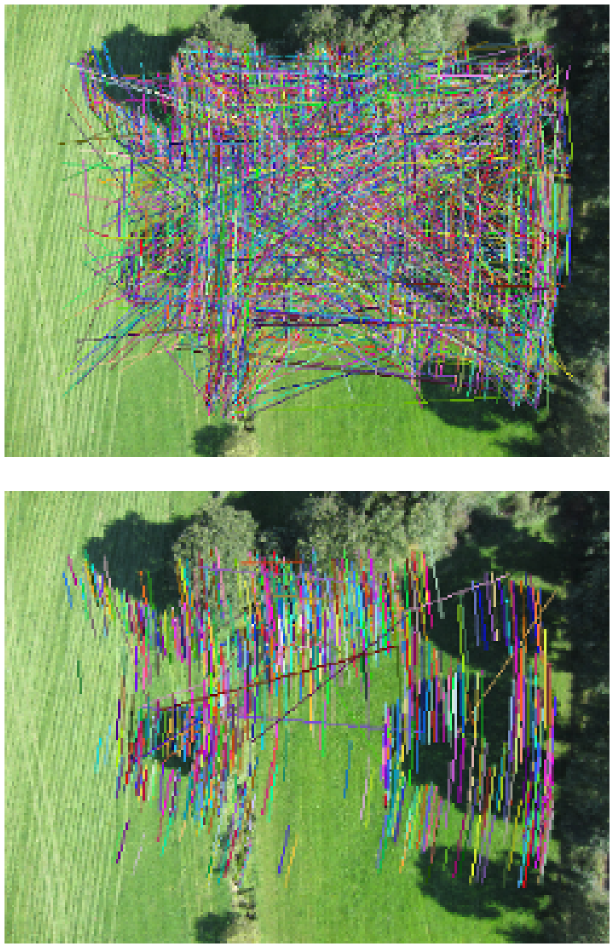
Figure 2: Movement of the matched features from two consecutive frames with unfiltered matches on the top and filtered by ratio test and hamming distance on the bottom.

2.3 Transformation estimation

After generating features and determining their corresponding points the transformation estimation starts. The most common method for real-time capable mosaicing approaches is the identification of the homography. It describes the relative position of two planes by a 3x3 matrix with 8 degrees of freedom, called perspective transformation. How-

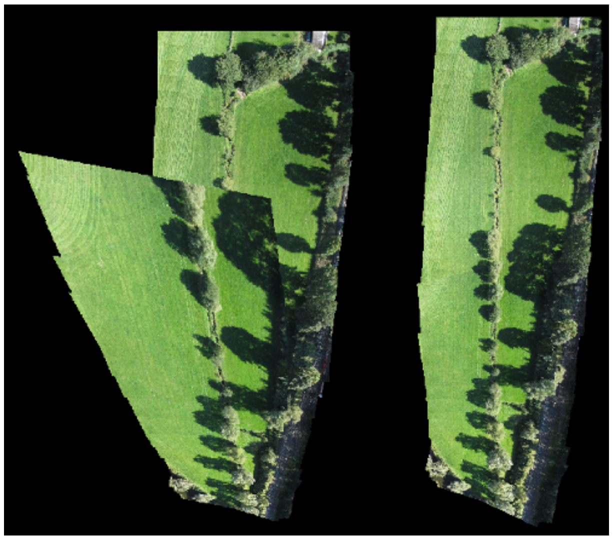

Figure 3: Stitched result with unfiltered matches on the left and with ratio and hamming filtering on the right for a test dataset.

ever in practice it has proven to be more reliable to use pitch and roll stabilised image data for mapping, reducing the overall complexity of the homography to

$$H = \begin{bmatrix} A & t \\ O^T & 1 \end{bmatrix} \tag{1}$$

where A represents a 1D rotation and scaling matrix and t the translation vector. Thus with 3 corresponding points the similiar transformation can be identified. It is important to be aware, that while estimating the homography the minimal geometric error between input and reference image is calculated. However the reference image in the pipeline is always the last stitched image. To achieve global consistent solutions it is therefore convenient to first transform all features used for calculation into the reference coordinate system and then compute the homography [5]. That way the input image is aligned relatively to the global reference and not to the untransformed previous image. Additionally random sampling consistency (RANSAC) is applied to reduce the number of outliers and obtain the maximum transformation accuracy during the process.

2.4 Image composition

In case a valid transformation matrix was found the input image must then be composited visually with the rest of the data. This step contains a high risk of breaking the real-time requirement, as the growing global map must be updated in a fixed time frame. Therefore only the section of the image that actually changed has to be considered. To achieve this a common approach is to project the edges of the input image into the global reference system using the previously estimated homography. In conclusion the maximum dimension of the warped input image is known and can be reduced to a region of interest for the visualization.

Additionally to increase performance even more the visualization process can be decoupled from the rest of the pipeline in aspects of image resolution. While working on lower resolution data for fast but stable feature extraction and matching the calculated matrix can be scaled afterwards. This approach also implies sending only raw image data and the corresponding transformation informations to the user. Once received, the frame can be aligned in high resolution to the global reference by using the complete processing power of the ground station. Furthermore implementing it that way, lost data packages during uplink to the UAV will not be a relevant problem as the solution on-board stays consistent the whole time.

3 Hardware selection

In the previous section a potential real-time capable image mosaicing algorithm was descriped. The next step is to choose a lightweight hardware that is able to satisfy the defined requirements.

3.1 SBC Selection

The performance of the proposed algorithm is mainy influenced by two parameters. First is the image resolution, second the number of features that are extracted. However these parameters are not independent from each other. A high resolution image has a lot of details that can be detected as markable points. The higher the resolution, the more markable points and the more features can be extracted. To ensure matching features it is therefore necessary to extract as many markable points as possible while keeping the computational time low. This is aggravated by the fact that during matching brute force is used. In conclusion every feature is compared with each other resulting in an $O(n^2)$ complexity.

Following the approach given by [6] the selection of a sufficently capable single board computer (SBC) is now possible. Their framework quantifies the performance of the developers computer in OpenCV by running two different standard algorithms and measuring their respective processing time T_1 and T_2. The first is a simple $(Complexity C_1 = 0\%)$, the second a complex one $(C_2 = 100\%)$. By interpolating linear between these two sampling points the user is able to identify the complexity C_{alg} of his own algorithm with

$$C_{alg} = \frac{T_{alg} - T_1}{T_2 - T_1} C_2 + C_1 \tag{2}$$

In the next step [6] tested common SBCs with both algorithms defining T_1 and T_2 on common SBC hardware. With a known C_{alg} on the developers computer T_{alg} on the SBC can then be estimated using

$$T'_{alg} = \frac{C_{alg} - C_1}{C_2} (T'_2 - T'_1) + T'_1 \tag{3}$$

Table 1 shows the resulting processing performance of the proposed mosaicing algorithm with a working resolution of 480 x 360 pixels and roughly 500 extracted features per image on different SBCs. Furthermore table 2 shows the specifications of the boards allowing to choose the best matching component concerning size and weight. Overall the Brix board is the best compromise by performance and weight. It also outperforms the Intel NUC while being less than half as heavy. However considering the goal of a very lightweight setup the Odroid XU3 offers the best solution. In our finals system an Odroid XU4 was selected as it comes with the same processing hardware but less periphery.

Board	Estimated fps
ITX i7	31.3
Brix	53.5
NUC	43.3
Odroid XU3	12.3
Odroid U3	8.1
ITX atom	9.3
Jetson	6.6
Rasp. Pi B	0.6

Table 1: Processing performance for the proposed algorithm with different SBCs [7].

3.2 *Camera specifications*

At first it is important to determine the general mission requirements. In an emergency aid scenario a human should be indentifiable on the aerial images. Therefore a resolution of 5 pixels per 30 cm should at least be maintained resulting in a ground sampling distance (GSD) of 6 cm / pixel. Taking the image resolution defined in section 3.1 into account this leads to an image ground dimension of 28.8 x 21.6 m. The vehicle operation altitude can vary but was chosen to be at least 30 m to prevent collisions with high trees or other obsticles.

Figure 4: UI-1221LE camera developed by IDS.

For a lightweight setup the use of board cameras is a promising option. In figure 4 the μEye UI-1221LE developed by IDS is displayed. With a resolution of 0.36 Megapixel, a maximum of 87.2 fps and global shutter it is a suitable camera for this mission. Following the definition of the GSD

$$GSD = h_{rel} \frac{d}{f\, n_{Pixel}} \qquad (4)$$

with the relative altitude h_{rel} above ground, d as the camera chip width and n_{Pixel} as image width the focal length of the camera lens can be deduced. Applying h_{rel} = 30 m, d = 4.52 mm, n_{Pixel} = 480 pixels and GSD = 6 cm / pixel the focal length of the camera is estimated with equation 4 to

$$f = h_{rel} \frac{d}{GSD\, n_{Pixel}} = 4.7mm \qquad (5)$$

Finally to guarantee a stable stitching process a high overlap between two consecutive frames is recommendable. Tests showed that an overlap of at least 75% is required for a stable stiching process.

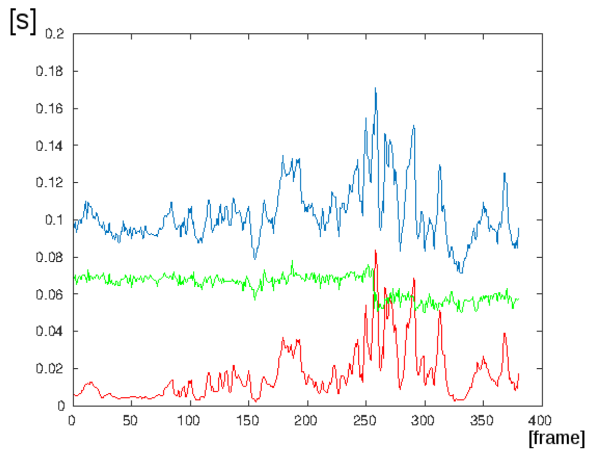

Figure 5: Processing time distribution for each image with image loading (green), feature detection (red) and total processing time (blue)

4 Evaluation

The proposed algorithm was tested on the determinded hardware using an available dataset. At first the performance of the SBC is analyzed and compared with the estimation made in the previous section. Subsequently the accuracy of the calculated trajectory is evaluated using a GPS reference. The data set used was published by [8] in 2016 and includes 381 images of a village captured by a UAV in an height of 165 m. Following section 3.2 the image resolution should be at least 480x360 pixels with a height of 30 m and 75% overlap. The latter parameters are overly fulfilled by the data set, which results in higher stability of the algorithm with less ground resolution. This can not be transferred directly to the

Name	Processor	Memory	Weight [gram]	Power@100% [Watt]	Volume[cm^3]
mini-ITX I7	Intel i7-4770S	16GB	684	68	1815
Brix	Intel i7-4500	8GB	172	26	261
NUC	Intel i5-4250U	8GB	550	20	661
Odroid XU3	Samsung Exynos 5422	2GB	70	11	131
Odroid U3	Samsung Exynos 4412	2GB	52	7	79
mini-ITX atom	Intel Atom D2500	8GB	427	24	1270
Jetson	Cortex A15	2GB	185	13	573
Rasp. Pi B	ARM1176JZF-S	512MB	69	4	95

Table 2: Processing performance for the proposed algorithm with different SBCs [7].

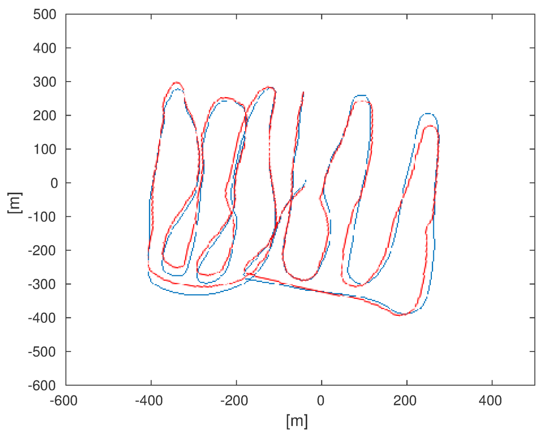

Figure 6: Trajectory generated by the GPS receiver (blue) and the stitching algorithm (red) with additional neighbour frames

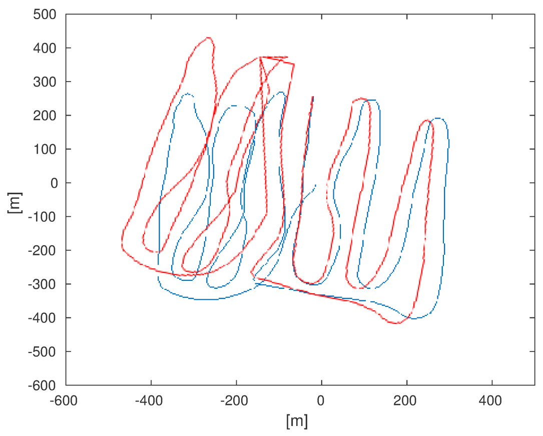

Figure 7: Trajectory generated by the GPS receiver (blue) and the stitching algorithm (red) without additional neighbour frames

mission requirements, as the resolution is not enough to identify humans safely. But it should be sufficent to verify the general performance and accuracy capabilities. Additionally the resolution can be adjusted in the real scenario as proposed in section 2.4 for visualization purposes.

4.1 *SBC Performance*

The extrapolations made in section 3.1 indicate the Odroid XU4 to run the stitching pipeline at 12.3 FPS. Using the given data set a mean runtime of 103 ms per frame was achieved. This concludes 9.7 FPS for the designed system making it slightly slower than estimated. The processing time for each image is displayed in figure 5. It can be noticed that image reading from the harddrive (green) shapes the mean processing time significantly, while feature matching (red) defines the overall variance. Despite the small variations around the mean value the complexity can be identified as constant per frame.

4.2 *GPS - Image trajectory comparison*

To evaluate the accuracy of the transformations and therefore the quality of the mapping process a comparison between UAV trajectories will be displayd. The trajectory produced by the proposed algorithm was analyzed by following the x- and y-coordinates of image centroids in the global reference frame. The scale was extracted by identifying markable centroids from the images and measuring their distance in satellite images. Calculating the meters per pixel and applying this informations to the rest of the data produced the red output in figure 6. The GPS trajectory measured by the UAV (standalone, single frequency receiver) on the other hand is displayed in blue. Figure 7 in contrast shows the calculated trajectory for the same data set without additional neighbour frame matching. It can be noticed, that the overall visual consistency only fits the first leg flown. Even though an alignment of the trajectories was found visually by assuming the error to be minimal at start, the error propagation of the transformation estimation obviously grows and affects the mapping solution negatively. However for future analysis another possibility might be to align the starting points, calculate a least squares transformation for the first centroids and applying this to the rest of the data. That way error propagation can be directly calculated between every single measurement, allowing a more distinct analysis.

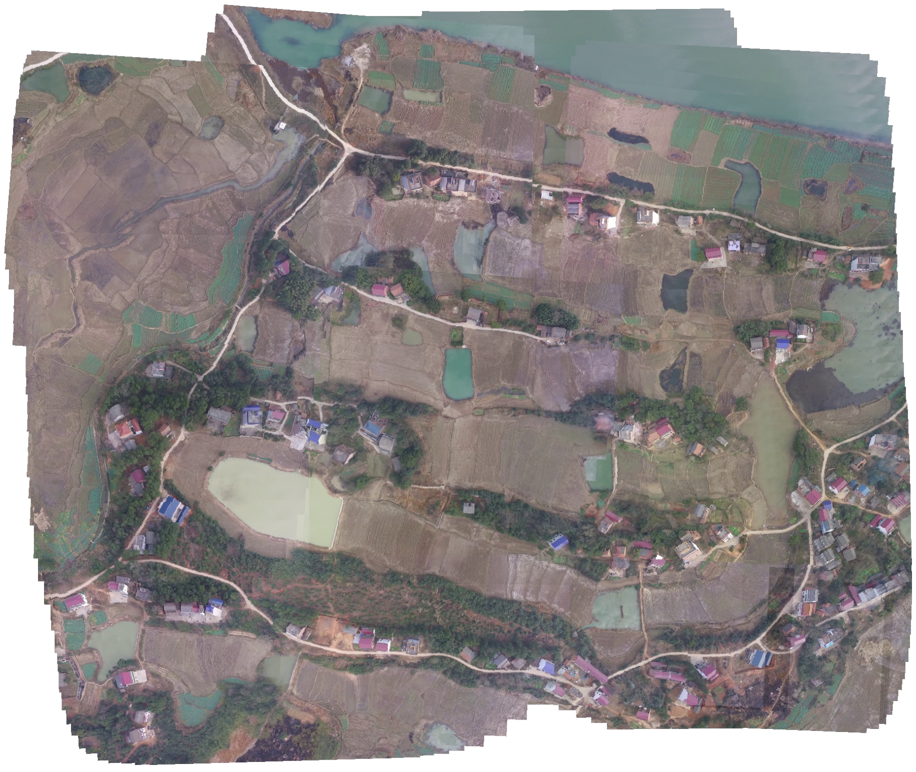

Figure 8: Public image sequence data set visualized by the proposed stitching algorithm

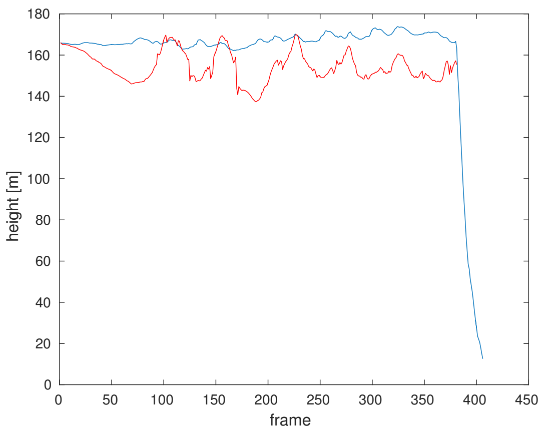

Figure 9: Height measurements by GPS (blue) and stitching algorithm (red)

Additional analysis can be achieved by decomposing the homography into rotation, scale and translation. Subsequently by multiplying the resulting scale with the initial UAV height of 165 m a direct comparison to the GPS height measurements can be done. Figure 9 shows this comparison. On the left the total height measured by image processing is displayed in red while GPS data is in blue. Significant is the sinusoidally characteristic of the plot. This is also an indication for the reduced error propagation achieved by neighbour frame matching. The first leg flown by the UAV ends at frame 77 followed by the second leg until frame 105. In this exact period the error drops down to zero. This repeats constantly following the flight routine of the UAV with a steady growing error offset.

5 Conclusion

A lightweight setup for image mosaicing was introduced together with a real-time capable stitching algorithm. The feature based approach solves the problem using only image data and no additional sensor informations. Tests with a public dataset showed promising results reducing the error propagation through homography estimation by extracting high amounts of markable points and filtering the matches afterwards.

The evaluation of available SBCs revealed that the Odroid XU4 is capable of running the pipeline with an average of 9.7 FPS. In combination with a board camera like the UI-1221LE the whole setup is light and can further be developed to a standalone module making real-time mapping available for a variety of MAVs. Final flight tests with the defined setup will show if an additional camera stabilization is required and are planned for the very near future.

REFERENCES

[1] Kaushik Mitra and Rama Chellappa. A scalable projective bundle adjustment algorithm using the l infinty norm. *Sixth Indian Conference on Computer Vision, Graphics Image Processing*, 2008.

[2] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[4] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[5] Taygun Kekec, Alper Yildrim, and Mustafa Unel. A new approach to real-time mosaicing of aerial images. *Robotics and Autonomous Systems*, 62:1755–1767, 2014.

[6] Dries Hulens, Jon Verbeke, and Toon Goedem. How to choose the best embedded processing platform for on-board uav image processing? *VISSAPP 2015*, 2015.

[7] Dries Hulens. Embedded processing board selection tool. www.eavise.be/hulens/selectiontool.html, 2016.

[8] Gang Wan Zhenbao Liu Shuhui Bu, Yong Zhao. Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam. 2016.