Mapping on the Fly: Real-Time 3D Dense Reconstruction, Digital Surface Map and Incremental Orthomosaic Generation for Unmanned Aerial Vehicles

Timo Hinzmann, Johannes L. Schönberger, Marc Pollefeys, and Roland Siegwart

Abstract The reduced operational cost and increased robustness of unmanned aerial vehicles has made them a ubiquitous tool in the commercial, industrial and scientific sector. Especially the ability to map and surveil a large area in a short amount of time makes them interesting for various applications. Generating a map in real-time is essential for first response teams in disaster scenarios such as, e.g. earthquakes, floods, or avalanches or may help other UAVs to localize without the need of Global Navigation Satellite Systems. For this application, we implemented a mapping framework that incrementally generates a dense georeferenced 3D point cloud, a digital surface model, and an orthomosaic and we support our design choices with respect to computational costs and its performance in diverse terrain. For accurate estimation of the camera poses, we employ a cost-efficient sensor setup consisting of a monocular visual-inertial camera rig as well as a Global Positioning System receiver, which we fuse using an incremental smoothing algorithm. We validate our mapping framework on a synthetic dataset embedded in a hardware-in-the-loop environment and in a real-world experiment using a fixed-wing UAV. Finally, we show that our framework outperforms existing orthomosaic generation methods by an order of magnitude in terms of timing, making real-time reconstruction and orthomosaic generation feasible onboard of unmanned aerial vehicles.

Timo Hinzmann
Autonomous Systems Lab, ETH Zurich, Switzerland,
e-mail: hitimo@ethz.ch

Johannes L. Schönberger
Computer Vision and Geometry Group, ETH Zurich, Switzerland,
e-mail: jsch@inf.ethz.ch

Marc Pollefeys
Computer Vision and Geometry Group, ETH Zurich, Switzerland,
e-mail: pomarc@inf.ethz.ch

Roland Siegwart
Autonomous Systems Lab, ETH Zurich, Switzerland,
e-mail: rsiegwart@ethz.ch

Supplementary Material

Video of the experiments:
www.timohinzmann.com/videos/fsr2017
Source code for map generation:
www.github.com/ethz-asl/aerial_mapper

1 Introduction

A fast and precise overview of an area is important for first aid teams in disaster scenarios such as earthquakes, floods, or avalanches. In particular, digital surface models (DSM) and orthomosaics are essential tools to support the human operator in quick decision-making. An orthomosaic gives a broad overview of the surroundings and helps the human operator to find regions of interest. Furthermore, orthomosaics enable every agent with a camera to infer its own absolute pose by employing feature extraction or image matching. The orthomosaic can therefore be used to localize the robot and other unmanned aerial vehicles (UAVs) while solely relying on an image stream [1]. An orthomosaic image is obtained by correcting aerial images for perspective and camera distortion using the information about the camera intrinsics and camera poses such that the generated image is true to scale and corresponds to a map projection throughout the image. The task of true orthorectification requires a three-dimensional model of the scenery. This is necessary in order to appropriately map intensities observed by the perspective camera to their location with respect to the orthographic camera. The DSM represents the three-dimensional model in form of a height map and furthermore helps to detect changes in elevation or to plan robot or human missions. The literature distinguishes between a DSM and a digital terrain model (DTM). The DSM includes the earth's surface and all objects such as buildings and trees on top of it. In contrast, the DTM models the bare earth's surface. In this publication, we are only interested in generating DSMs.

2 Related Work

The literature for creating overview images can be roughly categorized into panorama and mosaic generation where we utilize the distinction from [2, p. 12]: "Panorama is an extension of field of view (FOV) while mosaic is an extension of point of view (POV)". The mosaic generation can be divided into forward projection, using e.g. homographies or dense point clouds, and backward projection, using e.g. ray tracing in combination with grids or triangle meshes. An overview of the categories is given in Fig. 1. In this publication, we describe and compare a homography-based and point cloud-based forward projection, as well as a batch, and incremental grid-based backward projection approach by analyzing the advantages and disadvantages

in particular with respect to their real-time capabilities. All of the approaches above

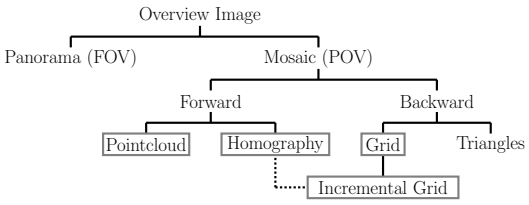


Fig. 1: Categories for generating an overview image. In this paper, we analyze a homography-based and point cloud-based forward projection, as well as a batch and incremental grid-based backward projection approach.

are incorporated in our end-to-end mapping framework (cf. Fig. 2) that tightly couples IMU odometry, GPS position and visual cues in a smoothing-based estimator and thus does not detach state estimation from orthomosaic generation. In summary, we claim the following contributions:

- A real-time incremental end-to-end dense reconstruction and orthomosaic generation framework for UAVs that tightly couples state estimation and seamless mosaic generation.
- Most importantly, we propose an incremental grid-based orthomosaic generation algorithm that is suitable for real-time applications in arbirary terrain by considering the surface model and best viewing angle. We validate its performance on a synthetic and real-world dataset with respect to homography-based, point cloud-based, and batch alternatives.
- We open-source our framework `aerial_mapper` consisting of all described DSM and orthomosaic generation approaches. Our framework augments the efficient and versatile `grid_map` library [3] with utilities for georeferenced mapping from aerial views.

2.1 Panorama Generation

Many approaches exist to generate a panoramic view given a set of images by applying a homography. Brown et al. presents in [4] an approach to robustly stitch a set of unordered images to a seamless panorama assuming rotations only around the optical axis. The main steps consist of feature extraction, matching in feature space, applying RANSAC and then computing the homography and applying bundle adjustment. Steedly et al. [5] build up on [4] and predict overlapping images more efficiently by utilizing the fact that the video stream is not unordered. Agarwala et al. [6] generate a multi-viewpoint panorama of a street using a homography and Markov Random Field (MRF) optimization. Laganière et al. [7] use homographies to generate bird-eye views for teleoperation of a robot. All of the approaches have in common that they focus on obtaining seamless and visually appealing panoramas

or bird-eye views and are not concerned about georeferencing or georeferencing errors. However, stitching using only feature correspondences leads to error accumulation and distorted maps when directly applied to UAVs as demonstrated e.g. in [8, p. 20]. The same is true when only the first image is georeferenced and the subsequent images are incrementally stitched to this reference image.

2.2 Mosaic Generation

2.2.1 Forward Projection

In UAV applications, where we are rather interested in generating a seamless and georeferenced mosaic, additional sensor measurements are used to obtain camera pose measurements or estimates: Hemerly et al. [9] describe the process of obtaining a single georeferenced image using a UAV. Olawale et al. [10] recover the camera intrinsics and extrinsics using GPS and manually collected ground control points in combination with the commercial photogrammetric software (*Agisoft*) and generate an orthomosaic. Yahyanejad et al. present in [8, 2] the results of homography-based image mosaicing from sensor data recorded on board of a rotary-wing UAV with a down-looking camera.

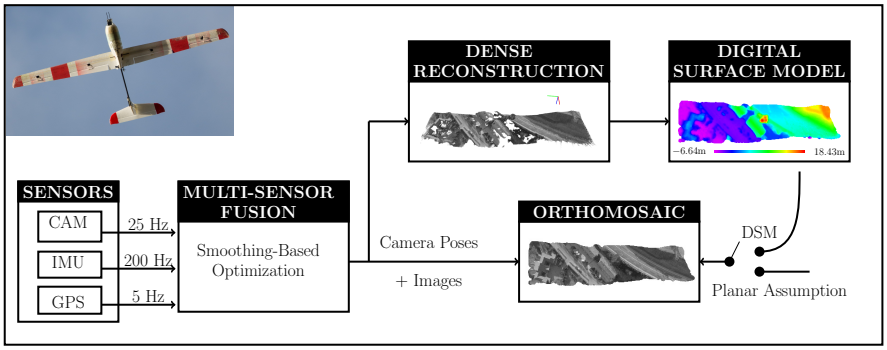


Fig. 2: System overview: The IMU, camera, and GPS measurements are fused in a smoothing-based optimizer. The optimized camera poses and images are used as input for the dense reconstruction and orthomosaic generation. The DSM is updated incrementally from the 3D dense georeferenced point cloud. The orthomosaic is computed via an incremental backward grid-based approach while employing the DSM or a planar assumption and considering the optimal viewing angle.

As presented, many approaches use a camera pose estimate and an image as input and then apply a robust but costly feature detection and matching algorithm. For instance, [8, 2] assume noisy IMU and GPS measurements and deal with this

by designing a quality function that finds a trade-off between geo-referencing error and seamless stitching. In contrast, we do not detach state estimation and orthomosaic generation but fuse GPS and IMU measurements as well as feature tracks in a consistent smoothing-based state estimation. The small offset between images in combination with gyroscope measurements enables fast and subpixel-accurate Lucas-Kanade feature tracking (KLT) [11]. The use of KLT is also supported by the findings in [12] claiming that KLT achieves the best quantitative results in the context of orthomosaic generation. Our philosophy is that accurate and efficient estimation of the camera poses is the backbone of consistent dense 3D reconstruction and seamless orthomosaic generation. Furthermore, the literature was previously not concerned about presenting runtime results and [12], [2] deplore lack of quantitative performance measures. We tackle this absence of information by presenting the runtime of all methods and an open-source Gazebo-based HIL environment [13] capable of generating synthetic datasets.

2.2.2 Backward Projection

Note that none of the homography-based forward projection approaches presented in the previous section employ a DSM as input. In contrast, in order to generate true orthomosaics, [14] employ triangle-based backprojection, also known as ray tracing, in combination with a DSM. Our backprojection approach is very similar to [14] but we utilize a grid of squares to simplify the raytracing process. Furthermore, we present a novel incremental grid-based orthomosaic generation approach to speed up the computation.

3 Methodology

The methodology section follows the data flow illustrated in Fig. 2: Sec. 3.1 presents the smoothing-based GPS-IMU-Vision fusion. Given the input images and corresponding optimized camera poses, a dense georeferenced point cloud can be generated using planar rectification, as demonstrated in Sec. 3.2. Sec. 3.3 presents how this dense point cloud can be used to generate a DSM by employing inverse distance weighting (IDW). Finally, Sec. 3.4 presents our approaches to generate an orthomosaic from a stream of images, optimized camera poses, and DSM using (a) forward projection and (b) backward projection.

3.1 Multi-Sensor Fusion

In this section, we present the core elements of our proposed multi-sensor fusion framework. We distinguish three coordinate systems: the global frame $\mathscr{F}_G$, the cam-

era frame $\mathscr{F}_C$, and the body frame $\mathscr{F}_B$. To avoid unnecessary conversions due to the vision-based fusion, we choose the Universal Transverse Mercator (UTM) coordinate system where $\mathbf{p}_B^G$ expresses easting, northing, and elevation. We seek to estimate the robot states $\mathbf{x}_R$ as well as the set of landmarks $\mathbf{x}_L$. We define the robot state as: $\mathbf{x_R} := \left[\mathbf{p}_B^G\ \mathbf{q}_B^G\ \mathbf{v}_B^G\ \mathbf{b}_a\ \mathbf{b}_g\right]$ where the orientation, position and velocity of the body frame expressed in global coordinates are denoted with $\mathbf{q}_B^G$, $\mathbf{p}_B^G$ and $\mathbf{v}_B^G$. The remaining state vector consists of accelerometer bias $\mathbf{b}_a$ and gyroscope bias $\mathbf{b}_g$.

3.1.1 Vision Front-End

FAST features [15] are extracted from every input image and tracked from frame to frame using KLT with subpixel refinement. To speed up the tracking process and to avoid outliers, we use the gyroscope of the IMU to predict the location of the pixel in the subsequent image. Furthermore, we employ feature bucketing to guarantee uniformly distributed features across the image for improved vision-based motion estimation.

3.1.2 Smoothing-Based State Estimation

For sensor fusion and pose estimation we use the incremental smoothing and mapping algorithm *iSAM2* [16]. For the employed reprojection residual, we refer to [17]. Every reprojection factor has a Cauchy M-Estimator associated with it to reduce the influence of outliers[1]. The IMU measurements are preintegrated and summarized in a single relative motion constraint connecting two time-consecutive poses as described in [18]. The residual and Jacobian of the GNSS position factor is calculated by "lifting" the residual: $\mathbf{e} = \tilde{\mathbf{t}}_B^G - \mathbf{t}_B^G$, $\frac{\partial \mathbf{e}}{\partial \delta t} = -\frac{\partial}{\partial \delta t}\left(\mathbf{t}_B^G + \mathbf{R}_B^G \delta t\right) = -\mathbf{R}_B^G$ where $\tilde{\mathbf{t}}_B^G$ is the measured position transformed to UTM coordinates [2]. All measurements are inserted into the factor graph once they become available. For every measurement, the factor graph is augmented by a state node. To estimate the initial position, orientation as well as accelerometer biases, at the beginning of every experiment, the plane is kept level for few seconds. During this time, the GPS position measurements are averaged to determine the initial position. The averaged accelerometer readings are used for coarse gravity alignment and bias estimation. After take-off is detected, the vision measurements are incorporated into the factor graph. Note that in this publication only open-loop SLAM was employed, i.e. no loop closures or inter-matches were included in the factor graph. Albeit we did not experience any inconsistencies in the generated dense reconstruction or orthomosaics we consider to integrate an online loop-closure or map-tracking module in future work to guarantee the global consistency of the map.

[1] The Cauchy weight is $k^2/(k^2+e^2)$, where e is the residual and k is a constant set to 3.0.

[2] Note that we neglect the translational offset between GNSS antenna and IMU since for our setup this corresponds to few centimeters.

3.2 Dense Reconstruction

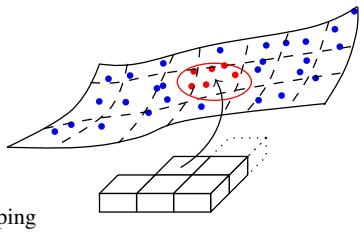
Given the optimized camera poses of our monocular camera rig, a virtual stereo-pair is generated using planar rectification [19]. The dense point cloud is then computed by applying efficient stereo block matching. Note that planar rectification assumes that the epipoles of a virtual stereo-pair are outside the field of view which is fulfilled for our fixed-wing UAV with down-looking camera due to the approximately fronto-parallel motion with respect to the ground.

3.3 Digital Surface Map Generation

The georeferenced dense point cloud serves as input for the digital surface map. The algorithm consists of a for-loop that iterates over all affected cells in the grid. A fast

Algorithm 1 Grid-Based DSM
p_r : Radius of the kd-tree used for interpolation.
λ : Factor to increase the interpolation radius.

```
1: function DSM(POINT CLOUD,
   CAMERA POSES)
2:     cells ←identifyAffectedCells(T_C^G)
3:     for c : cells do
4:         N ← {}
5:         while N == {} do
6:             p_r ← λ · p_r
7:             N ← kd-tree(x_c, y_c, p_r)
8:         end while
9:         Apply interpolation methods
10:        (Optional:) Height-to-color mapping
11:    end for
12: end function
```



DSM

kd-tree[3] implementation returns the set of nearest points N found within the interpolation radius p_r. Next, inverse distance weighting (IDW) is applied as interpolation method. IDW intuitively determines the cell's height by using a linearly weighted combination of the nearest neighbors, where the weight corresponds to the inverse distance to the cell center, thus giving higher weight to points that are closer to the cell center. An adaptive interpolation radius is utilized (cf. Alg. 1) that is guaranteed to return an interpolated height value in sparse regions and still keeps a high level of detail in dense regions.

[3] *nanoflann*: nano fast library for approximate nearest neighbors.

3.4 (Ortho-)Mosaic Generation

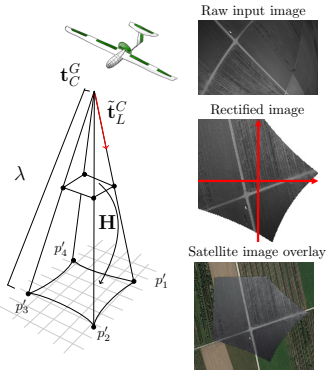
In this section, we present the implemented approaches for computing an (ortho-)mosaic while focusing on the proposed incremental grid-based orthomosaic generation.

3.4.1 Homography-Based Mosaic (Forward Projection)

A perspective homography $\mathbf{H}$ is computed which relates the border pixel coordinates of the image to points on the ground surface. The homography is then applied to the

Algorithm 2 Homography-Based Mosaic
w : Image width in pixel. h : Image height in pixel.

function MOSAICHOMOGRAPHY(IMAGE, CAMERA POSE, CAMERA INTRINSICS)
  $p_1 \leftarrow (0,0)$, $p_2 \leftarrow (w,0)$, $p_3 \leftarrow (w,h)$,$p_4 \leftarrow (0,h)$
  undistort(image)
  // Obtain ground points.
  for $i = 1:4$ **do**
    // Computing the scale.
    $\lambda_i \leftarrow -(z_C^G - h_{ground})/(\mathbf{R}_C^G \tilde{\mathbf{t}}_L^C)_z)$
    // Computing the ground position [UTM].
    $p_i' \leftarrow \mathbf{t}_C^G + \lambda_i \mathbf{R}_C^G \tilde{\mathbf{t}}_L^C$
  end for
  $\mathbf{H} \leftarrow$computeHomography$(\mathbf{p}, \mathbf{p}')$
  $image_{transf.} \leftarrow$ applyHomography$(H, image)$
  mosaic $\leftarrow$ iterativeBlending$(image_{transf.})$
end function



undistorted input image and the transformed single rectified image is blended with the overall mosaic using feathering. The pseudo code of the algorithm and the results are presented in Alg. 2 and Fig. 6, respectively.

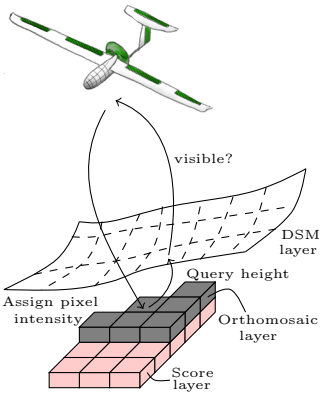
3.4.2 Grid-Based Orthomosaic (Backward Projection)

The grid-based orthomosaic generation in *batch* formulation iterates over all cells and, for every cell, queries the corresponding height from the DSM layer. An additional for-loop iterates over all images and, given the corresponding camera pose and camera intrinsics, checks if the cell is within the visible camera cone. Since every cell is usually observed from several camera frames, the question poses which is the ideal pixel intensity value to be assigned to the cell of the orthomosaic. Various mosaic strategies exist [14]. We propose to extract the pixel intensity from the image where the corresponding camera pose is the closest to nadir. This elevation angle is defined as the observation vector from the camera to the cell center. Instead of performing these operations on all cells in the grid, our proposed *incremental* for-

mulation (Alg. 3) identifies the subset of cells that needs to be updated, as illustrated in green in Fig. 6. The cells are identified by projecting the border-pixels of the current image onto the plane defined by the minimal elevation obtained from the DSM (cf. Alg. 2). All cells which are potentially visible given the current camera pose are obtained by min/max operation. Depending on the image distortion, a tighter approximation could be achieved using e.g. the Bresenham algorithm [20]. Only those cells which show a higher elevation angle than currently stored and which are visible from the current camera configuration are updated.

Algorithm 3 Incremental Grid-Based Orthom.

1: **function** INCREMENTALORTHOMOSAIC-GRID(IMAGE, CAMERA POSE, CAMERA INTRINSICS)
2: cells ←identifyAffectedCells(T_C^G)
3: **for** c : *cells* **do**
4: $z_c \leftarrow dsm(c)$
5: **if** visibility(x_c, y_c, z_c, T_C^G) **then**
6: score ← computeScore(x_c, y_c, z_c, T_C^G)
7: **if** score > score(c) **then**
8: (u, v) ← backproject($x_c, y_c, z_c, T_C^G, image$)
9: ortho(c) ←pixelIntensity($u, v, image$)
10: **end if**
11: **end if**
12: **end for**
13: **end function**



3.4.3 Point Cloud-Based Orthomosaic (Forward Projection)

In contrast to the approach described in Section 3.4.2 one can directly use the dense 3D reconstruction of the environment (cf. Section 3.2) to generate an orthomosaic view and hence avoid the costly backprojection step. The proposed point cloud-based orthomosaic generation approach closely follows Alg. 2 but instead of the height we compute the IDW of the pixel intensity.

4 Platform And Sensors

For our experiments, we use *Techpod* (cf. Fig. 2), a small unmanned research plane with a wingspan of 2.60 m. The IMU *ADIS16448*, and the grayscale camera *Aptina MT9V034* of the sensor pod are hardware-synchronized using a VI-Sensor [21] and run at 200 Hz and 25 Hz, respectively. The camera *Aptina MT9V034* has a focal length of 2.8 mm, a sensor diagonal of $1/3$ inch, and a resolution of 752×480 pixels.

The *ublox LEA-6H* GPS receiver and the pressure sensors are connected to the *Pixhawk* autopilot running a real-time EKF [22].

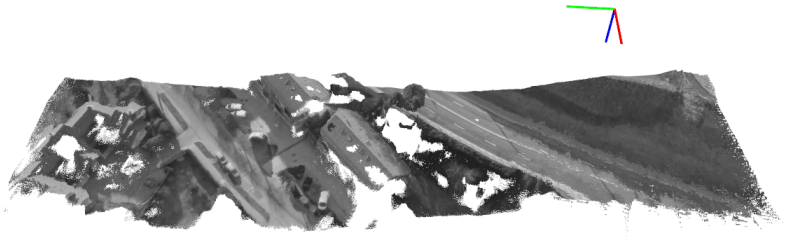
5 Simulation Experiments

The Gazebo-based HIL environment was used to validate the DSM and orthomosaic generation is illustrated in Fig. 3. The aerodynamic coefficients and mounted sensors closely model our UAV *Techpod*. Fig. 4 shows the results from a simulated single

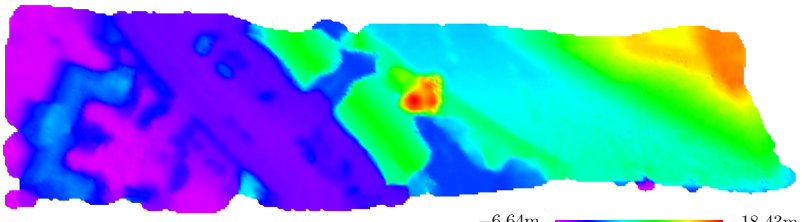
(a) Fixed-wing UAV and synthetic image. (b) Output of *QGroundControl* in HIL mode.

Fig. 3: Gazebo-based HIL environment for fixed-wing UAVs

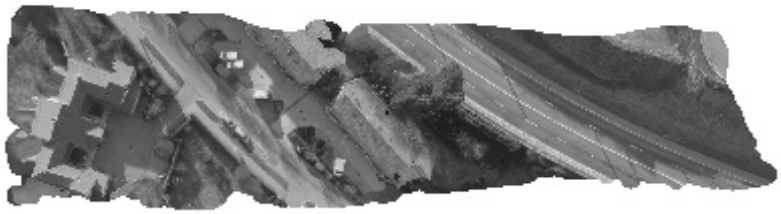
scan line at a relatively low altitude of 50 m above the mesh of *Pix4D*'s *cadastre* [23] dataset. For this experiment, 429 images are rendered at a frame rate of 20 Hz and each image is associated with the ground truth pose. Fig. 4(a) shows the coordinate system of the last camera pose and the point cloud generated by the planar rectification algorithm using every 10 th image. In this experiment, we deliberately do not use every frame for the dense reconstruction to underline the framework's potential to handle sparse regions or holes in the point cloud. The DSM layer, which is given in Fig. 4(b), is generated by applying IDW with an initial radius of 5 m to the dense point cloud. Fig. 4(c) depicts the incremental grid-based orthomosaic in which the pixel intensity is queried from the first camera that is in line of sight of the respective cell. In contrast, Fig. 4(d) shows the incremental grid-based orthomosaic where the pixel intensities are obtained from the camera frame with the view closest to nadir. The corresponding elevation angles between selected camera pose and orthomosaic cell are shown in Fig.4(f), (g). In particular in regions with a small altitude to terrain height ratio (e.g. tree in center) one can observe that the nadir-view approach renders an improved orthorectified view and avoids double object mapping. As Fig.4(e) illustrates, the result of our nadir-view approach is in accordance with the orthomosaic generated by *Pix4D*, for which we used the same georeferenced images as input. The homography approach is not shown since the underlying flat plane assumption results in the predicted large orthomosaic distortions.



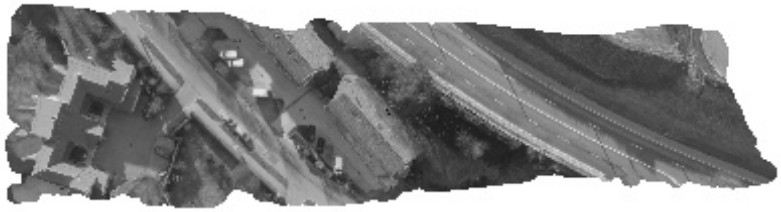
(a) Dense Reconstruction



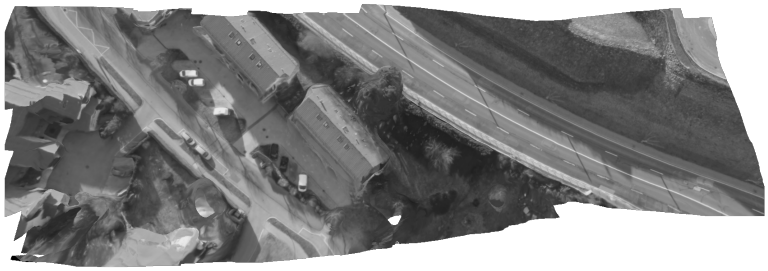
(b) DSM (grid resolution: 0.5 m)



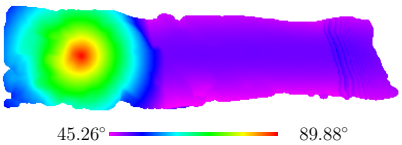
(c) Incremental Grid-Based Orthomosaic (First view; grid res.: 0.5 m)



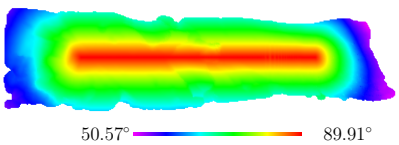
(d) Incremental Grid-Based Orthomosaic (Max. elevation angle; grid res.: 0.5 m)



(e) Pix4D Orthomosaic



(f) Observation angle (First view)



(g) Observation angle (Max. elevation angle)

Fig. 4: Simulation results for dataset *cadastre* [23].

6 Real-World Experiments

In this section, we present the results obtained from the semi-autonomous flight at an altitude of 100 m above ground (cf. Fig. 5). The dense point cloud and ortho-

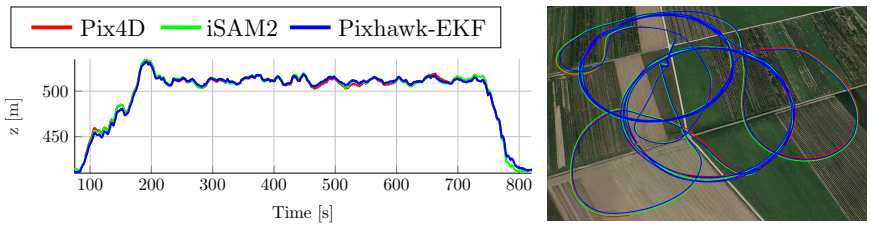
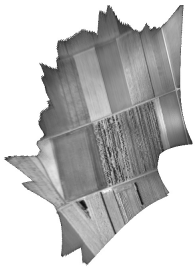
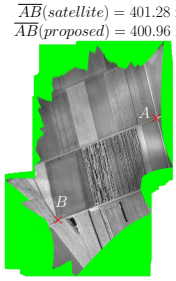


Fig. 5: Comparison of *Pix4D*, *Pixhawk*-EKF [22] and *iSAM2*-based estimation.

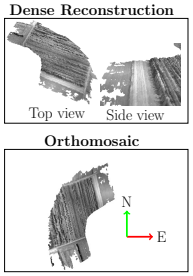
mosaics are presented in Fig. 6. In contrast to the previous experiment, we present the output of the incremental grid based on a flat DSM. Due to the high altitude to terrain height ratio in this experiment, the assumption does not introduce measurable orthomosaic inconsistencies with respect to *Google* imagery (cf. Fig. 6b). The homography-based orthomosaic with applied feathering, shown in Fig. 6a, can handle an image stream of up to 57 Hz. Given the measured runtime in Table 1, the combination of dense reconstruction and point cloud-based orthomosaic is even slightly faster than the homography-based approach. The caveat of the former is that, due to image distortions at the outer regions, a smaller field of view will be covered by the virtual stereo pair (cf. Fig. 6c). Both the homography- and point cloud-based approach outperform the methods presented in [2] by an order of magnitude. Note that the variants proposed in [2] do not generate a DSM and thus visual artifacts are introduced into the orthomosaic when the planar assumption is violated. The pro-



(a) Homography-based orthomosaic (Section 3.4.1)



(b) Incr. grid orthomosaic (Section 3.4.2, first view)



(c) Point cloud-based orthomosaic (Section 3.4.3)

Fig. 6: Mapping results based on the *iSAM2* state estimates using a fixed-wing UAV.

	Time/image	Total time	# images	Resol.	CPU	Type	Impl.
Homography (Sec. 3.4.1)	17.4 ms	4.33 s	249	-	2.8 GHz	Forw.	C++
Dense rec. (Sec. 3.2)	16.7 ms	0.384 s	23/249	-	2.8 GHz	Forw.	C++
Point cloud (Sec. 3.4.3)	0.2 ms	0.51 s	249	10 m	2.8 GHz	Forw.	C++
Point cloud (Sec. 3.4.3)	0.79 ms	1.97 s	249	1 m	2.8 GHz	Forw.	C++
Point cloud (Sec. 3.4.3)	31 ms	7.72 s	249	0.1 m	2.8 GHz	Forw.	C++
"Position" [2]	0.47 s	17.31 s	37	n/a	2.66 GHz	Forw.	Matlab
"Pose" [2]	0.5 s	18.33 s	37	n/a	2.66 GHz	Forw.	Matlab
"Image" [2]	12.41 s	459.2 s	37	n/a	2.66 GHz	Forw.	Matlab
"Hybrid" [2]	3.68 s	136.28 s	37	n/a	2.66 GHz	Forw.	Matlab
Grid (batch) (Sec. 3.4.2)	0.43 s	107.41 s	249	10 m	2.8 GHz	Backw.	C++
Grid (batch) (Sec. 3.4.2)	1.73 s	430.27 s	249	1 m	2.8 GHz	Backw.	C++
Grid (incr.) (Sec. 3.4.2)	171 ms	42.6 s	249	1 m	2.8 GHz	Backw.	C++
Triangle Mesh [14]	62 min	620 min	10	0.15 m	2.8 GHz	Backw.	C#, Matl.

Implemented Proposed

Table 1: Runtime results for dense reconstruction and orthomosaic generation.

posed incremental grid-based approach speeds up the computation by a factor of 10 compared to the batch variant. Both, the batch and incremental grid approach are several magnitudes faster than the triangle mesh implementation [14]. However, the implementation in [14] also performs color matching and identifies obscured pixels during the ray casting process adding up to a runtime of 62 minutes per image.

7 Conclusion

In this publication, we demonstrated that incremental end-to-end dense reconstruction and orthomosaic generation for UAVs is feasible in real-time allowing, for instance, advanced autonomous missions of UAV fleets by relying on orthomosaic-based localization only. We highlight the characteristics of our implemented orthomosaic generation approaches in particular with respect to runtime and the influence of the flight altitude to terrain height ratio: The advantage of homography-based orthomosaic generation is the seamless blending, the fast computation and the optimal integration of all pixels but is only suited for planar scenery or, alternatively, high flight altitudes. The benefit of the point cloud-based orthomosaic is the lowest computation time among the evaluated methods, the seamless blending and the direct way of considering the surface elevation. However, depending on the dense reconstruction algorithm the area of coverage is smaller and sparse regions can only be overcome by interpolating nearby point intensities potentially leading to incorrect orthomosaics. Our proposed backward incremental grid-based orthomosaic is suited for arbitrary terrain, renders a true orthomosaic by considering the surface model and optimal viewing angle and still achieves real-time performance.

Acknowledgements The research leading to these results has received funding from ArmaSuisse and the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°600958 (SHERPA). The authors thank Andreas Jäger and Sammy Omari for the implementation of the planar rectification algorithm, Lucas Pinto Teixeira for the synthetic image rendering pipeline, and Thomas J. Stastny for comments that greatly improved the publication.

References

1. A. Yol *et al.*, "Vision-based absolute localization for unmanned aerial vehicles," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3429–3434, Sept 2014.
2. S. Yahyanejad and B. Rinner, "A fast and mobile system for registration of low-altitude visual and thermal aerial images using multiple small-scale UAVs," *ISPRS J. of Photogrammetry and Remote Sensing*, vol. 104, pp. 189–202, 2015.
3. P. Fankhauser *et al.*, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS)*, vol. 1, ch. 5, Springer, 2016.
4. M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, pp. 59–73, Aug. 2007.
5. D. Steedly, C. Pal, and R. Szeliski, "Efficiently registering video into panoramic mosaics.," in *ICCV*, pp. 1300–1307, IEEE Computer Society, 2005.
6. A. Agarwala *et al.*, "Photographing long scenes with multi-viewpoint panoramas," *ACM Trans. Graph.*, vol. 25, pp. 853–861, July 2006.
7. R. Laganière, "Composing a birds eye view mosaic," *Vision Interface*, pp. pp. 382–386, 2000.
8. S. Yahyanejad, *Orthorectified Mosacking of Images from Small-scale Unmanned Aerial Vehicles*. PhD thesis, Alpen-Adria Universität Klagenfurt, 2013.
9. E. M. Hemerly, "Automatic Georeferencing of Images Acquired by UAV's," *Int. J. of Automation and Computing*, vol. 11, no. 4, pp. 347–352, 2014.
10. B. O. Olawale *et al.*, "A Four-Step Ortho-Rectification Procedure for Geo- Referencing Video Streams from a Low-Cost UAV," vol. 9, no. 8, pp. 1445–1452, 2015.
11. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pp. 674–679, 1981.
12. P. Azzari *et al.*, *An Evaluation Methodology for Image Mosaicing Algorithms*, pp. 89–100. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
13. F. Furrer *et al.*, *RotorS—A Modular Gazebo MAV Simulator Framework*, pp. 595–625. Cham: Springer International Publishing, 2016.
14. M. Nielsen, "True orthophoto generation," Master's thesis, Technical Univ. of Denmark, 2004.
15. Y. Biadgie and K. A. Sohn, "Feature detector using adaptive accelerated segment test," in *2014 Int. Conf. on Information Science Applications (ICISA)*, pp. 1–4, May 2014.
16. M. Kaess *et al.*, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering.," in *ICRA*, pp. 3281–3288, IEEE, 2011.
17. S. Leutenegger *et al.*, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," in *Proc. of Robotics: Science and Systems*, (Berlin, Germany), June 2013.
18. C. Forster *et al.*, "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation," in *Proc. of Robotics: Science and Systems*, (Rome, Italy), July 2015.
19. A. Fusiello *et al.*, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.
20. J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, pp. 25–30, Mar. 1965.
21. J. Nikolic *et al.*, "A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM," in *ICRA*, pp. 431–437, 2014.
22. S. Leutenegger *et al.*, "Robust state estimation for small unmanned airplanes," in *Control Applications (CCA), 2014 IEEE Conf. on*, pp. 1003–1010, Oct 2014.
23. "Pix4D dataset Cadastre." `https://support.pix4d.com/hc/en-us/articles/202561399-Example-Datasets-Available-for-Download-Cadastre-`.