2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Real-Time Photo-Realistic 3D Mapping for Micro Aerial Vehicles

Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys

Abstract— **We present a real-time photo-realistic 3D mapping framework for micro aerial vehicles (MAVs). RGBD images are generated from either stereo or structured light cameras, and fed into the processing pipeline. A visual odometry algorithm runs on-board the MAV. We improve the computational performance of the visual odometry by using the IMU readings to establish a 1-point RANSAC instead of using the standard 3-point RANSAC to estimate the relative motion between consecutive frames. We use local bundle adjustment to refine the pose estimates. At the same time, the MAV builds a 3D occupancy grid from range data, and transmits this grid together with images and pose estimates over a wireless network to a ground station. We propose a view-dependent projective texture mapping method that is used by the ground station to incrementally build a 3D textured occupancy grid over time. This map is both geometrically accurate and photo-realistic; the map provides real-time visual updates on the ground to a remote operator, and is used for path planning as well.**

I. INTRODUCTION

Real-time 3D textured reconstruction of a MAV's environment is useful in many ways. A geometrically-accurate and photo-realistic 3D map can be used for object recognition, localization, and path planning. At the same time, the map gives an operator in a far-away location better real-time situational awareness than live video feeds. In addition, the geometrical integrity of the map easily facilitates the addition of augmented reality. Texture-mapping out a MAV's environment in real-time is a challenging area of research as the MAV's payload constraint does not permit access to high-end GPS/INS systems that can provide high-fidelity pose estimates and facilitate seamless stitching of image and range data. Furthermore, existing 3D laser rangefinders are far too heavy for MAVs which often have to rely on depth-sensing cameras to perform 3D reconstruction.

We use a 3D occupancy grid as the basis of our 3D textured map as an occupancy grid is robust to sensor measurement noise, accounts for the uncertainty inherent in range measurements from stereo cameras, and explicitly models free space. Furthermore, the grid map can be used for path planning [1] on board our MAV platform as shown in Figure 1, allowing us to efficiently utilize the limited computational resources on the MAV. Such utilization is made efficient further by offloading the 3D map reconstruction task to a ground station. In this case, the ground station requires sets of image and depth data, and the pose from which each dataset was captured; the MAV transmits the required data wirelessly to the ground station.

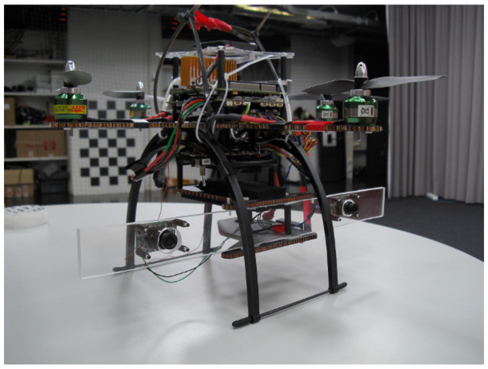


Fig. 1. The PIXHAWK quadrotor with a 25 cm-baseline stereo rig.

The MAV estimates its pose via a visual odometry approach which uses the RANSAC algorithm [2]. In the standard case, we use 3-point RANSAC to find the relative pose between consecutive frames. By using IMU measurements, we establish a 1-point RANSAC which significantly speeds up the visual odometry. We then use local bundle adjustment to refine the pose estimates.

On the MAV, depth information from a RGBD image which is obtained from either a stereo camera or Kinect device is used to build a local occupancy grid. The mapping module merges this grid with an existing global grid which is used by the path planner, and at the same time, the local grid is sent together with the corresponding RGB image and pose estimate to the ground station. On the ground station, the local grid is merged with the stored global grid. We acknowledge that the same map merging process is done two times: one on the MAV, and one on the ground station; since a global grid grows indefinitely, the global grid is too large to be transmitted wirelessly, and transmission of local grid data is a far more optimal use of wireless bandwidth. Still, the global grids maintained by the MAV and ground station are assumed to be identical. For each facet of a voxel in the occupancy grid, we determine which images the facet is visible in. Subsequently, we split the facets that are partially visible in one of the images such that the resulting subfacets are either visible or not visible in each image. The reason for splitting the facets into subfacets is to enforce the constraint that each subfacet is either wholly visible or wholly invisible in any image. Visibility is determined by a GPU-based depth mapping algorithm which identifies visible subfacets of all voxels in each camera viewpoint; we choose the image with the closest viewing direction for each subfacet.

The authors are with the Computer Vision and Geometry Lab, ETH Zurich, 8092 Zurich, Switzerland.
{hengli,glee}@student.ethz.ch
{friedrich.fraundorfer, marc.pollefeys}@inf.ethz.ch

Our approach is novel in two ways: we propose a 1-point RANSAC method for visual odometry using IMU information, and we incrementally build in real-time a global map of the environment with photo-realistic textures.

A. Related Work

Debevec et al. [3] first showed view-dependent projective texture mapping with static polygon models. [4] uses a DSLR camera and a high-end 3D laser rangefinder to generate 6D point clouds which are used to build a 3D occupancy grid; a coarsely textured map is built offline by colorizing each voxel considered as occupied using the average color of the points falling in that voxel. The map is used to localize a robot indoors. [5] uses the same sensor setup to perform automated 3D reconstruction using a wheeled robot which conducts frontier-based exploration; data collected at the end is used to build a high-quality 3D textured model offline. Real-time mapping of an unmanned ground vehicle's surroundings is achieved in [6] with a GPS/INS system, nodding laser rangefinder and a camera. They use the most recent image to texture new voxels in the 3D model; the texture of existing voxels cannot be changed regardless of whether there are better images with closer view directions. [7] collects pose estimates from a high-end GPS/INS system and image data from multiple cameras on a car moving in urban environments, computes the camera pose and a stereo depthmap for each video frame via a multi-view plane-sweep algorithm, and subsequently, generates a 3D textured n-layer heightmap model. However, the visual details of the model are not sharp due to local color averaging over multiple images. The approaches discussed so far build an accurate geometric model of the environment with stable ground-based platforms, high-quality pose estimates, and in most cases, extremely accurate ranging sensors. In contrast, mapping is a far more daunting task for MAVs; computational resources are severely limited, their MEMS IMUs although light-weight provide lower-quality estimates which are further degraded by mechanical vibration during flights, and the choice of 3D depth sensors is restricted to stereo cameras and structured light scanners; these sensors have significantly more measurement noise.

The MAVs of [8] and [9] apply a variant of the 3-point RANSAC algorithm to stereo data for visual odometry, and use a SLAM algorithm and a laser rangefinder to build a 2D map. Similarly, [10] constructs a multi-layer 2D map using data from a laser rangefinder. [11] extends the mapping capabilities of MAVs to 3D using a time-of-flight 3D camera. However, these maps lack texture. In [12], a hand-held RGBD camera is moved in an indoor environment, and a 3D surfel-based model is built offline with a SLAM algorithm estimating the camera poses. This RGBD camera cannot work in outdoor settings, and for real-time outdoor mapping, stereo cameras are the only 3D sensing modality for MAVs. Known work on real-time pose estimation methods for MAVs do not provide estimates reasonably good for alignment of projected images.

[13] exploits the nonholonomic constraints of wheeled vehicles to achieve a 1-point RANSAC instead of doing the standard 5-point RANSAC for relative motion estimation of a monocular camera. The accuracy of this algorithm is however limited to the fidelity of the nonholonomic motion model. [14] alleviates this limitation by using the prior information from Extended Kalman Filter (EKF) estimation as the motion estimation model in the RANSAC algorithm. However, none of these methods makes use of the IMU readings to achieve the 1-point RANSAC. Other existing works such as [15] improve the visual odometry results by fusing them together with IMU readings using the EKF estimator.

II. RGBD IMAGES

We rectify all camera images such that it is easy to infer 3D points from 2D image points and vice-versa via perspective projection. This conversion is heavily used throughout this paper, and the assumption that each image is rectified enables an efficient processing pipeline.

Each pixel in a RGBD image has both color and depth information. In the next subsections, we describe how we get depth information from a stereo camera and Kinect device. We compute the 3D coordinates of each pixel relative to the camera coordinate system:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} = zM \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \text{ where } M = \frac{1}{f} \begin{bmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & f \end{bmatrix} \quad (1)$$

where z is the depth associated with the pixel, (c_x, c_y) are the coordinates of the principal point of the camera, f is the focal length, and (i, j) are the image coordinates of the pixel. The values of c_x, c_y, and f, together with the stereo baseline b are obtained from an one-time calibration.

We then find the world coordinates of each point:

$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \end{bmatrix} = {}^{imu}_{world}H \, {}^{cam}_{imu}H \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} \quad (2)$$

where ${}^{imu}_{world}H$ is the homogeneous transform from the world frame to the IMU frame and is estimated by the visual odometry, and ${}^{cam}_{imu}H$ is the homogeneous transform from the IMU frame to the camera frame and is estimated using the InerVis calibration toolbox [16].

A. Depth from Stereo Camera

With each stereo image pair, we use the OpenCV implementation of the stereo correspondence algorithm to build a dense 640 x 480 disparity map. Subsequently, we compute the depth to the points in the scene relative to the camera coordinate system:

$$z_{cam} = \frac{bf}{d} \quad (3)$$

where d is the disparity. Differentiation of Equation 3 with respect to d yields:

$$\Delta z_{cam} = \frac{bf}{d^2} \Delta d \quad (4)$$

Δz_{cam} denotes the resolution of the range measurement corresponding to d. To avoid spurious range measurements due to small disparities from affecting the map quality, we set the minimum disparity:

$$d_{min} = \left\lceil \sqrt{\frac{bf\Delta d}{r}} \right\rceil \tag{5}$$

where r is the grid resolution. In our case, we choose a conservative value of $\Delta d = 0.5$.

B. Depth from Kinect

We use the OpenNI framework to obtain depth information which is hardware-registered to the RGB image. In other words, we do not have to find the extrinsic parameters that transform the depth points from the infrared camera's reference frame to the RGB camera's reference frame, as OpenNI automatically does this transformation.

III. 1-Point RANSAC and Visual Odometry

The standard approach to find the relative pose between two consecutive frames is to use the 3-point RANSAC algorithm [2]. In each frame, we find keypoints using the FAST detector [17], and extract Calonder feature descriptors [18]. First, 2D-2D correspondences between two consecutive frames are established. The 3D-3D correspondences are then obtained from the depth map. Next, random sets of 3-point correspondences are generated to compute the hypotheses of the relative camera motion R and t in the RANSAC loop using the absolute orientation [19] formulation:

$$\{{}^a_bR^*, {}^a_bt^*\} = \underset{\{{}^a_bR, {}^a_bt\}}{\operatorname{argmin}} \sum_i^n \|X_i^a - ({}^a_bRX_i^b + {}^a_bt)\|^2 \tag{6}$$

where X_i^a and X_i^b are the 3D point sets from the previous and current frames respectively. A closed-form solution to the absolute orientation problem is:

$$\begin{aligned} H &= \frac{1}{N}\sum_i^N (X_i^b - \bar{X}^b)(X_i^a - \bar{X}^a)^T && (7) \\ {}^a_bR^* &= VU^T && (8) \\ {}^a_bt^* &= \bar{X}^a - {}^a_bR^*\bar{X}^b && (9) \end{aligned}$$

where U and V are the left and right singular vectors from the singular value decomposition (SVD) of H. $\bar{X}^a$ and $\bar{X}^b$ are the centroids of the 3D point sets from X_i^a and X_i^b respectively. Finally, all the inliers from the hypothesis with the highest RANSAC score are used to compute a least squares estimate of the relative camera pose given by Equation 6.

3-point correspondences are needed to compute R and t from Equation 6. However, the IMU readings conveniently provide us with the relative rotation R between two consecutive frames. As such, we propose to use the relative rotation R provided by the IMU to align the orientation of the consecutive 3D-3D point correspondences and this reduces the unknown parameters in Equation 6 to the relative translation t:

$${}^a_bt^* = \underset{{}^a_bt}{\operatorname{argmin}} \sum_i^n \|X_i^a - (X_i^{b'} + {}^a_bt)\|^2 \tag{10}$$

where $X_i^{b'}$ is X_i^b rotated with R from the IMU reading. Our approach of using IMU readings for 1-point RANSAC differs significantly from other 1-point RANSAC approaches mentioned earlier in Section IA. Consequently, only 1-point correspondences are needed to get all the inliers from the RANSAC process. To improve the relative pose estimates, we further propose to compute the final R and t from all the inliers found from the 1-point RANSAC with Equation 6.

The number of hypotheses needed in the RANSAC process is given by:

$$N = \frac{\log(1-p)}{\log(1-\alpha^s)} \tag{11}$$

where s is the number of features needed, p is the probability that all selected features are inliers (p is usually assigned as 0.99), and a is the probability that any selected is an inlier. We can deduce from Equation 11 that the number of hypotheses needed decreases tremendously for the 3-point case ($s = 3$) and our 1-point case ($s = 1$). For example, the number of hypotheses needed drops from 35 to 7 assuming that $p = 0.99$ and $a = 0.5$. As a result, the computation time for the relative pose estimation process is reduced significantly. Another advantage of the 1-point RANSAC is that it is computationally feasible to carry out an exhaustive search on all point correspondences for inliers as compared to the 3-point RANSAC.

Lastly, we obtain the camera pose by concatenating the relative poses estimated from our 1-point RANSAC. To ensure the accuracy of the visual odometry over an extended distance, we further refine the estimated poses and 3D structure points with local bundle adjustment [20].

We use a modified version of the VSLAM package [21] available in ROS.

IV. Occupancy Grid Mapping

To downsample the range data in each RGBD frame and remove outliers, we place the range points in a spherical grid centered at the camera's position. For each cell in the spherical grid, we find the median of the range points in that cell, and refer to the line from the grid origin to that median as a virtual ray. We define a virtual scan as a set of virtual rays [1]; we traverse the rays in each scan, and at the same time, update the occupancy grid cells intersected by the rays. Another advantage of computing virtual scans is that updating of the occupancy grid is more efficient as we rasterize the rays in the virtual scan instead of those from the camera to each point in the entire point cloud which often constitutes many thousands of points.

We use the multi-volume occupancy grid of [22] which explicitly models both occupied and free space. As we traverse in order the cells intersected by each ray in the virtual scan, we insert negative volumes in these cells until we reach the cell where the endpoint of the ray is located in; we insert a positive volume in that cell. Positive volumes indicate obstacles while negative volumes indicate free space. Modeling of both free and occupied space is important as erroneous range readings, especially from stereo, can be later corrected. The occupancy probability of a point is determined by the occupancy density of the positive volume it is in, divided by the sum of the occupancy densities of both the positive and negative volumes. We use a low occupancy threshold, as having parts of the foreground mapped on ground voxels is undesirable and is seen as degrading the photorealism of the textured map.

We merge a local grid transmitted by the MAV with the global grid stored in the ground station by simply merging both the positive and negative volumes with those of the corresponding cells in the local grid.

V. PROJECTIVE TEXTURE MAPPING

We project the images onto the occupancy grid as if by a slide projector. However, projective texture mapping requires us to determine which parts of which facets of each voxel are visible in each image. For clarity, we use the term subfacet to refer to a part of a facet, and that each subfacet is defined to be either wholly visible or wholly invisible in any image.

The outline of the algorithm is:

1) For each image,
    a) Construct the view frustum.
    b) For each cell in the occupancy grid,
        i) Determine the section of the cell that is within that view frustum. This cell section is defined using two values: the height of its bottom face and the height of its top face, and we refer to these two values as a z-interval.
        ii) For each voxel in the cell, we project the facets of each voxel located within the z-interval into image space, and back to object space. The resulting 3D points identify the visible parts of the facets.
        iii) Using the visibility information, we split the facets into two distinct sets of visible and non-visible subfacets.
2) For each subfacet in the visible set, choose the image with the closest viewing angle for that subfacet.

Figure 2 shows an example of how one facet of a voxel is textured given two images in which that facet is visible.

For real-time performance, a hybrid CPU-GPU pipeline is used to perform texture mapping; step 1b in the above-mentioned algorithm is implemented in the GPU.

To avoid projecting black border pixels from rectified images which could otherwise create black spots on the textured map, we determine the region of interest (ROI) in the images in which no black border pixel is present. The ROI is represented as a rectangle where (x, y) and (x', y') represent the two opposite corners and $x' = x+w$, $y' = y+h$. We project this ROI instead of the rectified image onto subfacets of voxels in the occupancy grid that are visible from the viewpoint.

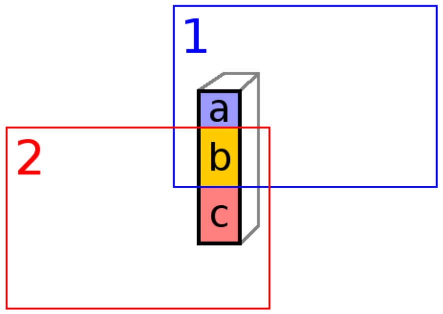


Fig. 2. The facet of a voxel outlined in black is non-occluded and is partially visible in both images 1 and 2. Hence, the facet is partitioned into three subfacets: a, b, and c, such that each subfacet is either wholly or not visible in each image. Subfacet a is assigned a texture from image 1, while subfacet c is assigned a texture from image 2. Since subfacet b is visible in both images, subfacet b is assigned a texture from the image with the closer viewing angle.

A. View Frustum

The method of checking whether each voxel in the occupancy grid lies in the view frustum for each camera viewpoint takes time linear in the number of viewpoints, and thus, is not scalable to large environments. We propose an efficient method that is guaranteed to run in constant time.

When a new camera viewpoint is added, an one-time update is applied to each cell in the grid: the cell stores for that viewpoint the range of z-values that lie within the view frustum associated with the viewpoint.

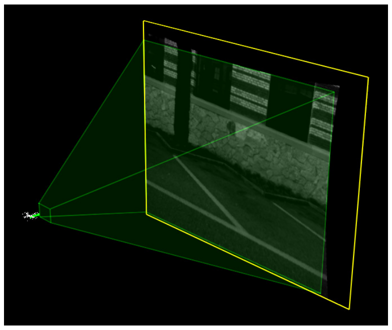


Fig. 3. The view frustum is marked by green lines with the border of the superimposed rectified image marked in yellow.

The view frustum is modeled as a rectangular pyramid with 8 corners as shown in Figure 3. Using the matrix M in Equation 1 and the ROI coordinates, the set of corner coordinates is defined:

$$C = \left\{ zM \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, zM \begin{bmatrix} x' \\ y \\ 1 \end{bmatrix}, zM \begin{bmatrix} x \\ y' \\ 1 \end{bmatrix}, zM \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \right\} \quad \text{for } z \in \{z_{near}, z_{far}\} \qquad (12)$$

where $z_{near} = \frac{bf}{d_{max}}$ and $z_{far} = \frac{bf}{d_{min}}$ are the depths of the near and far planes respectively. The six planes that make up the frustum can be computed from these frustum corners.

Algorithm 1 Check if z-interval $[z_{min}, z_{max}]$ in cell with coordinates (x, y) and width w exists within the view frustum, and if so, compute the z-interval.

```
1:  $z_{min} = \infty$
2:  $z_{max} = -\infty$
3:  for $i = 0$ to 6 do
4:      $[\hat{n}_x\ \hat{n}_y\ \hat{n}_z\ d] = \pi_i$
5:      $v_n = [x - \frac{w}{2} sign(\hat{n}_x)\ \ y - \frac{w}{2} sign(\hat{n}_y)]$
6:      $z = [-\hat{n}_x\ \ -\hat{n}_y\ \ -d\ \ 0]^T \cdot [v_n\ 1\ 0]^T$
7:      if $\hat{n}_z > 0$ then
8:          if $z_{max} < z$ then
9:              $z_{max} = z$
10:         end if
11:     else
12:         if $z_{min} > z$ then
13:             $z_{min} = z$
14:         end if
15:     end if
16: end for
17: if $\forall(\hat{n}, d) \in \Pi$ such that $\hat{n} \cdot [x\ y\ z_{min}]^T \geq -d$ and
    $\hat{n} \cdot [x\ y\ z_{max}]^T \geq -d$ then
18:     $[z_{min}, z_{max}]$ lies within view frustum
19: else
20:     $[z_{min}, z_{max}]$ does not lie within view frustum
21: end if
```

We determine the z-interval of the cell that is within the view frustum by using algorithm 1. For simplicity, we assume an axis-aligned box spanning the cell cross-section and with infinite height. We find the z-intersection of the box with each frustum plane, and infer the z-interval from the 6 intersection points.

In line 4, we retrieve the parameters $\hat{n}$ and d of frustum plane i where $\hat{n}$ is the plane normal which points towards the center of the view frustum, d is the distance of the plane from the origin, and $\hat{n} \cdot [x\ y\ z]^T = -d$. Line 5 finds the (x, y) coordinates of the negative vertex of the box where the negative vertex is defined to be the vertex of the box nearest to the plane along the normal's direction. The z-coordinate of the negative vertex is found in line 6. Lines 7-15 iteratively narrows the z-interval until it spans the view frustum. Line 16 ensures that the z-interval lies in the view frustum by checking that the signed distance of each of the two interval end-points to every frustum plane is positive.

B. Depth Mapping

We determine which parts of the facets are visible in each image; that image is considered as a candidate for projection onto these visible subfacets. We use the OpenGL depth buffer feature to project all front-facing facets of the voxels within the view frustum to the depth buffer in image space. Each pixel in the depth buffer stores the depth to the nearest voxel with an example shown in Figure 4. In projecting the facets, we use the intrinsic parameters of the camera that was used to capture the image. At the same time, we assign an unique ID to each facet. In this way, occluded parts of facets do not show up in the image. We project the facet points from image space to object space, and for each facet, we accumulate all points associated with that facet. We compute a set of z-intervals from the z-coordinates of the points with the constraint that any two points further than a minimum distance belong to different intervals. These z-intervals mark the visible parts of the facet.

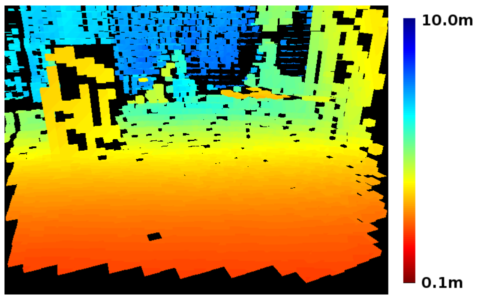


Fig. 4. Visualization of a depth buffer given a camera viewpoint. For each pixel, the more cool (blue) the color, the further the distance between the camera viewpoint and the object occupying that pixel.

C. Facet Splitting & View Selection

Using the visibility information from depth mapping, we horizontally split each facet into subfacets such that the resulting subfacets either lie within or outside the z-intervals. We determine the set of images that each subfacet is visible in, and we project the image with the closest viewing direction onto that subfacet. We calculate the viewing direction using:

$$\theta = \arccos\left(\frac{p_{subfacet} - p_{camera}}{\|p_{subfacet} - p_{camera}\|} \cdot \hat{n}\right) \tag{13}$$

where $p_{subfacet}$ is the position of the mid-point of the subfacet, p_{camera} is the camera position, and $\hat{n}$ is the unit normal of the subfacet.

VI. IMPLEMENTATION

Our platform is a PIXHAWK quadrotor equipped with infrastructure necessary for real-time computer vision algorithms. A forward-looking stereo rig with a 25 cm baseline serves as the main exteroceptive sensor. The cameras are synchronized to the IMU, allowing precise IMU metadata to be attached to each stereo image pair. To demonstrate the extensibility of our approach with multiple sensor types, we also use a Kinect device; we maintain a queue of IMU measurements, and when a frame arrives, it is timestamped, and the timestamp is used to compute the IMU data for that frame by interpolating between IMU measurements with close timestamps. Data communications within the MAV and with the ground station is maintained over a Wi-Fi network using MIT's LCM middleware. In particular, we use Intel SIMD extensions to accelerate the projection of points between image space and object space.

When the camera has moved either a minimum Euclidean distance or angular distance, the current RGBD image is marked as a keyframe. In our case, the minimum Euclidean distance and angular distance are 0.2 m and 0.1 radians respectively. At this point, the visual odometry module publishes the camera pose associated with the keyframe. Upon receiving the camera pose, the 3D mapping module computes the local occupancy grid and rectified reference image based on sensor data associated with the camera pose. This 3-tuple dataset is then transmitted to the ground station which merges the local occupancy grid with its locally-stored rolling-window based global occupancy grid. The size of the global grid is limited to 60 m x 60 m; any cell falling outside the grid is simply removed. The ground station subsequently constructs and shows the 3D textured map in a Qt-based visualization window.

VII. EXPERIMENTS AND RESULTS

We build several 3D textured maps in both outdoor and indoor settings. For the outdoor setting, we mount a stereo camera on the quadrotor, and for the indoor setting, we mount a Kinect device. We estimate the camera pose via our visual odometry with 1-point RANSAC. Figure 5 compares the ground truth from the Vicon motion capture system with the visual odometry estimates from the quadrotor with a Kinect device.

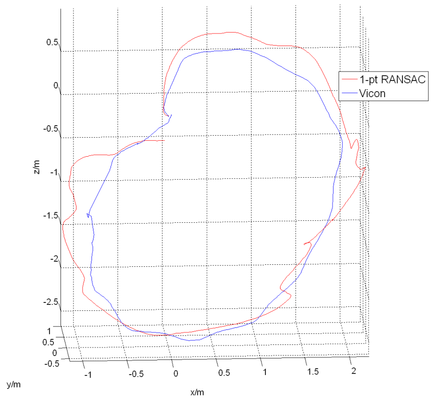


Fig. 5. Comparison of visual odometry output with Vicon ground truth.

In Figures 6 and 7 which correspond to the outdoor and indoor datasets respectively, we show the 3D texture maps together with the underlying occupancy grids. To demonstrate the photo-realistic aspect of the 3D texture maps, we compare these maps against the point cloud representations which are shown to be not as clearly defined as the 3D texture maps. For the outdoor and indoor data sets, the camera trajectories estimated by the visual odometry module are marked in red.

Our 3D textured map is far more compact than a point cloud representation, and still, shows a slightly higher quality of image detail compared to the point cloud representation. In addition, the path planner successfully uses these maps to plan an obstacle-free flight path to user-defined goal points. The visual odometry runs at 10 Hz, while the texture mapping module runs at 6 Hz. Table I shows the breakdown of computational time for the 3D reconstruction on the ground station.

TABLE I

BREAKDOWN OF COMPUTATIONAL TIME FOR 3D RECONSTRUCTION ON THE GROUND STATION. (2.80 GHZ INTEL®CORE^{TM}I7, NVIDIA®GEFORCE®GTX 285)

Process	Average Computational Time
Merging local map with global map	37 ms
View frustum computation	9 ms
Depth mapping	95 ms
Facet splitting and view selection	19 ms
Total	160 ms

The accompanying video shows the live 3D reconstruction of a MAV's environment using both a stereo camera and Kinect device.

VIII. CONCLUSIONS

We have shown our real-time mapping framework to produce geometrically accurate and photo-realistic 3D maps which can be used for both visualization and path planning. Our visual odometry based on 1-point RANSAC estimates camera poses sufficiently accurate for creating photo-realistic maps using RGBD images from many viewpoints. We acknowledge that when mapping an environment using visual odometry, the camera pose drifts over time, and an area visited twice may show up in two different locations on the map. Hence, we are working towards incorporating loop closure in our visual odometry algorithm in order to obtain globally-consistent maps.

IX. ACKNOWLEDGMENTS

The first author was funded by the DSO National Laboratories Postgraduate Scholarship. In addition, this work was supported in parts by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #231855 (sFly) and by the Swiss National Science Foundation (SNF) under grant #200021-125017.

REFERENCES

[1] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, Autonomous Obstacle Avoidance and Maneuvering on a Vision-Guided MAV Using On-Board Processing, In *Proc. International Conference on Robotics and Automation*, 2011.

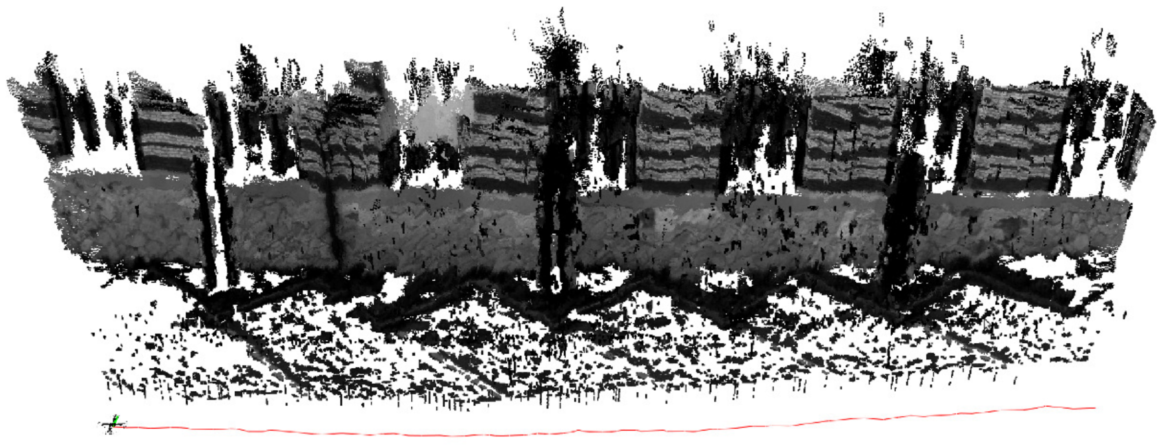
[2] M. Fischler and R. Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography, *Comm. of the ACM, 24:381-395*, 1987.

[3] P. Debevec, Y. Yu, and G. Borshukov, Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping, In *Proc. Eurographics Rendering Workshop*, 1998.

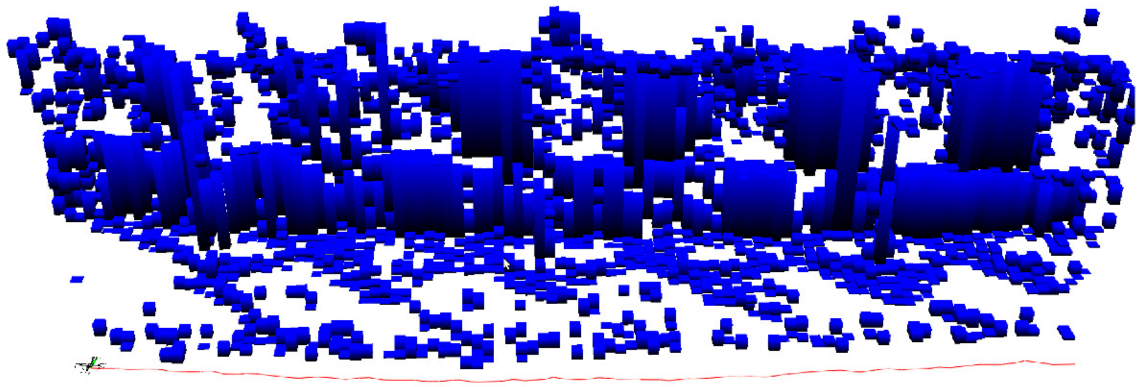
[4] J. Mason, S. Ricco, and R. Parr, Textured Occupancy Grids for Monocular Localization Without Features, In *Proc. International Conference on Robotics and Automation*, 2011.

[5] B. Pitzer, S. Kammel, C. DuHadway, and J, Becker, Automatic reconstruction of textured 3D models, In *Proc. International Conference on Robotics and Automation*, 2010.

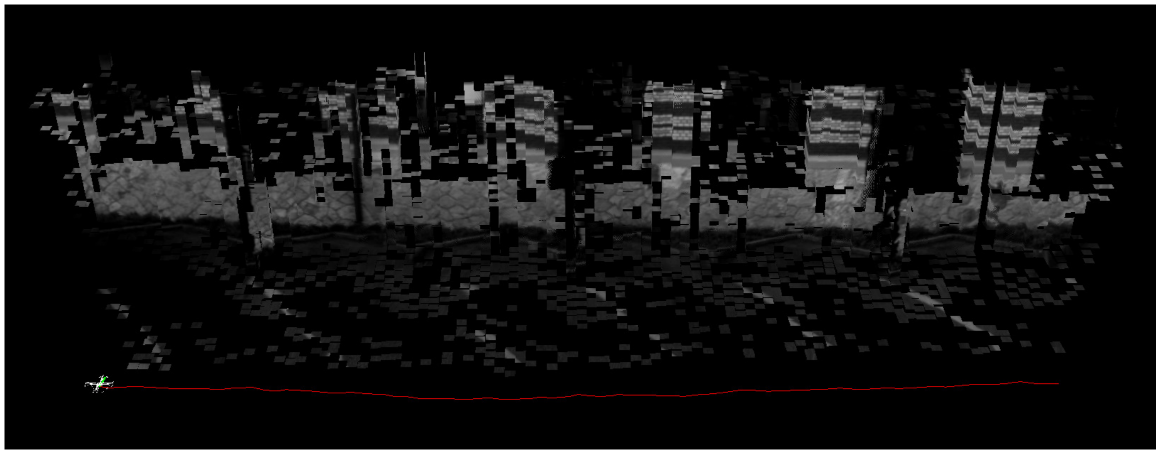
[6] D. Huber, H. Herman, A. Kelly, P. Rander, and J. Ziglar, Real-time Photo-realistic Visualization of 3D Environments for Enhanced Tele-operation of Vehicles, In *Proc. International Conference on 3-D Digital Imaging and Modeling*, 2010.



(a) Raw point cloud data

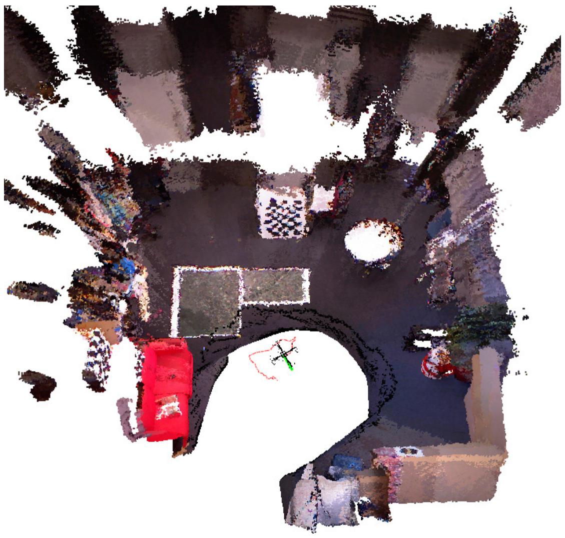


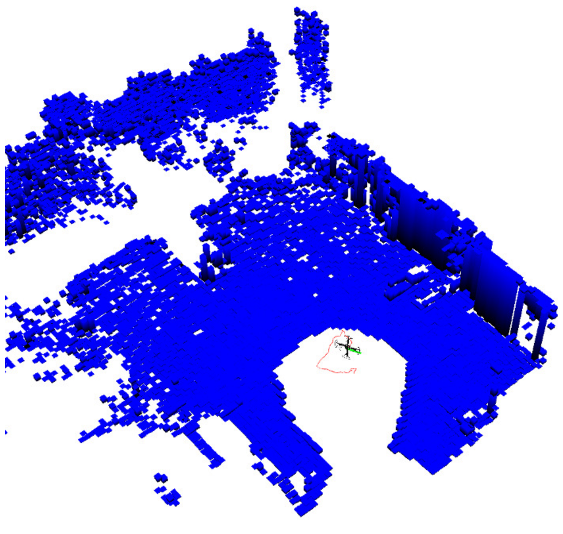
(b) 3D occupancy map

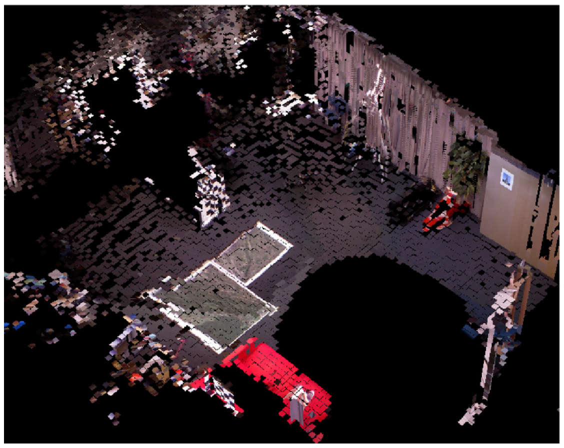


(c) 3D textured map - Front view

Fig. 6. A comparison of the raw point cloud representation with the occupancy grid representation and 3D textured map representation for the stereo camera with poses estimated by our visual odometry with 1-point RANSAC. The grid resolution is 0.2m.


(a) Raw point cloud data


(b) 3D occupancy map


(c) 3D textured map - Front view

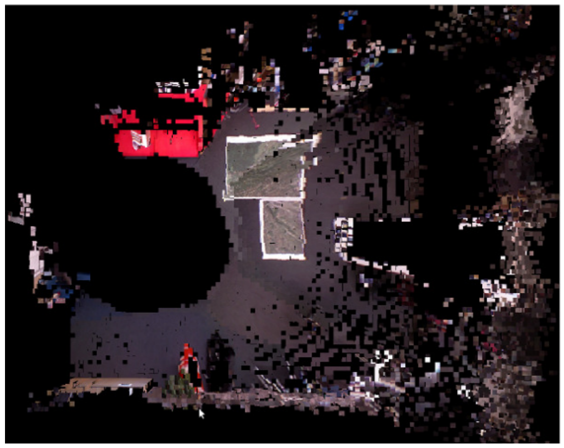

(d) 3D textured map - Top view

Fig. 7. A comparison of the raw point cloud representation with the occupancy grid representation and 3D textured map representation for the Kinect device with poses estimated by our visual odometry with 1-point RANSAC. The grid resolution is 0.1m.

[7] D. Gallup, M. Pollefeys, and J. Frahm, 3D Reconstruction Using an n-Layer Heightmap, In *Proc. Annual Symposium of the German Association for Pattern Recognition*, 2010.

[8] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-denied Indoor Environments, In *Proc. SPIE Conference on Unmanned Systems Technology XI*, 2009.

[9] A. Bachrach, R. He, and N. Roy, Autonomous Flight in Unknown Indoor Environments, *International Journal of Micro Air Vehicles, 1(4): 217-228*, 2009.

[10] S. Shen, N. Michael, and V. Kumar, Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV, In *Proc. International Conference on Robotics and Automation*, 2011.

[11] W. Morris, I. Dryanovski, and J. Xiao, 3D Indoor Mapping for Micro-UAVs Using Hybrid Range Finders and Multi-Volume Occupancy Grids, In *Proc. Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Robotics: Science and Systems Conference*, 2010.

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments, In *Proc. International Symposium on Experimental Robotics*, 2010.

[13] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, Real-Time Monocular Visual Odometry for On-Road Vehicles with 1-Point RANSAC, In *Proc. International Conference on Robotics and Automation*, 2009.

[14] J. Civera, O. Grasa, A. Davidson, and J. Montiel, 1-Point RANSAC for EKF-Based Structure from Motion, In *Proc. International Conference on Intelligent Robots and Systems*, 2009.

[15] K. Konolige, M. Agrawal, and J. Solà, Large Scale Visual Odometry for Rough Terrain, In *Proc. International Symposium on Research in Robotics*, 2007.

[16] J. Lobo, and J. Dias, Relative Pose Calibration Between Visual and Inertial Sensors, In *International Journal of Robotics Research, 26(6):561-575*, 2007.

[17] E. Rosten, and T. Drummond, Machine learning for high-speed corner detection, In *In Proc. European Conference on Computer Vision*, 2006.

[18] M. Calonder, V. Lepetit, and P. Fua, Keypoint Signatures for Fast Learning and Recognition, In *In Proc. European Conference on Computer Vision*, 2008.

[19] B. Horn, Closed-form solution of absolute orientation using unit quaternions, In *Journal of the Optical Society of America, 4(4):629-642*, 1987.

[20] Z. Zhang and Y. Shan, Incremental Motion Estimation Through Local Bundle Adjustment, *Microsoft Research, Tech. Rep. MSR-TR-01-54*, 2001.

[21] K. Konolige, VSLAM package, Robot Operating System, available at `http://www.ros.org/wiki/vslam`

[22] I.Dryanovski, W. Morris, and J. Xiao, Multi-Volume Occupancy Grids: an Efficient Probabilistic 3D Mapping Model for Micro Aerial Vehicles, In *Proc. International Conference on Intelligent Robots and Systems*, 2010.