MAV Visual SLAM with Plane Constraint

Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys
Computer Vision and Geometry Laboratory, Department of Computer Science, ETH Zürich, Switzerland
glee@student.ethz.ch, fraundorfer@inf.ethz.ch, marc.pollefeys@inf.ethz.ch

Abstract— **Bundle adjustment (BA) which produces highly accurate results for visual Simultaneous Localization and Mapping (SLAM) could not be used for Micro-Aerial Vehicles (MAVs) with limited processing power because of its $O(N^3)$ complexity. We observed that a consistent ground plane often exists for MAVs flying in both the indoor and outdoor urban environments. Therefore, in this paper, we propose a visual SLAM algorithm that make use of the plane constraint to reduce the complexity of BA. The reduction of complexity is achieved by refining only the current camera pose and most recent map points with BA that minimizes the reprojection errors and perpendicular distances between the most recent map points and the best fit plane with all the pre-existing map points. As a result, our algorithm is approximately constant time since the number of current camera pose and most recent map points remain approximately constant. In addition, the minimization of the perpendicular distances between the plane and map points would enforce consistency between the reconstructed map points and the actual ground plane.**

I. INTRODUCTION

In the recent years, there is a growing interest among various research groups [1]–[3] in the development of autonomous MAVs for urban environments. This increase of interests could largely be attributed to the high maneuverability of MAVs which promises the possibilities of achieving more sophisticated robotics tasks in urban environments. For the MAVs to interact autonomously with the environment, it must possess the ability to learn a map of its environment and to localize itself with respect to this map. This is referred to as the SLAM problem. A camera is an excellent sensor for MAVs to do SLAM. Besides being a light-weight device which is suitable for MAVs with limited payload, images from a camera also provide rich information of the environment.

Two main groups of algorithms exist for visual SLAM - BA from structure from motion [6]–[8] and the Bayesian filters such as the Extended Kalman Filter (EKF) [9] and Particle Filter [10]. BA works by minimizing the reprojection errors from an initial estimate of the full camera trajectory and map points using non-linear least square minimization methods such as the Levenberg-Marquardt algorithm. Results of high accuracy are often obtained from BA. However, because BA requires an initial estimate of the full camera trajectory and map points, it is therefore an offline algorithm for visual SLAM. In addition, non-linear least square minimization of reprojection errors involves inversion of Jacobian matrices which has a complexity of $O(N^3)$, making it difficult to work for long trajectories and MAVs with limited on-board processing power.

The Adaptive Relative Bundle Adjustment [4] reduces the complexity of BA by re-parameterizing the absolute coordinates of the camera trajectory and map points into relative coordinates. This allows optimization of only the parameters that might change when there is new information thus greatly reduces computation cost. However, the recovery of absolute coordinates from relative coordinates becomes extremely slow as the number of camera poses and map points grow. In the Local Bundle Adjustment [5], computation complexity is reduced by only considering a number of most recent camera frames and the map points seen by these camera frames in the BA process. This keeps the number of parameters in BA approximately constant and hence its complexity is approximately constant time. However, significant drifts could be observed in the estimates from Local BA because it completely ignores all the previous camera poses and map points.

Klein demonstrated monocular SLAM in real-time with his PTAM framework [22]. The PTAM framework is divided into two threads. One thread constantly does local localization which estimates the camera pose with respect to the existing map. The other thread does a BA on the whole map and all the camera poses when resources become available. The PTAM framework works very fast with the local localization thread. However, it slows down significantly when the BA thread is activated thus causing it to work only in small environments.

Bayesian Filters on the other hand provide the solution to online Visual SLAM. Davison's MonoSLAM [9] made used of the EKF which is free from Jacobian inversion thus making it an online SLAM. However, its limitation as an online SLAM algorithm sets in as the number of features on the map grows. This is because EKF maintains a covariance matrix of all features and current camera pose that requires $O(N^2)$ complexity to update. Furthermore, the linearization of the non-linear camera projection model causes important information to be lost and eventually leading to inconsistencies in the estimation of the camera poses and map [11]. Ethan tried to overcome these problems in his Scalable Monocular SLAM [10] by making use of the particle filter. However, the particle filter also suffers the same fate of inconsistencies because it does not store the covariance between map features and camera poses [12]. In addition, particle filter suffers from the curse of dimensionality. Infinite number of particles is needed for optimal estimates. However, this also requires infinite processing time.

This paper focuses on the development of a visual SLAM algorithm suitable for MAVs navigating in urban environments. We observed that a consistent ground plane often exists in most indoor and outdoor urban environments, for example, the ground plane along a corridor or a street. By mounting a downward looking camera on the MAV, features could be extracted from the ground plane. With the prior knowledge of the plane constraint, our visual SLAM algorithm estimates the plane parameters from RANSAC [14] plane fitting with all the pre-existing map points, followed by a refinement of the current camera pose and the most recent map points with a BA process that minimizes the reprojection errors and the perpendicular distances between the plane and 3D map points. This process is repeated with the incoming of every set of new camera pose and map points. Our method is most similar to Local BA, instead of optimizing the full $(6n + 3m)$ camera poses and map features with BA, we reduces complexity significantly by estimating only 4 plane parameters with RANSAC, and $(6 + 3m')$ current camera pose and most recent map points with BA where $m' << m$. Since the total number of current camera pose and m' map points stay approximately constant, our visual SLAM with plane constraint is approximately constant time. In contrast with Local BA, our method has a significant advantage of information from previous estimates not being lost because we use the pre-existing map points to estimate the plane for enforcing planar constraint on incoming map points.

This paper is structured as follows. In Section II, we briefly discuss BA which is a prerequisite to understand our visual SLAM algorithm. Section III describes our visual SLAM algorithm in detail. In Section IV, we show the simulation and implementation results of our visual SLAM algorithm. Lastly, we show in Section V that it is also possible to extend our algorithm to handle environments with multiple planes.

II. Bundle Adjustment

In this section, we briefly describe the BA process which is needed to derive our visual SLAM algorithm in the next section. A more detailed explanation is given in [8]. Given the initial estimates of the m 3D map points $[X_1^T, X_2^T, ..., X_m^T]^T$, their corresponding 2D image features over n views $[X_{11}^T, X_{12}^T, ..., X_{1m}^T, X_{21}^T, X_{22}^T, ..., X_{2m}^T, ..., X_{nm}^T]^T$, and the initial estimates of the camera poses $[C_1, C_2, ..., C_n]^T$. The objective of BA is to find the optimal values for the 3D map points and camera poses that minimize a cost function that is made up by the total reprojection errors of the map features onto the respective images as shown in Equation 1. $d(.)$ denotes the Euclidean distances between the image points and its reprojection, and $Q(.)$ is the camera projection function. Note that $d(.) = 0$ when X_j is not in the view of camera C_i.

$$\underset{C,X}{\operatorname{argmin}} \sum_{i}^{n} \sum_{j}^{m} d\left(Q\left(C_i, X_j\right) - x_{ij}\right) \tag{1}$$

The Levenberg-Marquardt algorithm is the most commonly used method for solving the BA problem. Let $\beta = [C_1, C_2, ..., C_n, X_1^T, X_2^T, ..., X_m^T]^T$. Levenberg-Marquardt iteratively moves β towards the optimal values for minimizing Equation 1 by solving for $\delta\beta$ in the augmented normal equation shown in Equation 2. A better β value for minimizing the reprojection errors is obtained from $\beta_i = \beta_{i-1} + \delta\beta$ after every iteration. The Jacobian $J = [J_C, J_X]$ where J_C and J_X denote the camera projection function $Q(.)$ with respect to the camera poses and 3D map points, ϵ is the reprojection errors and λ is the non-negative damping parameter for controlling the rate of convergence. The Levenberg-Marquardt iteration is terminated when ϵ falls below a threshold value.

$$\left(J^T J + \lambda I\right) \delta = -J^T \epsilon \tag{2}$$

The Levenberg-Marquardt algorithm is very effective in minimizing the total reprojection errors. However, the algorithm requires an inversion of $\left(J^T J + \lambda I\right)$ at every iteration which has a complexity of $O(N^3)$. $\left(J^T J + \lambda I\right)$ grows linearly with the number of camera poses and map features. This means that it is difficult to do BA on MAVs with limited processing power over a long trajectory.

III. Our Visual SLAM Algorithm

The 3D map points of our visual SLAM algorithm are initialized from the first two keyframes. With the prior knowledge of plane constraint, we optimize the plane parameters, first two camera poses and 3D map points with a BA process that minimizes the reprojection errors and perpendicular distances between the plane and map points. Subsequent camera poses and map points are estimated by a process that estimates the plane parameters from RANSAC plane fitting with all the pre-existing map points, followed by refinement of the current camera pose and map points with a BA process that minimizes the perpendicular distances between the plane and map points. This process is repeated with the incoming of every set of new camera pose and map points. A subtle difference between initialization and subsequent estimations is that the plane parameters are included in the BA for initialization. This is to ensure accurate estimation of the plane parameters in the absence of pre-existing map points. Our algorithm is approximately constant time since we are only estimating 4 plane parameters with RANSAC and $(6+3m')$ current camera pose and map points where the number of map points m' remains approximately constant.

A. Features Extraction and Correspondences

The primary step for all visual SLAM algorithms is the extraction of salient features from the image and get their 2D-2D correspondences across a sequence of images. In this paper, we choose to extract SIFT features from the images and their 2D-2D correspondences are found via the second nearest neighbor approach [15]. Other features such as SURF [16], Randomized Trees [17] and Ferns [18] could also be used.

Not all frames in the image sequence are used for feature correspondences because the 5-point and linear 4-point algorithms (see Sections III-B and III-C for more details) for estimating the camera poses work best when there is sufficient movements between the camera poses. As such, only keyframes with sufficient movements are selected for feature extraction and correspondences. An image frame is chosen as a keyframe if the average pixel movements of all the feature correspondences exceed a threshold value.

B. Initialization

Our SLAM algorithm is initialized with the first two keyframes from the image sequence. First, the Essential matrix that relates this pair of images is computed from the 5-point algorithm [13]. Next, the relative camera pose between the two keyframes is extracted from the Essential matrix. The RANSAC algorithm [14] is used for a more robust estimation of the Essential matrix. Lastly, with the knowledge of the relative pose between the first two keyframes, triangulation of the image features to get the 3D map points follows. These 3D map points generally do not lie on a plane due to the noise presented in the features extracted from the images. Figure 1 shows a possible distribution of the 3D map points around the ground plane π.

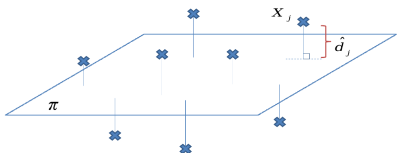

Fig. 1. A possible distribution of 3D map points obtained from triangulation.

We propose to improve the accuracy of the initial 3D map points and relative pose between the first pair of keyframes by minimizing the cost function given in Equation 3. This cost function is made up from the reprojection errors and the perpendicular distances $\widehat{d}$ between the 3D map points and the ground plane π. The ground plane π is parameterized by 4 parameters $[\pi_1, \pi_2, \pi_3, \pi_4]$ which describe the plane equation $\pi_1 x + \pi_2 y + \pi_3 z + \pi_4 = 0$. An initial estimate of these plane parameters is obtained from a RANSAC plane fitting process of the initial 3D map points. $d_\perp(.)$ is the function used to compute the perpendicular distance $\widehat{d}$ between a plane and a point, and it is given by Equation 4.

$$\underset{C_1, C_2, X, \pi}{\operatorname{argmin}} \sum_{j}^{m} \left\{ \sum_{i=1}^{2} d\left(Q\left(C_i, X_j\right) - x_{ij}\right) + d_\perp\left(\pi, X_j\right) \right\} \tag{3}$$

$$d_\perp\left(\pi, X_j\right) = \frac{|\pi_1 x_j + \pi_2 y_j + \pi_3 z_j + \pi_4|}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}} \tag{4}$$

The cost function from Equation 3 is minimized by iterating through the augmented normal equation shown in Equation 2 from the Levenberg-Marquardt algorithm mentioned in Section II. This can be achieved by augmenting $d_\perp(.)$ into the normal equation. As a result, the parameters which we try to estimate is given by $\beta = [C_1, C_2, X_1^T, X_2^T, ..., X_m^T, \pi_1, \pi_2, \pi_3, \pi_4]^T$, and the error $\epsilon = [\epsilon_d, \epsilon_{d_\perp}]^T$ where ϵ_d is the reprojection errors and $\epsilon_{d_\perp}$ is the perpendicular distances of each 3D map feature from the estimated plane. The Jacobian is given by

$$J = \begin{pmatrix} J_C & J_X & 0 \\ 0 & J_{X\pi} & J_\pi \end{pmatrix} \tag{5}$$

where $J_{X\pi}$ and J_π are the Jacobians of $d_\perp(.)$ with respect to the 3D map points and plane parameters.

C. Subsequent Pose Estimations

The 3D map points are used to estimate the subsequent camera poses after the two frames initialization. This involves finding the 2D-3D correspondence between the features in the current image frame and the 3D map points followed by solving the perspective problem [19] to obtain the current camera pose.

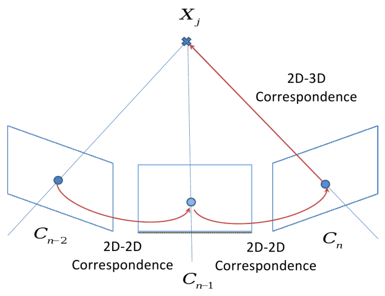

Fig. 2. The relation between 2D-2D and 2D-3D correspondences.

Figure 2 gives an illustration on how we can find out the 2D-3D correspondences between the current image features and the existing 3D map points. Supposed that X_j is a 3D map point triangulated from the 2D-2D correspondence between frames C_{n-2} and C_{n-1} , the 2D-3D correspondence between X_j and a image feature from the current frame C_n would be established if the image feature from the current frame is found to be a 2D-2D correspondence with the image feature that correspondence to X_j from the previous frame.

Once all the 2D-3D correspondences are found, the current camera pose is obtained from the linear 4-point algorithm [19]. The advantage of this method is that the relative scale of the current pose would also be found. We improve the robustness of the algorithm by coupling it with the RANSAC process. This is followed by addition of new points into the map. These new points are triangulated from new 2D-2D correspondences between the current and previous frames.

D. Bundle Adjustment with Plane Constraint

BA to the current camera pose and newly added 3D map points is done after solving the perspective problem for every new camera pose. Prior to BA, the plane parameters has to be estimated from RANSAC plane fitting of all the pre-existing map points. The estimated plane parameters are then used in BA as an additional constraint. In this way, we ensured that global information from all the pre-existing map points are not lost in the BA process. At the same time, we reduce the number of parameters to be estimated in the BA from all the camera poses and map points $(6n+3m)$ to the current camera pose and the newly added map points $(6+3m')$ where $m' << m$. As a result, our visual SLAM algorithm becomes approximately constant time since the number of current camera pose and newly added points stays approximately constant. To make our algorithm more robust, we remove any outliers from the map after RANSAC plane fitting.

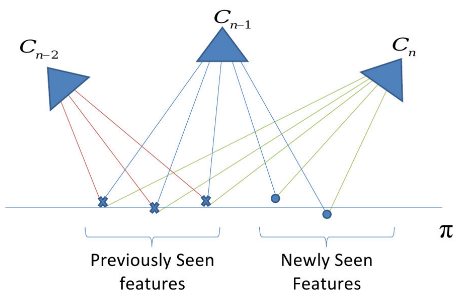

Fig. 3. Parameters used in our bundle adjustment.

Figure 3 shows an illustration of the parameters used in our BA. π is the plane parameters estimated from the RANSAC process mentioned earlier. The current camera pose C_n and m' newly seen 3D map points are the parameters to be estimated in our BA. Let us denote the m' newly seen features as X^{new}. The cost function for our BA is defined in Equation 6. Note that the previously seen points are also included into the cost function. Previously seen features are the 3D map points seen by the most recent 3 frames, C_n, C_{n-1}, C_{n-2} as illustrated in Figure 3. The reason for including the previously seen features is to provide additional constraints thus preventing wrong convergence of our BA. X' denotes all the previously and newly seen 3D map points and m'' denotes the total number of X'. It is important to note that the cost function in Equation 6 is minimized only with respect to C_n and X^{new}. The plane parameters π remain unchanged here. This is because the plane parameters contains information from the pre-existing map points and should only be changed after each BA process.

$$\underset{C_n, X^{new}}{\operatorname{argmin}} \sum_{j=1}^{m''} \left\{ \sum_{i=n-1}^{n} d\left(Q\left(C_i, X'_j\right) - x'_{ij}\right) + d_{\perp}\left(\pi, X'_j\right) \right\} \tag{6}$$

Similar to the 2 frames initialization, we minimize the cost function in Equation 6 with the Levenberg-Marquardt algorithm. The parameters that we estimate now is given by $\beta = [C_n, X_1^{newT}, X_2^{newT}, ..., X_{m'}^{newT}]^T$, and the error $\epsilon = [\epsilon'_d, \epsilon'_{d_\perp}]^T$ where ϵ'_d is the repojection errors for all X' in C_{n-1} and C_n, and $\epsilon'_{d_\perp}$ is the perpendicular distances of all X' with the estimated plane parameters prior to the BA. The Jacobian is given by

$$J = \begin{pmatrix} J_{C_n} & J_{X^{new}} \\ 0 & J_{X^{new}\pi} \end{pmatrix} \tag{7}$$

where J_{C_n} and $J_{X^{new}}$ are the Jacobians of the projection function $Q(.)$ with respect to the current camera pose C_n and the newly seen 3D map points X^{new}. $J_{X^{new}\pi}$ is the Jacobian of $d_\perp(.)$ with respect to the newly seen map points.

IV. Simulation and Implementation Results

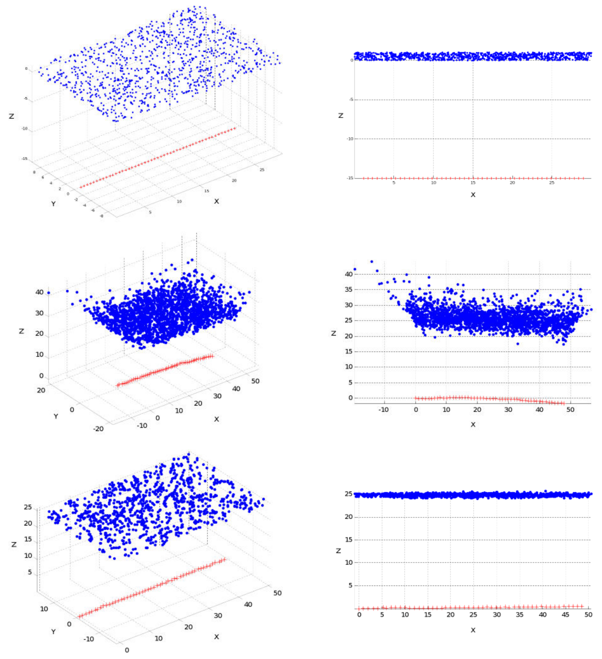

Fig. 4. (Top) Simulated planar environment. (Middle) Camera poses and 3D map points estimated from Local BA. (Bottom) Camera poses and 3D map points estimated from our visual SLAM algorithm.

Figure 4 (Top) shows a simulated environment with 1000 points lying on a plane with unit thickness (to show that the points need not fully lie flat on a plane for our algorithm to work) and 50 camera poses distributed evenly along the x-axis with no rotation. Image features are generated by projecting the points onto the images with a camera projection matrix. For each image, we retain the features that are found within the defined image size. The image features are perturbed with Gaussian noise to make the simulation more realistic. Figure 4 (Middle) and (Bottom) show the estimated camera poses and 3D map points from Local BA and our visual SLAM algorithm. A window size of 3 is used for the Local BA. Note that the plots have different scales because the absolute scale could not be recovered from a single camera. We can see that the estimated camera poses from the Local BA drifts significantly and the reconstructed 3D map points do not lie on a plane. In comparison, there is almost no drift in the camera poses estimated from our visual SLAM algorithm and the reconstructed 3D map points lie on a plane which are more consistent with the groundtruth.

Figure 5 (Right) shows the laboratory setup where image data is collected to verify our visual SLAM algorithm for the single plane case. Unique markers from the ARToolKitPlus software [20] are placed on the ground to increase the number of image features. These regular-shaped markers also make it easier for us to visually inspect the accuracy the map produced from our visual SLAM algorithm. Images are collected from a downward looking PointGrey Firefly MV [1] USB camera mounted on our quadrotor shown in Figure 5 (Left) overlooking the markers over a trajectory of approximately 930cm.

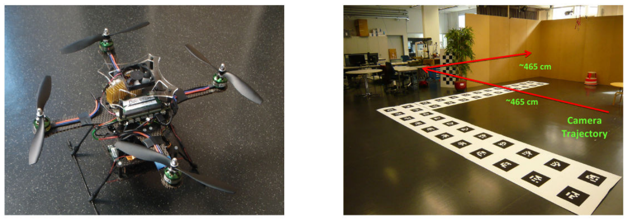

Fig. 5. (Left) Our Quadrotor with a downward looking camera. (Right) Laboratory setup where we collect visual data to verify our visual SLAM algorithm.

Figure 6 (Top) shows the camera trajectory and 3D map points estimated from the 5-point and linear 4-point algorithms. It can be seen from the plots that the map becomes less consistent with the ground plane as the camera advances through its trajectory. This is because the errors from the 5-point and linear 4-point algorithms are left unchecked and accumulates with the advances of the camera trajectory. We also observed from the arbitrary rotated side-view of the map that a significant number of map points do not lie on a plane. The L_2 norm of the distances between all points and their best fit plane is 111.7155 units (we do not assign any SI units for this distance since the scale of the reconstructed points is not known) and the L_2 norm of the reprojection errors is 113.3352 pixels.

Figure 6 (Bottom) shows the camera trajectory and 3D map points after our visual SLAM algorithm. Our visual SLAM algorithm is performed after pose estimation for every frame. It is obvious from the plots that our visual SLAM algorithm produces more consistent results with the ground plane. On top of being able to correct the errors from the pose estimation, our visual SLAM algorithm is also able to produce 3D map points that lies on a plane. This means that the 3D map points from our visual SLAM algorithm is closer to the groundtruth. The L_2 norm of the distances between the points and their best fit plane is 35.2910 units and L_2 norm of the reprojection errors is 103.4431 pixels.

We also did a comparison of the processing time needed for full BA and our visual SLAM algorithm. In this case, BA is performed after pose estimation for every frame. Figure 7 shows the recorded processing time from our Matlab implementations of BA and our visual SLAM. It is clear from the plot that the processing time for BA increases after the addition of every frame, but the processing time for our visual SLAM algorithm remains approximately constant.

V. Multiple Planes

We show theoretically in this section that our visual SLAM algorithm can be extended to handle environments with multiple planes.

[1]http://www.ptgrey.com/

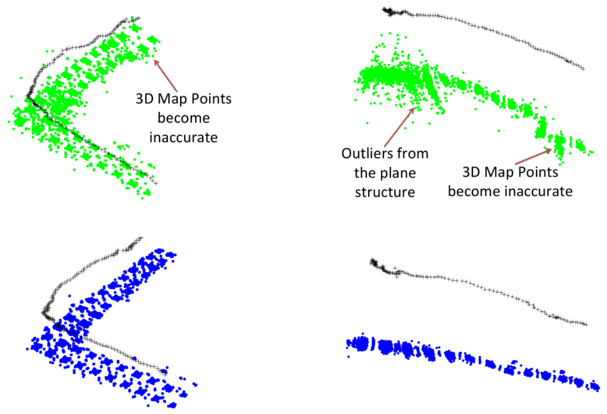

Fig. 6. (Top) Camera trajectory and 3D map points estimated with the 5-point and linear 4-point algorithms. (Bottom) Camera trajectory and 3D map points estimated from our visual SLAM algorithm.

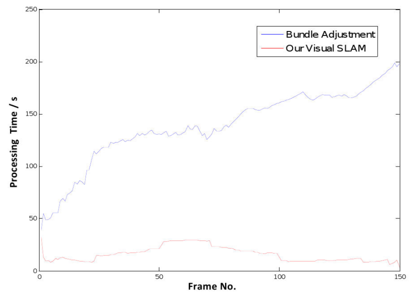

Fig. 7. The processing time for BA increases with each frame while the processing time for our visual SLAM algorithm stays approximately constant.

A. Initialization

In cases where there are more than one plane, we first do plane extraction on the images using methods such as [21] and classify the extracted features according to the plane it lie on. Next, the respective plane parameters are estimated with RANSAC after triangulation. Lastly, the relative pose, map points and plane parameters are optimized by minimizing the reprojection errors and the perpendicular distances of the map points and their respective plane. The cost function is given by

$$\underset{C_1,C_2,X,\Pi}{\operatorname{argmin}} \sum_{j}^{m} \left\{ \sum_{i=1}^{2} d\left(Q\left(C_i, X_j\right) - x_{ij}\right) + \sum_{k}^{p} d_{\perp}\left(\pi^k, X_j\right) \right\} \tag{8}$$

where $\Pi = [\pi^1, \pi^2, ..., \pi^p]$ and $\pi^p = [\pi_1^p, \pi_2^p, \pi_3^p, \pi_4^p]$ are the plane parameters. Note that $d_{\perp}(.) = 0$ if X_j does not belongs to the plane π^p. Similar to the single plane case, Equation 8 can be minimized with the Levenberg-Marquardt algorithm. The parameters which we try to estimate is now given by $\beta = [C_1, C_2, X, \Pi]^T$, and the error $\epsilon = [\epsilon_d, \epsilon_{d_{\perp}}]^T$ where ϵ_d is the reprojection errors and $\epsilon_{d_{\perp}}$ is the perpendicular distances of each 3D map feature from its corresponding plane. The Jacobian is given by

$$J = \begin{pmatrix} J_C & J_X & 0 \\ 0 & J_{X\Pi} & J_{\Pi} \end{pmatrix} \quad (9)$$

where $J_{X\Pi}$ and J_{Π} are the Jacobians of $d_{\perp}(.)$ with respect to the 3D map points and parameters from the multiple planes.

B. Subsequent Pose Estimations

Subsequent pose estimations for multiple plane case is the same as the single plane case (Section III-C) except for the need to detect and add new planes. Planes are detected with plane extraction on the image and are classified as new planes if all the 3D points found on it are newly added points.

C. Bundle Adjustment with Plane Constraint

BA for the multiple planes case is preceded by individual RANSAC process to estimate the parameters Π for the multiple planes from all existing map points. This is followed by a BA that estimates the current camera pose and newly seen map points by minimizing the reprojection errors and perpendicular distances between the newly seen points and their respective planes. The cost function is given by

$$\underset{C_n, X^{new}}{\operatorname{argmin}} \sum_{j=1}^{m''} \left\{ \sum_{i=n-1}^{n} d\left(Q\left(C_i, X'_j\right) - x'_{ij}\right) + \sum_{k}^{p} d_{\perp}\left(\pi^k, X'_j\right) \right\} \quad (10)$$

Note that $d_{\perp}(.) = 0$ if X_j does not belongs to plane π^p. The Levenberg-Marquardt algorithm is used to minimize the cost function in Equation 10. Similar to the single plane case, only the current camera pose and newly seen points given by $\beta = [C_n, X_1^{newT}, X_2^{newT}, ..., X_{m'}^{newT}]^T$ are optimized. The error is $\epsilon = [\epsilon'_d, \epsilon'_{d_{\perp}}]^T$ where ϵ'_d is the repojection errors for all X' in C_{n-1} and C_n, and $\epsilon'_{d_{\perp}}$ is the perpendicular distances of all X' with their corresponding plane parameters π^k estimated prior to the BA. The Jacobian is given by

$$J = \begin{pmatrix} J_{C_n} & J_{X^{new}} \\ 0 & J_{X^{new}\Pi} \end{pmatrix} \quad (11)$$

where $J_{X^{new}\Pi}$ is the Jacobian of $d_{\perp}(.)$ with respect to the newly seen points.

The computation cost of our visual SLAM algorithm in the multiple plane case remains approximately the same because the number of parameters to be estimated remain as $(6+3m')$ in the BA. The only part that incurs additional computation cost is the RANSAC processes required to estimate $4p$ parameters from the multiple planes, which however does not increases the computation cost by a huge amount.

VI. CONCLUSIONS

In this paper, we proposed a visual SLAM algorithm for MAVs navigating in the urban environments with a consistent ground plane. We showed from simulation and implementation results that our algorithm is able to enforce planar consistency between the reconstructed map points and the actual ground plane in approximately constant time. We also showed theoretically that an extension of our algorithm to handle multiple planes is possible.

VII. ACKNOWLEDGMENTS

This work was supported in part 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] "MIT Robust Robotics Group", http://groups.csail.mit.edu/rrg/index.html.

[2] "Ascending Technologies GmbH", http://www.asctec.de/.

[3] "Swarm of Micro Flying Robots: sFly", http://projects.asl.ethz.ch/sfly/doku.php.

[4] G Sibley, C Mei, I Reid and P Newman, "Adaptive Relative Bundle Adjustment", *in Robots: Science and Systems (RSS)*, June 2009.

[5] Zhengyou Zhang and Ying Shan, "Incremental Motion Estimation Through Local Bundle Adjustment", *in Internation Conference on Image Processing (ICIP)*, September 2003.

[6] Yi Ma, Stefano Soatto, Jana Kosecka and Shankar S. Sastry, "An invitation to 3-D vision: From images to geometrical approaches", *Springer-Verlag*, November 2003.

[7] Hartley, R. I. and Zisserman, A., "Multiple View Geometry in Computer Vision", Second Edition, *Cambridge University Press*, 2004.

[8] Bill Triggs, P. McLauchlan, Richard Hartley and A. Fitzgibbon, "Bundle Adjustment – A Modern Synthesis", *in Vision Algorithms: Theory and Practice, Lecture Notes in Computer Science*, pp 298–372, 2000.

[9] Andrew J. Davison, "Real-Time Simultaneous Localization and Mapping with a Single Camera", *in International Conference on Computer Vision (ICCV)* , 2003.

[10] Ethan Eade and Tom Drummond, "Scalable Monocular SLAM", *in IEEE Computer Vision and Pattern Recognition (CVPR)*, 2006.

[11] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens and Eduardo Nebot, "Consistency of the EKF-SLAM Algorithm", *in IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[12] Tim Bailey, Juan Nieto and Eduardo Nebot, "Consistency of the FastSLAM Algorithm", *in IEEE International Conference on Robotics and Automation (ICRA)*, 2006.

[13] D. Nister, "An efficient solution to the five-point relative pose problem", *in Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, Volume 2, pages. 195-202, 2003.

[14] Fischler, Martin A. and Bolles, Robert C.,"Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography", *in Readings in computer vision: issues, problems, principles, and paradigms*, 1987, pp. 726–740.

[15] David G. Lowe, "Distinctive image features from scale-invariant keypoints", *in International Journal of Computer Vision*, 2004, pp. 91-110.

[16] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool, "SURF: Speeded Up Robust Features", *in Computer Vision and Image Understanding (CVIU)*, Volume 110, Nr. 3, pp. 346–359, 2008.

[17] Vincent Lepetit and Pascal Fua, "Keypoint Recognition Using Randomized Trees", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 28, Issue 9, September 2006.

[18] M. Özuysal, M. Calonder, V. Lepetit, P. Fua, "Fast Keypoint Recognition using Random Ferns", *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 32, Nr. 3, pp. 448–461, March 2010

[19] Long Quan and Zhongdan Lan, "Linear N-point camera pose determination", *in IEEE Pattern Analysis and Machine Intelligence*, 1999.

[20] Wagner Daniel and Schmalstieg Dieter, "ARToolKitPlus for Pose Tracking on Mobile Devices", *Proceedings of 12th Computer Vision Winter Workshop*, February 2007.

[21] MIA Lourakis, AA Argyros, SC Orphanoudakis, "Detecting planes in an uncalibrated image pair", *in British Machine Vision Conference (BMVC)*, 2002.

[22] Georg Klein and David Murray, "Parallel Tracking and Mapping for Small AR Workspaces", *in International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.