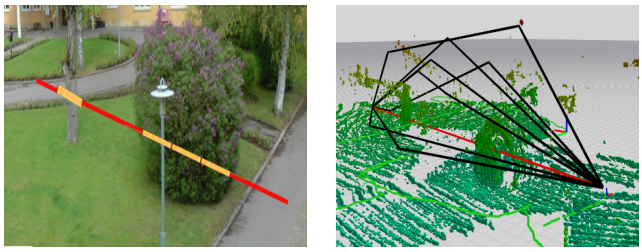
Outdoor Obstacle Avoidance based on Hybrid Visual Stereo SLAM for an Autonomous Quadrotor MAV

Radouane Ait-Jellal and Andreas Zell

***Abstract*— We address the problem of on-line volumetric map creation of unknown environments and planning of safe trajectories. The sensor used for this purpose is a stereo camera. Our system is designed to work in GPS denied areas. We design a keyframe based hybrid SLAM algorithm which combines feature-based stereo SLAM and direct stereo SLAM. We use it to grow the map while keeping track of the camera pose in the map. The SLAM system builds a sparse map. For the path planning we build a dense volumetric map by computing dense stereo matching at keyframes and inserting the point clouds in an Octomap. The computed disparity maps are reused on the direct tracking refinement step of our hybrid SLAM. Safe trajectories are then estimated using the RRT* algorithm in the SE(3) state space. In our experiments, we show that we can map large environments with hundreds of keyframes. We also conducted autonomous outdoor flights using a quadcopter to validate our approach for obstacle avoidance.**



(a) (b)

Fig. 1. (a) Our outdoor environment. The red line shows the straight line between the start and destination positions going through two obstacles (trees). (b) the 3D reconstruction using our stereo camera and a set of collision-free 3D paths generated by our system.

I. INTRODUCTION

Quadcopters are mobile robots which are suited for many robotic applications. Building inspection, agricultural fields surveillance and package delivery are some applications of quadcopters, just to name a few examples. Unlike fixed wing flying robots, quadcopters can achieve high manoeuvrability about all three axes and in all directions and they have the ability to hold the position (hovering). To achieve full autonomous flights, a quadcopter should rely only on its on-board sensors. Limited by the payload it can carry and by the real-time requirements for safe navigation of a quadcopter, many sensors are not suitable for autonomous quadcopters. Laser scanners which have sufficient frame rates and scan-lines are usually too heavy to be carried by a small quadcopter and they have high power consumption as well. The requirement to work in outdoor environments excludes the choice of using IR pattern based RGBD sensors as main sensors since, with substantial infra-red light from the sun, these sensors fail to estimate the depth images. Stereo cameras can provide data at high frame rates, they are lightweight, passive (do not illuminate the scene), energy efficient and customizable. Given enough data to build dense volumetric maps, the quadcopter can safely navigate in cluttered environments by avoiding obstacles. However, we need to run dense stereo matching on the on-board CPU to estimate the disparity map and use this disparity map to estimate the depth. To allow real time operation, the overhead, which is introduced by the on-board computation of the disparity, can be minimized by computing the disparity only at keyframes and by reducing the resolution. The volumetric mapping of large unknown space requires usage of SLAM (simultaneous localization and mapping) designed to operate at large scale and a memory efficient representation of the dense volumetric map. In this work, we design a hybrid SLAM that works in large scale environments based on the open-source ORB-SLAM2 [1] and for the memory efficient volumetric map we use Octomap [2] [3]. For efficient 3D path planning, we use the RRT* [4] algorithm. The output of the 3D planner is then set as desired way-points for our quadcopter micro aerial vehicle (MAV).

In this work, we propose a system for 3D outdoor obstacle avoidance using stereo. This system was tested on a quadcopter MAV and the quadcopter was able to run all software packages, shown in Fig. 2, in real-time. In addition to the module for 3D obstacle avoidance, we propose a hybrid visual stereo SLAM algorithm which combines a feature-based method and a direct image alignment method. Our choice is motivated by three reasons: first, we want to use more information from the images and not only abstract them to a set of features. Second, since we compute the relatively costly dense stereo matching for the keyframes, we want to benefit from that not only for building volumetric maps but also for refining the tracking poses. Third and finally, in some aspects feature-based methods and direct methods are complementary. So by combining them they compensate for each other's drawbacks. For example, when the stereo camera has moved over large distances between two frames the direct methods might diverge, since they need a good initial guess. In the best case they might need considerably more iterations to converge. This might make real time operation difficult to achieve. Using a coarse-to-fine approach and/or a

R. Ait-Jellal is with the Chair of Cognitive Systems, headed by Prof. A. Zell, Computer Science Department, University of Tübingen, Sand 1, D-72076 Tübingen, Germany {radouane.ait-jellal@uni-tuebingen.de, andreas.zell@uni-tuebingen.de}

motion model might help in some cases. On the other hand, the feature based approach can deal very efficiently with large camera movements. Having to deal with very sparse features, we can afford to enlarge the search domain for correspondences, while still keeping real-time capabilities. The benefit of using direct methods rather than feature based methods is in cases of degraded image quality due to camera defocus and motion blur. In these cases, feature extraction might fail, causing tracking loss while the image alignment would give a reasonably good motion estimation.

II. RELATED WORK

A basic capability that a mobile MAV should fulfill for safe navigation is obstacle avoidance. Heng et al. [5] have proposed a stereo approach for obstacle avoidance. They build an Octomap and use the anytime dynamic A* planner [6] to achieve collision-free path planning in 2D. For pose estimation they use either artificial landmarks or a Vicon tracking system. In our approach, we plan paths in full 3D-space using the efficient RRT* [4] algorithm and we run a hybrid SLAM on-board. The feature based part of our hybrid visual stereo SLAM system uses the popular ORB-SLAM2 [1] algorithm. The ORB-SLAM2 algorithm splits the tracking from the local mapping using two separate CPU threads. This architecture was initially proposed by Klein et al. [7] in their popular parallel tracking and mapping (PTAM) algorithm. The PTAM algorithm was then successfully used by many mobile robotic researchers [8] [9]. The ORB-SLAM2 algorithm starts by extracting a (sparse) set of ORB [10] features on both left and right images of the current stereo frame. The features are then matched on the previous stereo frame to establish 2D-2D correspondences. Using the map we get 2D-3D correspondences. A PnP solver is then used to optimize the initial pose. Using the co-visibility graph (a graph which connects keyframes which share enough map points) a local map is extracted and the pose gets refined by using more map points. The local map contains the keyframes and the map points observed by these local keyframes. The final pose we get from ORB-SLAM2 is estimated using only sparse features. Valuable information might be missed by abstracting the images to a set of ORB features. We propose to further refine the pose by using direct image alignment and use more data from the image. We do not use all of the pixels but only those which have valid depth (from dense stereo matching) and have an image gradient larger than a given threshold. Direct methods for SLAM are becoming popular, and efficient implementations exist [11] [12] [13]. The afore-mentioned implementations use the forward additive or forward compositional [14] variants of the Lucas Kanade [15] [14] algorithm. We use the more efficient inverse compositional alignment algorithm [16], which allows the pre-computation of the Jacobian (and Hessian). Forster et al. [17] introduced sparse direct image alignment where they use the inverse compositional algorithm to align a set of sparse features. They extend their approach for multiple cameras in the recent work [18]. Unlike [17] and [18] we build a hybrid SLAM and not only a visual odometry system.

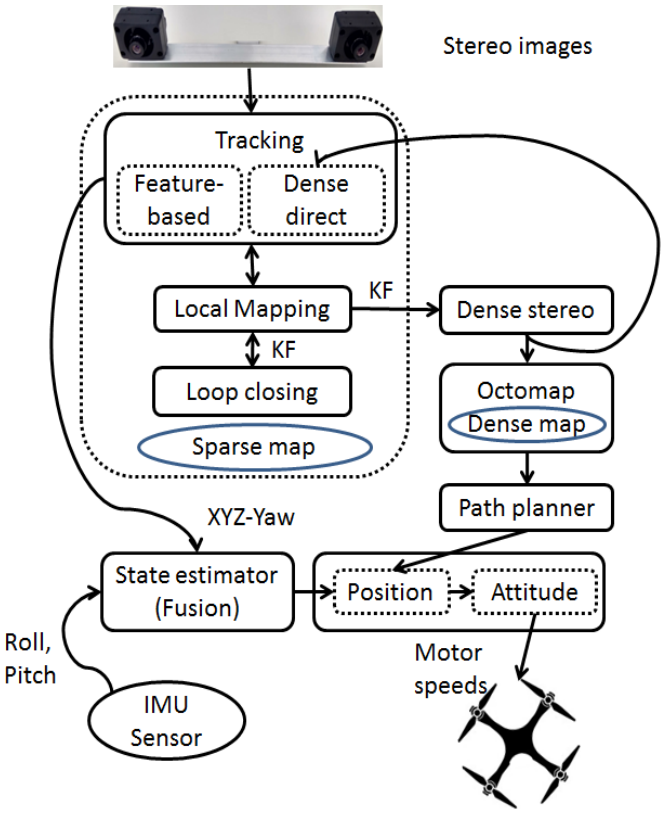


Fig. 2. System overview. The system includes a hybrid SLAM algorithm and a path planner. The tracking thread of the SLAM algorithm provides an estimate of the current pose to the controller. The path planner provides intermediate destinations to the controller. The Pixhawk position controller manages to fly smoothly through the intermediate way-points.

III. OVERVIEW

A. Block diagram of the proposed system

Fig. 2 shows an overview of our system. The hybrid stereo SLAM algorithm is in charge of keeping track of the quadcopter pose in real time while at the same time building a sparse map of the environment. At keyframes we compute dense stereo matching. The resulting disparity map is then used to create a 3D point cloud which is inserted in the occupancy grid map. The disparity map is also used by the tracking thread to refine the poses using direct image alignment. The planner gets an up to date binary occupancy grid map and the start (current) pose and destination pose and then plans a safe path for obstacle avoidance. The planner outputs smooth trajectory of way-points. The poses from the localization thread of the SLAM system are fused with the measurements from the on-board IMU using an Extended Kalman Filter (EKF) to estimate the quadcopter state. The controller gets the desired pose and the current pose and controls the motor speeds to fly to the desired position. The Pixhawk [19] controller is a cascaded controller which includes a high level position controller and a low level

attitude controller. The low level controller outputs the motor speeds.

B. Least squares problems

The least squares (LS) algorithm is used for optimizing non-linear cost functions in many parts of our system. We use least squares in the feature based tracking to optimize re-projection distances. Then, we use least squares in the direct image alignment to minimize photometric errors (intensity differences). On the local mapping thread, least squares are used in local bundle adjustment (LBA) to locally optimize the SLAM map. On the SLAM back-end, least squares are used to optimize a pose graph when loops are detected and afterwards to optimize the whole system (all keyframe poses and all map points) by global bundle adjustment (GBA). In general, given an error function $e(x) = [e_1(x), ..., e_m(x)]^T \in \mathbb{R}^m$, where the variable $x = [x_1, ..., x_n]^T \in \mathbb{R}^n$ is a vector of parameters. We seek to estimate the optimal solution x^* that minimizes the energy function $E(x)$ (Eq. 1).

$$x^* = \underset{x}{\operatorname{argmin}} E(x) \tag{1}$$

$$= \sum_{i=0}^{m} |e_i(x)|^2 \tag{2}$$

$$= e(x)^\intercal e(x) \tag{3}$$

In general, e is non-linear so we compute an approximation using the first order Taylor expansion and solve the optimization problem iteratively. Around the initial guess $\breve{x}$ we consider a small perturbation δx :
$e(\breve{x} + \delta x) \approx e(\breve{x}) + J\delta x$, here J is the Jacobian of $e(x)$ computed in $\delta x = 0$ (in $x = \breve{x}$). The elements E_i of the energy function $E(x)$ can then be approximated such that:

$$E(\breve{x} + \delta x) = e(\breve{x} + \delta x)^T e(\breve{x} + \delta x) \tag{4}$$

$$\approx (e(\breve{x}) + J\delta x)^T (e(\breve{x}) + J\delta x) \tag{5}$$

$$= E(\breve{x}) + 2e(\breve{x})^\intercal J\delta x + \delta x^\intercal J^\intercal J\delta x \tag{6}$$

$$= E(\breve{x}) + 2e(\breve{x})^\intercal J\delta x + \delta x^\intercal H\delta x \tag{7}$$

Where $H = J^\intercal J$ is the approximate Hessian. By computing the derivative with respect to δx and setting it to zero we can compute the increment δx^* which minimizes Eq. 7 and solves the following linear system (Eq. 8):

$$H\delta x^* = -e(\breve{x})^\intercal J \tag{8}$$

This linear system can be written as:

$$A\mathrm{x} = b \tag{9}$$

where: $A = H$, $b = -e(\breve{x})^\intercal J$ and $\mathrm{x} = \delta x^*$. In our implementation we use Cholesky factorization to solve the linear system in Eq. 8. The increment δx^* is then added to the initial guess $\breve{x}$.

$$x^* = \breve{x} + \delta x^* \tag{10}$$

For the inverse compositional direct alignment step of our algorithm, the Hessian and its inverse can be pre-computed (for all iterations). Thus, solving the linear system in Eq. 8 becomes trivial. The parameter update is also different since it involves inverted composition rather than forward additive increments (See Eq. 26).
The Gauss Newton solver iterates the following steps: first, it locally approximates the non-linear cost function with a linear function according to Eq. 5, then it computes in closed form the increment δx^* according to Eq. 8, and finally it updates the parameter vector according to Eq. 10. While the intermediate problem (linear approximation) is solved in closed form, the initial non-linear problem in Eq. 1 needs to be iteratively solved. In every iteration, we use the current updated parameter vector as an initial guess (linearization point). The iterative process continue until some termination criteria are met, e.g., when the algorithm converges or we reach a maximum number of iterations.

IV. The Hybrid SLAM system

We propose a hybrid visual stereo SLAM algorithm which combines a feature-based method and a direct image alignment method. As discussed on the introduction, feature-based approaches are generally fast and can handle large camera movements. We make use of this characteristic and design a hybrid visual stereo SLAM which uses the motion estimation from a feature-based visual SLAM as an initial guess for a direct image alignment method refinement step. Our hybrid stereo SLAM is based on the popular ORB-SLAM2 algorithm [1]. We modify the tracking thread such that we refine the poses using direct image alignment and we modify the mapping thread such that we compute dense binocular stereo matching at keyframes.

A. Notation

We use the following notation in this work:

- $\{I^l_{cur}, I^r_{cur}\}$ and $\{I^l_{ref}, I^r_{ref}\}$: the left and right images of the current frame and reference keyframe respectively.
- $x = (u, v)$: pixel with the coordinates (u, v).
- $p = (X, Y, Z)$: point in 3D space corresponding to the image pixel $x = (u, v)$.
- $T \in SE(3)$: a 3D rigid body transform. $T = \{R, t\}$ where $R \in SO(3)$ and $t \in \mathbb{R}^3$.
- $\xi \in se(3)$: minimal parametrization of the 3D rigid body transform in the Lie algebra.
- T_f: transform estimated by the feature-based approach.
- T_d: transform estimated by the direct image alignment approach using the inverse compositional algorithm.
- $W(x, \xi)$: 3D warp which maps pixels from the reference keyframe to the current frame.
- ∇I: image intensity gradients (Jacobians).
- π, π^{-1}: pinhole camera projection model and its inverse.
- f_x, f_y, c_x, c_y: camera intrinsic parameters.
- ρ: Huber robust cost function.
- e: error function.
- J and H: Jacobian and Hessian.
- $\breve{x}$: initial guess for the parameter x which we use to initialize the optimization.

- x^*: optimal value of the parameter x, which we get after the optimization.

B. Lie algebra parametrization for motion estimation

A rigid body transform g transforms as point $p \in \mathbb{R}^3$ in the 3D space to a point $g(p) \in \mathbb{R}^3$ in 3D space:

$$\begin{aligned} g &: \mathbb{R}^3 \rightarrow \mathbb{R}^3 \\ p &\rightarrow g(x) \end{aligned} \tag{11}$$

The rigid body transform g can be expressed by a 4×4 matrix $T \in SE(3)$, which is composed by a rotation matrix $R \in SO(3)$ and a translation vector $t = (t_x, t_y, t_z) \in \mathbb{R}^3$.

$$g(p) = g(p, T) = T.p = Rp + t \tag{12}$$

where

$$T = \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{13}$$

and

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{14}$$

We use the Lie group $SE(3)$ and its corresponding Lie algebra $se(3)$ to get a minimal parametrization of the 3D rigid body transforms. On the associated Lie algebra $se(3)$, the corresponding transform is a 6-dimentional vector $\xi = (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6)$. These coordinates are called the twist coordinates, where the first (ξ_1, ξ_2, ξ_3) are the linear velocities and (ξ_4, ξ_5, ξ_6) are the angular velocities. Here, we represent the rotation part of the transform with exactly three parameters (ξ_4, ξ_5, ξ_6) rather than nine parameters as in the rotation matrices representation. The use of the Lie group solves the singularities of the compact Euler-angles representation.
A rigid body transform T can be mapped to its corresponding ξ using the logarithmic map as follow:

$$\begin{aligned} log &: SE(3) \rightarrow se(3) \\ g &\rightarrow \xi = log(T) \end{aligned} \tag{15}$$

The inverse of this operation is the exponential map.

$$\begin{aligned} exp &: se(3) \rightarrow SE(3) \\ \xi &\rightarrow T = T(\xi) = exp(\xi) \end{aligned} \tag{16}$$

C. Tracking: Minimizing distances and intensity differences

Our system uses ORB-SLAM2 to estimate ξ_f^*, the pose of the current frame based on features. Any other feature-based stereo SLAM can be used. We compute T_f by minimizing the re-projection error (see Eq. 18) between predicted pixel locations and the observed pixels locations (see Eq. 17).

$$e_f(x_i) = x_i - \pi(RX_i + t) \tag{17}$$

$$T_f^* = \underset{T_f}{\operatorname{argmin}} \sum_{i \in \aleph} \rho\left(\|x_i - \pi(RX_i + t)\|_{\Omega_i}\right) \tag{18}$$

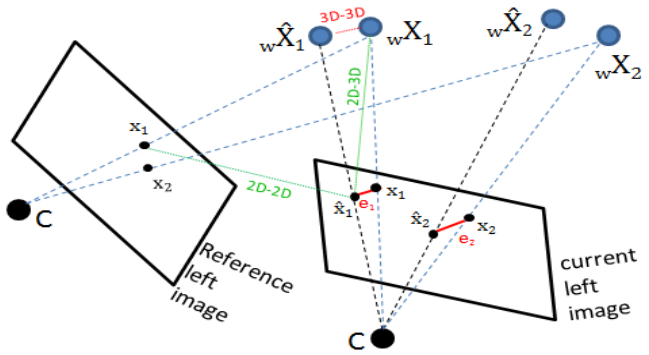


Fig. 3. Feature based motion estimation: we try to find the transform (pose of the current camera) for which the sum of all re-projection distances e_i (shown in red, the unit is Pixel) is minimal. Note that in a 3D-3D correspondence approach, distances (in Meter unit) between 3D points. Due to triangulation inaccuracy, these methods are generally less accurate than approaches based on 2D-3D correspondences.

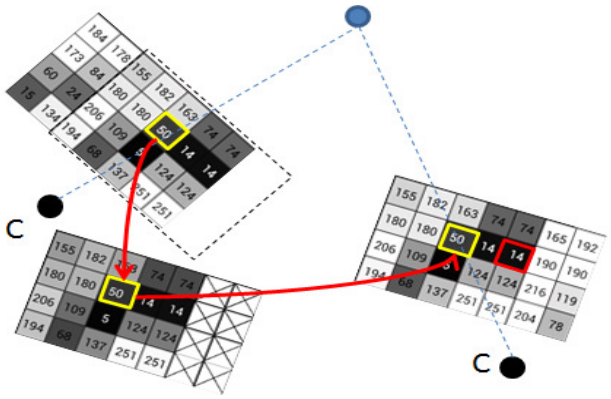


Fig. 4. Direct image alignment motion estimation: illustration of the inverse compositional algorithm. Here, we minimize the photometric errors. The color (gray values) of the squares encode the intensity value [0-255]. The search for the warp increment (image bottom left) is done in the reference image (top left) and then this warp is inverted and composed (Eq. 26) with the current warp of the target image (right image). The result, is a direct warp from the template image to the target image.

where Ω_i is a weighting (inverse covariance matrix) associated to the scale at which the feature was extracted. And ρ is the Huber robust cost function. We note that we minimize distances (in pixels) on the image plane. We illustrate this case in Fig 3. The final pose ξ_f^* we get with the feature based motion estimation is used as initial guess for our direct image alignment motion refinement. This is described in the following equation (Eq. 19):

$$\breve{\xi}_d = \xi_f^* \tag{19}$$

The 3D warp $W(x_i, \xi_d)$ which maps a pixel $x_i \in I_{ref}^l$ in the reference keyframe into its corresponding pixel in the image I_{cur}^l using the pinhole camera projection model π is given by:

$$\begin{aligned} W &: \mathbb{R}^2 \times \mathbb{R}^6 \rightarrow \mathbb{R}^2 \\ (x, \xi_d) &\rightarrow W(x, \xi_d) = \pi(\pi^{-1}(x, Z), g(T(\xi_d))) \end{aligned} \tag{20}$$

where

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{21}$$

and its inverse maps a 2D pixel $x = (u, v)$, with known depth Z, to its corresponding 3D point $p = (XYZ)$:

$$\pi^{-1}\left(\begin{bmatrix} u \\ v \\ Z \end{bmatrix}\right) = \begin{bmatrix} \frac{u-c_x}{f_x} Z \\ \frac{v-c_y}{f_y} Z \\ Z \end{bmatrix} \tag{22}$$

We compute T_d by minimizing the re-projection photometric error between the predicted intensities of the pixels and the observed intensities. We illustrate this case in Fig 4. We work with 8-bit gray-scale images with intensities in the interval [0-255]. The error is defined as follow (Eq. 23):

$$e_d(x_i) = I^l_{ref}(W(x_i, \delta\xi_d)) - I^l_{cur}(W(x_i, \xi_d)) \tag{23}$$

We optimize using pixels $x_i \in D_{ref}$ where D_{ref} is the set of pixels $x_i \in I^l_{ref}$ such that x_i has a valid depth and the magnitude of the intensity gradient at x_i is larger than a given threshold. In practice we use about 15% of the image pixels. Note that the domain D_{ref} here is defined in I^l_{ref} for the inverse compositional image alignment. For the forward additive algorithm and the compositional forward algorithm, the domain is defined on I^l_{cur}.
We iteratively optimize the following energy function to align I^l_{cur} with I^l_{ref}:

$$\xi^*_d = \underset{\xi_d}{\operatorname{argmin}} \sum_{x \in D_{ref}} \|I^l_{ref}(W(x_i, \delta\xi_d)) - I^l_{cur}(W(x_i, \xi_d))\|^2 \tag{24}$$

$$= \underset{\xi_d}{\operatorname{argmin}} E_d(\delta\xi_d) \tag{25}$$

Since the increment $\delta\xi_d$ is computed for the reference keyframe image (the template image not the target image) we need to invert it and and compose it with the current parameter estimate. The warp update at iteration $k+1$ is given by Eq. 26.

$$T^{k+1}_d = T^k_d * exp(-\delta\xi^*_d) \tag{26}$$

In the iterative optimization process, we linearize around $\delta\xi = 0$ at every iteration using Eq. 5 applied to the cost function in Eq. 24.

$$E_d \approx \sum_{x \in D_{ref}} \|I^l_{ref}(W(x, 0)) - I^l_{cur}(W(x, \xi_d)) + J_d \delta\xi_d\|^2 \tag{27}$$

The derivation of the Jacobians for our inverse compositional algorithm is similar to the forward compositional algorithm in [20]. However, Jacobians in our case are computed for the I^l_{ref} image and not for I_{cur} as in [20].

$$J_d(x, \xi_d) = \frac{\partial E_d}{\partial \xi_d} \tag{28}$$

The Jacobian can be computed using the chain rule.

$$\begin{aligned} J_d(x, \xi_d) =& J_I J_\pi J_g J_T \\ =& \frac{\partial I^l_{ref}(W(x, \delta\xi_d))}{\partial \pi}\Big|_{x=\pi(g(p_i,T(0))=x_i} \cdot \\ & \frac{\partial \pi(p)}{\partial g}\Big|_{p=g(p_i,T(0))=p_i} \cdot \\ & \frac{\partial g(p,T)}{\partial T}\Big|_{\substack{T=T(0)=I^l_{cur}, \\ p=p_i}} \cdot \\ & \frac{\partial T(\xi_d)}{\partial \xi_d}\Big|_{\xi_d=0} \end{aligned} \tag{29, 30}$$

The individual Jacobians in Eq. 28 can be derived as follow. The intensity Jacobian J_I is evaluated at $x = \pi(g(p_i, T(0))$, where $T(0) = I_{4x4}$.

$$\begin{aligned} J_I =& \frac{\partial I^l_{ref}(W(x, \delta\xi_d))}{\partial \pi}\Big|_{x=\pi(g(p_i,T(0))=x_i} && (31) \\ =& (\nabla I^l_{ref,u} \nabla I^l_{ref,v}) && (32) \end{aligned}$$

The camera projection Jacobian is:

$$\begin{aligned} J_\pi =& \frac{\partial \pi(p)}{\partial g}\Big|_{p=g(p_i,T(0))=p_i} && (33) \\ =& \begin{bmatrix} f_x \frac{1}{z} & 0 & -f_x \frac{x}{z^2} \\ 0 & f_y \frac{1}{z} & -f_y \frac{y}{z^2} \end{bmatrix} && (34) \end{aligned}$$

The Jacobian of the function g is:

$$\begin{aligned} J_g =& \frac{\partial g(p,T)}{\partial T}\Big|_{\substack{T=T(0)=I^l_{cur}, \\ p=p_i}} \\ =& \begin{bmatrix} x.I_{3\times3} & y.I_{3\times3} & z.I_{3\times3} & I_{3\times3} \end{bmatrix} && (35) \end{aligned}$$

The Jacobian of T is:

$$\begin{aligned} J_T =& \frac{\partial T(\xi_d)}{\partial \xi_d}\Big|_{\xi_d=0} && (36) \\ =& \begin{bmatrix} & 0 & 0 & 0 \\ 0_{3\times3} & 0 & 0 & 1 \\ & 0 & -1 & 0 \\ & 0 & 0 & -1 \\ 0_{3\times3} & 0 & 0 & 0 \\ & 1 & 0 & 0 \\ & 0 & 1 & 0 \\ 0_{3\times3} & -1 & 0 & 0 \\ & 0 & 0 & 0 \\ \\ I_{3\times3} & & 0_{3\times3} & \end{bmatrix} && (37) \end{aligned}$$

Finally, we get the expression for our per-pixel Jacobian:

$$\begin{aligned} J_d =& J_I J_\pi J_g J_T && (38) \\ =& (\nabla I^l_{ref,u} f_x, \nabla I^l_{ref,v} f_y) \cdot \\ & \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} & -\frac{xy}{z^2} & (1+\frac{x^2}{z^2}) & -\frac{y}{z} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} & -(1+\frac{y^2}{z^2}) & \frac{xy}{z^2} & \frac{x}{z} \end{bmatrix} && (39) \end{aligned}$$

Note that the per-pixel Jacobian J_d is a 6-dimensional vector. By checking the dimensions of the different matrices which

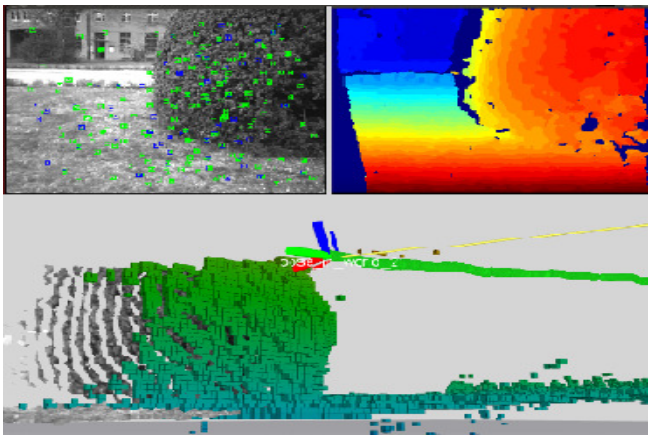


Fig. 5. This figure shows the features tracked on the current frame, the disparity map of the last keyframe (the color encode the disparity value) and a view of the volumetric reconstruction. The current point cloud which overlaps with the map is also drawn.

compose J_d, we get:

$$\underbrace{J_d}_{1\times 6} = \underbrace{J_I}_{1\times 2} \cdot \underbrace{J_\pi}_{2\times 3} \cdot \underbrace{J_g}_{3\times 12} \cdot \underbrace{J_T}_{12\times 6} \tag{40}$$

V. Outdoor 3D Obstacle Avoidance using Stereo

In this section we describe our stereo-based 3D path planning system. A volumetric map (Octomap) is maintained by the system. Stereo dense disparity maps are projected to 3D to create point clouds (Octomap scans). These measurements are then used to update the octree [21]. Collision-free trajectories can then be planned for safe navigation in 3D.

A. Occupancy grid map and dense stereo matching

The map built by ORB-SLAM2 is too sparse to be used for path planning. Thus, we build a dense volumetric occupancy grid map. For real-time purposes we update the occupancy grid map only at keyframes. When the tracking thread of the SLAM system decides to create a new keyframe we also store (with the keyframe) the intensity images (left and right at half resolution). At the mapping thread we estimate the disparity map by using the popular semi-global matching (SGM) dense stereo matching [22]. While there exist more efficient local stereo matching algorithms [23] [24], the need to estimate the disparity maps only at keyframes motivates us to use the globally more accurate stereo algorithm SGM.

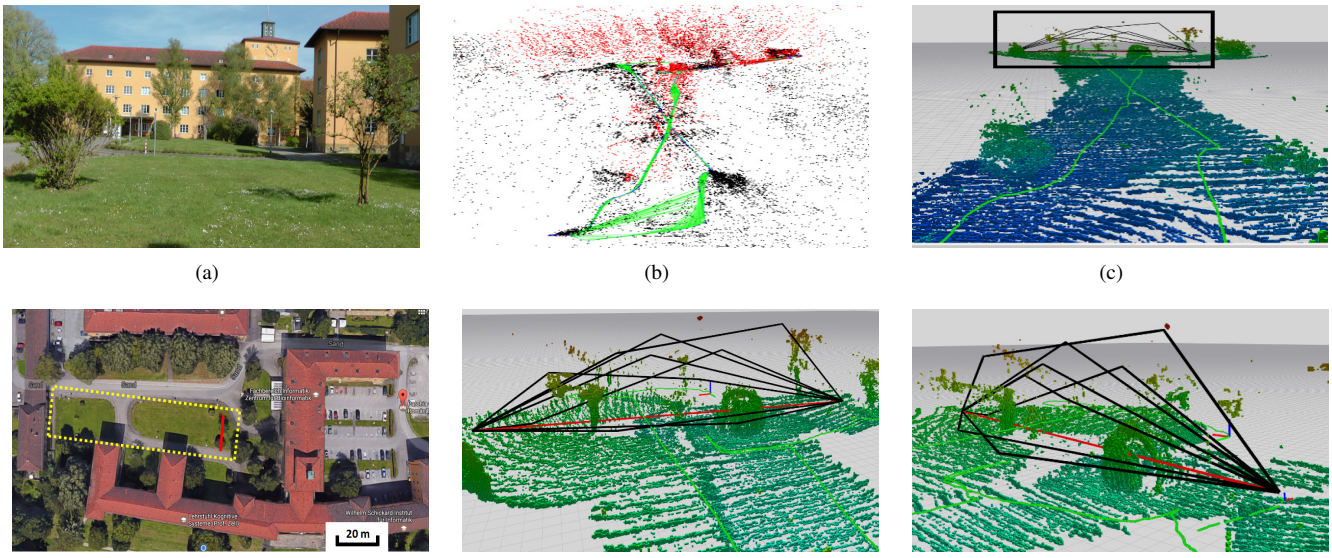
B. Path planning in 3D occupancy grid map

The 3D path planner gets a binary version of the up-to-date octree and updates its bounds using the current octree bounding boxes. The planning in 3D is more challenging than in 2D and efficient methods need to be used. Standard path planners such as A^* are not efficient for usage on-board a quadcopter. We choose to use a variant of the efficient rapidly-exploring random trees RRT [25]. We use the open-source library OMPL, which implements a set of planning algorithms. One of the algorithms used for path planning in this setup is RRT* [26]. It is a variant of the original RRT. In the following the choice of this algorithm is illustrated. RRT is a sampling-based algorithm and provides *probabilistic completeness* [26]. That means that the probability that the planner fails to find a solution, if there is one, goes to zero as the number of samples approaches infinity. Such a sampling-based planner requires a collision checking module as it does not represent obstacles explicitly [26]. A problem of RRT is that it doesn't provide any guarantees to an optimal solution. If we use RRT in this setup we get a valid solution very fast but as stated it is not necessarily optimal. Experiments show that it is very often not even close to optimal. But as we are planning paths for a Quadrotor optimal paths (or at least short paths) are important to not waste battery power. To achieve an optimization of the planned path RRT* is used. Basically the tree is constructed in the same manner as in normal RRT but not all feasible connections will be inserted. Normal RRT just inserts a connection between the new node and its nearest neighbor (considering the euclidean distance). The RRT* algorithm will check all nodes in the surrounding of a new node and only insert the shortest path to the new node, considering a cost function, into the tree. This cost function differs from the euclidean distance, because on the path from the root to the node, which will be connected to the new node, there might be some obstacle that increases the cost in comparison to a straight line connection. Therefore the nearest neighbor of the new node does not have to be on the shortest path to the new node. After that all the surrounding nodes are checked again whether there is a new shortest path to each of them using the newly inserted node. If that is the case the tree gets rewired to maintain it a tree structure. As stated in [26], RRT* is *probabilistically complete*, like the normal RRT, but moreover it is *asymptotically optimal*. That means that the returned solution converges almost surely to the optimal path [26]. This property made the RRT* algorithm a good choice for this path planning scenario. Moreover the algorithm is not limited to finding geometric shortest paths but can also optimize towards a mixed optimization objective taking different costs (e.g. avoiding power extensive maneuvers) into account. In the quadcopter online planning scenario it has to be considered that one has to make a trade-off between spending energy and time flying a non-optimal path vs. spending energy and time hovering too long to compute the optimal path plus flying this path. Once a path is planned the way-points are sent to the Pixhawk controller via a serial connection. The Pixhawk controller takes care of flying the quadcopter to the desired destination. The desired destination includes the position (XYZ) and the yaw angle.

VI. Results

A. Platform description

In our real experiments we used a custom-made quadcopter. It is shown in Fig. 7. The components of the quadcopter are as follows: a stereo camera with a pair of Point-Grey Firefly monochrome cameras with a resolution of 640x480 pixels. The baseline between the two camera is 22cm. The stereo camera delivers synchronized stereo pairs at 30Hz. The flight control is from the open-source project



(a) (b) (c)

(d) (e) (f)

Fig. 6. Thanks to the volumetric global map maintained by our algorithm, we can plan collision-free paths between any two positions on the global map. (a) a view of the outdoor environment in which we did the experiment. (d) a Google Maps view of this area. The yellow rectangle shows the mapped area and the red line inside it shows the straight line between the start and goal position for the planner. (b) the sparse map built by the SLAM algorithm. This map which is used for pose estimation and re-localization. (c) the 3D volumetric map (Octomap) which is used by the planner. The green dots show the robot trajectory. (e) and (f) show two views of a set of 3D paths between the start and destination positions. These views correspond to the black rectangle region of the map in (c).

Fig. 7. Quadcopter used for our outdoor real experiments with the forward looking stereo camera.

#Stereo pairs	9040
#Keyframes	1128
#Map points	30616
Feature based tracking (time)	52 ms
Direct alignment refinement (time)	21 ms
Dense stereo 320 × 240 pixels (time)	55 ms
Dense stereo 640 × 480 pixels (time)	183 ms
Octree update 320 × 240 pixels (time)	62 ms
Octree update 640 × 480 pixels (time)	191 ms
Octree memory (size)	360 MB
Binary Octree (size)	1.1 MB
Planning RRT* (time)	10 s
Average #WPs	86

TABLE I

SOME STATISTICS FROM THE OUTDOOR OBSTACLE AVOIDANCE EXPERIMENT.

Pixhawk [19]. The on-board computer is an Intel NUC mini-pc. It has an Intel core i7-5557U CPU with 2x3.1GHz, 4MB cache, 28 Watts thermal design power (TDP) and 8GB RAM.

B. Obstacle avoidance in outdoor environment

We performed outdoor experiments in an environment with vegetation (grass), trees, walls and asphalt. Some pictures of this environment can be seen in Fig. 1(a), 6(a) and 6(d). Fig. 6 shows the details of an outdoor experiment. It shows the sparse map and the volumetric map. A set of collision-free paths between two points are also shown. Table I shows some statics from the outdoor experiment. The running times are reported for the case where all software packages of our system (see Fig. 2) are running at the same time. The running times for performing dense stereo matching and octree update are given for two different resolutions (640×480 and 320×240). The planning time is set to 10 seconds. WPs means the number of intermediate way-points.

C. Results of the Hybrid SLAM on the Kitti Dataset

We evaluated our hybrid SLAM method on the Kitti odometry benchmark [27]. The Kitti datasets are recorded using a car with a stereo camera mounted on the top of the vehicle. The recorded trajectories have a total length of about 39 Km divided into 22 sequences. Ground truth trajectories are provided for the first 11 sequences. Some sequences include loop closures and dynamic objects (cars

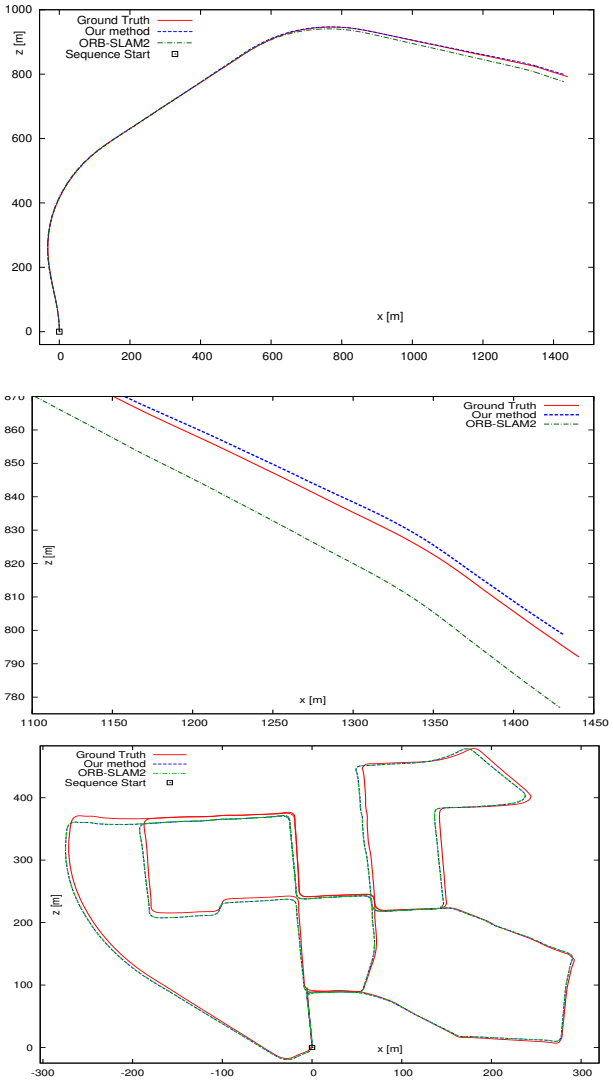


Fig. 8. Results on the Kitti odometry benchmark. Our algorithm (blue) is compared with the ORB-SLAM2 [1] (green). The ground truth is also drawn (red). Top: sequence 12. We achieve a considerable improvement. Our method accumulates less drift. Middle: for a better visualization we zoom in the last part of the trajectories of the previous image. Bottom: sequence 00 which has many loops. No significant improvement. The trajectories almost overlap.

driving around). Our results show that the refinement step (using the direct image alignment) improves the accuracy for sequences (e.g. sequence 12) with no loop closures. Fig. 8 shows the results obtained for the Kitti sequences 12 and 00. For the sequence 12 of the Kitti benchmark which has no loop closure the drift becomes larger with the travelled distance. Our refinement step helped to keep the drift very low. For sequences with many loop closures (e.g. sequence 00), we get almost the same results as the original algorithm. A loop closure detection and correction contribute to reduce the drift for the feature-based SLAM more than it does for our hybrid method. This can be confirmed by disabling the loop closure thread of the SLAM system.

REFERENCES

[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[2] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and B. W., "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," *ICRA*, 2010.

[3] K. Schauwecker and A. Zell, "Robust and efficient volumetric occupancy mapping with an application to stereo vision," *ICRA*, pp. 6102–6107, 2014.

[4] "The open motion planning library. ompl.kavrakilab.org."

[5] 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," *ICRA*, 2011.

[6] M. Likhachev, D. Ferguson, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," *International Conference on Automated Planning and Scheduling*, 2005.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," *ISMAR*, 2007.

[8] S. A. Scherer and A. Zell, "Efficient Onboard RGBD-SLAM for Fully Autonomous MAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, Tokyo Big Sight, Japan, November 2013.

[9] S. A. Scherer, D. Dube, and A. Zell, "Using Depth in Visual Simultaneous Localisation and Mapping," in *IEEE International Conference on Robotics and Automation*, St. Paul, Minnesota, USA, May 2012.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf," *International Conference on Computer Vision, ICCV*, 2011.

[11] J. Engel, J. Stueckler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[12] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[13] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, July 2016.

[14] B. Simon and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision, IJCV*, 2004.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," 1981.

[16] S. Baker, "Aligning images incrementally backwards. carnegie mellon university. the robotics institute," 2001.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[18] C. Forster, Z. Zichao, G. Michael, W. Manuel, and D. Scaramuzza, "SVO 2.0: Semi-direct visual odometry for monocular and multi-camera systems," in *IEEE Transactions on Robotics*, 2016.

[19] "The pixhawk open-source autopilot project."

[20] C. Kerl, "Master thesis: Odometry from rgb-d cameras for autonomous quadrocopters," Nov. 2012.

[21] D. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer," in *Rensselaer Polytechnic Institute (Technical Report IPL-TR-80-111)*, October 1980.

[22] H. Hirschmueller, "Stereo processing by semi-global matching and mutual information," *IEEE TPAMI*, pp. 328–341, 2008.

[23] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I*, ser. ACCV'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 25–38.

[24] R. Ait Jellal, M. Lange, B. Wassermann, S. Andreas, and A. Zell, "LS-ELAS: line segment based efficient large scale stereo matching," *ICRA*, 2017.

[25] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[27] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.