Autonomous Navigation and Mapping for Inspection of Penstocks and Tunnels with MAVs

Tolga Özaslan[1], Giuseppe Loianno[1], James Keller[1], Camillo J. Taylor[1], and Vijay Kumar[1]
Jennifer M. Wozencraft[2], and Thomas Hood[2]

***Index Terms*—Aerial Systems: Perception and Autonomy; Field Robots; Robotics in Hazardous Fields**

***Abstract*—In this paper, we address the estimation, control, navigation and mapping problems to achieve autonomous inspection of penstocks and tunnels using aerial vehicles with on-board sensing and computation. Penstocks and tunnels have the shape of a generalized cylinder. They are generally dark and featureless. State estimation is challenging because range sensors do not yield adequate information and cameras do not work in the dark. We show that the six Degrees of Freedom (DOF) pose and velocity can be estimated by fusing information from an inertial measurement unit (IMU), a lidar and a set of cameras. This letter discusses in detail the *range-based* estimation part while leaving the details of vision component to our earlier work [1]. The proposed algorithm relies only on a model of the generalized cylinder and is robust to changes in shape of the tunnel. The approach is validated through real experiments showing autonomous and shared control, state estimation and environment mapping in the penstock at Center Hill Dam, TN. To our knowledge, this is the first time autonomous navigation and mapping has been achieved in a penstock without any external infrastructure such GPS or external cameras.**

I. Introduction

MICRO Aerial Vehicles (MAVs) equipped with on-board sensors are becoming ideal platforms for autonomous navigation in complex and confined environments. This is due to the fact that these platforms are superior to ground vehicles in terms of their ability to traverse the 3D space with great ease. Application areas include, but are not bounded to exploration [2], inspection [3], [4], mapping [5], interaction with the environment [6], agricultural inspection, pest control [7], search and rescue, and tactical engagement. Simultaneous Localization and Mapping (SLAM) has been a major problem of interest for the robotics society for more than three decades [8], [9]. Various successful probabilistic methods [9] as well as graph-based solutions [10] have been proposed that brought the literature to the saturation point. As MAVs got more capable and more popular in the past few years, these versatile devices became one of the standard application platform for the SLAM problem. One can argue that the shift from ground vehicles to MAVs is due to several reasons some important of which are MAVs' versatility, low-cost, ease of manufacturing as well as their ability to easily traverse in the 3D space. In this work, we introduce robust state estimators and a complete system design on a MAV platform for autonomous inspection and local mapping of dam penstocks.

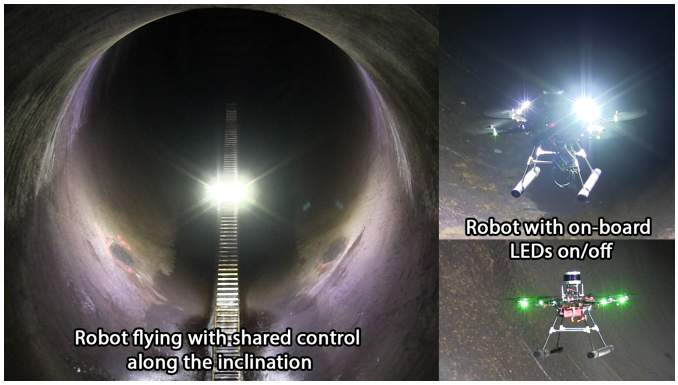


Fig. 1. The DJI F550 platform with modified arms flying autonomously in Center Hill Dam penstock, TN (Sec. II-B).

The key research challenges in the proposed inspection task are rooted in the lack of visual and geometric features, complete lack of external illumination, wet and muddy surfaces and steep inclinations (Fig. 1). The environment is also symmetric which poses several challenges for localization. Due to great oscillating loads coupled with aging infrastructure, penstocks and dams require regular inspection and maintenance. Lack of appropriate maintenance can result in catastrophic consequences such as demolishment of the structure, flood and fire. Conventional inspection techniques include building scaffolds along the whole structure, a worker swinging down from the tunnel gate or climbing along the inclination using ladders. However, the conventional methods are dangerous, potentially inaccurate, labor and cost intensive. Our work addresses the inspection task with novel sensor fusion algorithms for 6 DOF state estimation, control and map generation as well as our custom-design hex-rotor platform (Fig. 1). Our system functions autonomously inside a penstock with the least user intervention from a GUI running on the base station and no need for external illumination.

Relevant to our work are approaches focusing on sensor fusion for 6 DOF state estimation and autonomous navigation of

Manuscript received: February, 15, 2017; Revised April, 11, 2017; Accepted April, 11, 2017.

This paper was recommended for publication by Editor Jonathan Roberts upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the MAST Collaborative Technology Alliance - Contract No. W911NF-08-2-0004, ARL grant W911NF-08-2-0004, ONR grants N00014-07-1-0829, N00014-14-1-0510, ARO grant W911NF-13-1-0350, NSF grants IIS-1426840, IIS-1138847, DARPA grants HR001151626, HR0011516850

[1]The authors are with the GRASP Lab, University of Pennsylvania, 3330 Walnut Street, 19104 Philadelphia, USA. {ozaslan, loiannog, jkeller, cjtaylor, kumar}@seas.upenn.edu.

[2]The authors are with the United States Army Corps of Engineers (USACE).

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters

MAVs. Most of the recent work on SLAM focuses on utilizing cameras and IMU both for indoor and outdoor settings. The recent work [11] shows a small form-factor MAV navigating in unknown environments using only a single camera and a high-rate IMU. The authors fuse the sensory information in a central Unscented Kalman Filter (UKF) for fast pose estimation. While this work focuses on fast and accurate state estimation for aggressive maneuvering, robustness is possible only in environments with sufficient lighting and texture. In [12], the authors integrate noisy measurements from multiple sensors with different frame rates to obtain a globally consistent pose estimate in real time. With a rich sensor pack that includes a laser scanner, GPS receiver, stereo camera rig and an IMU, the authors demonstrate a successful system design that can attain robust control despite varying environmental conditions. However, it relies on the 2.5D assumption and rich texture for range and vision-based estimators respectively; both of which do not hold inside a penstock.

Cameras provide rich data about the surroundings of the robot which is invaluable for robust navigation. However, bundle adjustment or graph-pose optimization may fail to localize map points which in turn degrades the performance of the pose estimation. [13] uses both depth and color information for localization and mapping to overcome this problem. The authors propose a method that can utilize both sparse or dense pointclouds to augment each pixel with depth information for camera motion recovery.

Direct visual SLAM has shown great potential in the past several years. In [14], the authors demonstrate their tightly-coupled monocular Visual-Inertial Odometry (VIO) method on different datasets in real-time for indoor settings. This work, however, heavily depends on rich texture for both camera pose tracking and mapping which is very likely to fail in featureless environments with poor lighting.

There are several other studies which focus on utilizing various sensor combinations for MAV control and navigation. [15] presents a comparative study on various approaches based on RGB-D sensors. [16] uses a 2D laser scanner rectified with a mirror setup and an IMU to navigate a MAV in multi-floor indoor environments. Various studies focus on the application of the SLAM problem to real-life inspection problems such as [17], [18], [19]. In [17] the authors use a wheeled robot equipped with a fish-eye camera to inspect natural gas pipes. Also [18] and [19] use a MAV to inspect the interior of an industrial boiler. These works use a stereo camera rig and an IMU to autonomously traverse the boiler to collect measurements. Finally, directly related to this work are our previous contributions [3], [1]. However, these approaches consider the use of multiple 2D laser scanners and linearized controllers. In addition, a prior knowledge of the map and structure was necessary. Also these studies didn't allow any shared control.

II. Preliminaries

A. Notation

We define the world frame, $\mathcal{W}$, with the unit vectors $\{\hat{x}^{\mathcal{W}}, \hat{y}^{\mathcal{W}}, \hat{z}^{\mathcal{W}}\}$. $\hat{z}^{\mathcal{W}}$ is defined to be anti-parallel to the gravity vector and $\hat{x}^{\mathcal{W}}$ points along the axis of the tunnel at its hover state. If the robot starts in the horizontal section of the tunnel, $\hat{x}^{\mathcal{W}}$ is coincident with the tunnel axis. Vectors in the body frame, $\mathcal{B}$, are defined with respect to the reference triad $\{\hat{x}^{\mathcal{B}}, \hat{y}^{\mathcal{B}}, \hat{z}^{\mathcal{B}}\}$ where $\mathcal{W}$ and $\mathcal{B}$ are aligned when the robot is hovering with zero yaw along the horizontal section.

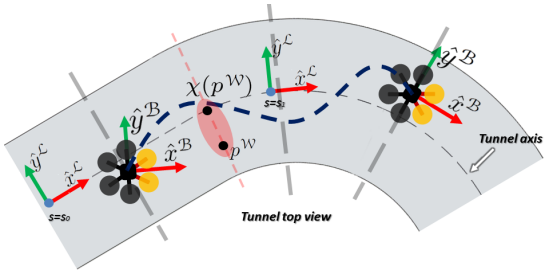


Fig. 2. Sample sketch of the tunnel from top-view with the hex-rotor MAV shown at different poses. Reference frames are coincident with the robots and $\{\hat{x}^{\mathcal{B}}, \hat{y}^{\mathcal{B}}\}$ labels denote body frames at each pose. Local frames are shown at two different positions along the tunnel centerline with axis labels $\{\hat{x}^{\mathcal{L}}, \hat{y}^{\mathcal{L}}\}$. Projection of a point $p^{\mathcal{W}}$ is shown as $\chi(p^{\mathcal{W}})$.

We also define local frames at every point along the tunnel axis that utilizes the uni-axial and symmetric shape of the tunnel. The local frame is defined with the triad $\{\hat{x}^{\mathcal{L}}, \hat{y}^{\mathcal{L}}, \hat{z}^{\mathcal{L}}\}$ where $\hat{x}^{\mathcal{L}}$ and $\hat{y}^{\mathcal{L}}$ are tangent and normal to the tunnel axis, χ, respectively (Fig. 2). Furthermore $\hat{y}^{\mathcal{L}}$ is chosen such that $\hat{y}^{\mathcal{L}} \bullet \hat{z}^{\mathcal{W}} = 0$ and $\hat{z}^{\mathcal{L}} \bullet \hat{z}^{\mathcal{B}} > 0$.

Inter-frame transformations are carried through multiplication with the rotation matrix ${}^{\mathcal{B}}\mathbf{R}_{\mathcal{A}}$. This matrix transforms a point given $\mathcal{A}$ into $\mathcal{B}$.

Finally, the robot state is defined as

$$\mathbf{r} = \begin{bmatrix} \mathbf{x}^\top & \mathbf{v}^\top & \Phi^\top & \mathbf{b}_a^\top & \mathbf{b}_\omega^\top \end{bmatrix}^\top, \tag{1}$$

where $\mathbf{x}$, $\mathbf{v}$, Φ are the position, velocity and orientation expressed in Euler angles respectively, $\mathbf{b}_a$ and $\mathbf{b}_\omega$ are the IMU biases. In this work, we use the ZXY Euler angles convention.

B. DJI-F550 Platform

In this work, we use DJI's F550 frame with extended arms and E600 propulsion systems[1]. It is equipped with a Velodyne Puck LITE[2] lidar, four Chameleon3[3] cameras for imagery collection and estimation, PixHawk[4] on-board controller and IMU, and an Intel i7 NUC PC (Fig. 1). The platform is also equipped with on-board LEDs to provide illumination. Each camera is rectified with four Cree XHP-50[5] high-power LEDs to provide on-board illumination The robot can fly for 10-15 minutes with a 5000mAh 6S - 50C battery with the total payload of 4.5 Kg.

C. Environment Assumptions

The environment is dam penstocks (Fig. 1). These structures lack geometric features and texture for visual processing. Furthermore, they are pitch-dark, uni-axial and axi-symmetric.

[1]https://www.dji.com/e600
[2]http://velodynelidar.com/vlp-16-lite.html
[3]https://www.ptgrey.com/chameleon3-usb3-vision-cameras
[4]https://pixhawk.org/modules/pixhawk
[5]http://www.cree.com/xlamp/xhp50_2

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters

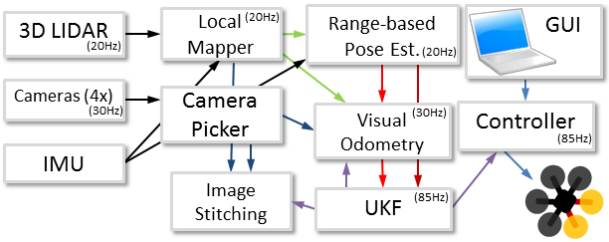


Fig. 3. The overall system diagram. The inputs to the system are sensory information from the three different types of sensors and the user input through the GUI. *Local Mapper* and *Range-based Pose Est.* uses the IMU and 3D pointcloud to generate a local map of the tunnel and localize the robot within that map. *Camera Picker* chooses one or more of the cameras and relay the corresponding frames to the *Visual Odometry* block. Refer to our previous work [1] for the details of *Visual Odometry*. The *UKF* fuses partial pose estimates from each estimator block to obtain 6 DOF pose estimate.

The lack of geometric features poses challenges in estimation of the robot pose in a number of degrees of freedom. In particular, it is not possible to estimate the position of the robot along the tunnel axis with range-based sensors for most of the cases. Section-V shows snapshots of the raw lidar data which depict this scenario.

III. Methodology

This section introduces the overall system structure and its components in details.

A. Overall System Design

The system is composed of several building blocks which include sensory inputs, estimators, controller and user interface. Fig. 3 shows that the inputs to the system are IMU, 3D pointcloud and image data from the heterogeneous set of sensors. The complete state estimation is carried with loosely coupled range and vision-based estimators. Partial state estimates from each estimator are then fused in a central UKF providing 6 DOF state estimate as well as the IMU biases defined in Equ. 1 [12]. We implemented a shared controller such that the user can give high-level way-point commands using a joystick or the GUI application running on the base station.

B. Surface Normal and Uncertainty Estimation

We introduce the main algorithm, based on the Velodyne Puck 3D lidar sensor for 5 DOF state estimation and local mapping. The remaining DOF is the robot position along the tunnel axis. It is either left to the operator control as explained later in Section IV or estimated using the optical flow approach explained in our previous work [1]. This and the subsequent sections explain in detail the blocks *Local Mapper* and *Range-based Pose Est.* depicted in Fig. 3.

Range readings beyond a certain range are not dense enough for accurate surface normal estimation. For this reason, we use only the subset of the pointcloud at most r^* distance from the sensor origin. We define the resulting pointcloud as P^* (Fig. 4(a)). In our tests, r^* ranges from 5 to 12 meters.

The surface normal estimation is at the core of the range-based estimation process. Surface normals are used to estimate local axes of pointcloud segments. The axis of a segment is estimated to be the vector which is perpendicular to all of its surface normals in the least squares sense. Unless otherwise stated, the calculations below are carried in the Velodyne frame, $\mathcal{V}$. We define the normal of the point P_i^*, $\hat{n}_i^*$, as the eigenvector corresponding to the smallest eigenvalue of the matrix $\Delta_{P_i^*}$ [20] defined as

$$\{j_{P_i^*}\} := j \quad | \quad \|P_j^* - P_i^*\| \leq r_{\hat{n}} \tag{2}$$

$$\mu_{P_i^*} = \frac{1}{|\{j_{P_i^*}\}|} \sum_{j \in \{j_{P_i^*}\}} P_j^* \tag{3}$$

$$\Delta_{P_i^*} = \frac{1}{|\{j_{P_i^*}\}|} \sum_{j \in \{j_{P_i^*}\}} (P_j^* - \mu_{P_i^*})(P_j^* - \mu_{P_i^*})^T \tag{4}$$

where $\{j_{P_i^*}\}$ is the set of indices, j, of the points P_j^* which are at most $r_{\hat{n}}$ distant from P_i^*. For radial search we use the Kd-tree implementation of the PCL framework [21].

We also estimate the uncertainty of $\hat{n}_i^*$, by propagating the uncertainty of each point as [22]

$$\Sigma_{\hat{n}_i^*} = \left(\sum_{j \in \{j_{P_i^*}\}} \frac{\partial \hat{n}_i^*}{\partial P_j^*} \left(\Sigma_{P_j^*}\right)^{-1} \frac{\partial \hat{n}_i^*}{\partial P_j^*}^T \right)^{-1} \tag{5}$$

where $\Sigma_{P_j^*}$ is the uncertainty of point P_j^*. Point uncertainty is given as

$$\Sigma_{P_i^*} = {}^{V}\mathbf{R}_{P_i^*} \, \Sigma_{P_i^*}^{\mathcal{P}_i^*} \, {}^{P_i^*}\mathbf{R}_V \tag{6}$$

where $\Sigma_{P_i^*}^{\mathcal{P}_i^*}$ is the uncertainty of the corresponding point in the frame $\mathcal{P}_i^*$ that relates to $\mathcal{V}$ through

$${}^{V}\mathbf{R}_{P_i^*} = \begin{bmatrix} \cos(\beta)\cos(\alpha) & -\cos(\beta)\sin(\alpha) & -\sin(\beta) \\ \sin(\alpha) & \cos(\alpha) & 0 \\ \sin(\beta)\cos(\alpha) & -\sin(\beta)\sin(\alpha) & \cos(\beta) \end{bmatrix} \tag{7}$$

where β and α are the elevation and azimuthal coordinates of P_i^* in $\mathcal{V}$ respectively. The uncertainty of this point as defined in the frame $\mathcal{P}_i^*$ is taken as $\Sigma_{P_i^*}^{\mathcal{P}_i^*} = \|P_i^*\|_2 \cdot diag[\ \sigma_r^2,\ \sigma_\alpha^2,\ \sigma_\beta^2\ ]$ where $\|\cdot\|_2$ is the $L2$ norm, $\sigma_r^2, \sigma_\alpha^2$ and σ_β^2 are the uncertainties in the range, azimuthal and elevation angles per unit range. $\Sigma_{P_i^*}^{\mathcal{P}_i^*}$ models the measurement uncertainty to be linearly increasing with the range reading. Note that, we are following the convention that vector and uncertainty matrix are defined in the frame $\mathcal{V}$ unless another frame is explicitly expressed in the equations as stated earlier.

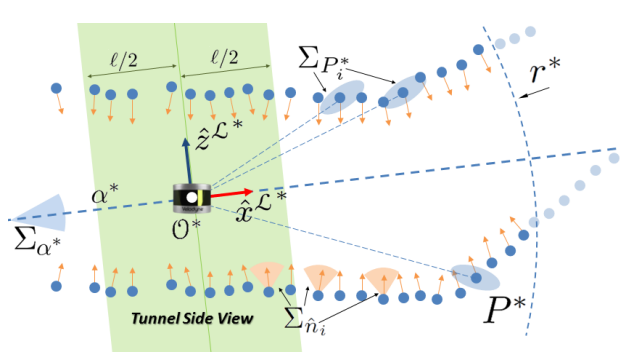
In order to complete the derivation of normal uncertainty, $\Sigma_{\hat{n}_i^*}$, we have to determine the Jacobian $\frac{\partial \hat{n}_i^*}{\partial P_j^*}$. Let λ_i be the smallest eigenvalue of $\Delta_{P_i^*}$ with the corresponding eigenvector $\hat{e}_i = \hat{n}_i^*$. Each column of the derivative of $\hat{n}_i^*$ is given as [23]

$$\frac{\partial \hat{n}_i^*}{\partial P_{j,c}^*} = \frac{1}{|\{j_{P_i^*}\}|} (\lambda_i \mathcal{I} - \Delta_{P_i^*})^{\dagger} \frac{\partial \Delta_{P_i^*}}{\partial P_{j,c}^*} \hat{n}_i^* \tag{8}$$

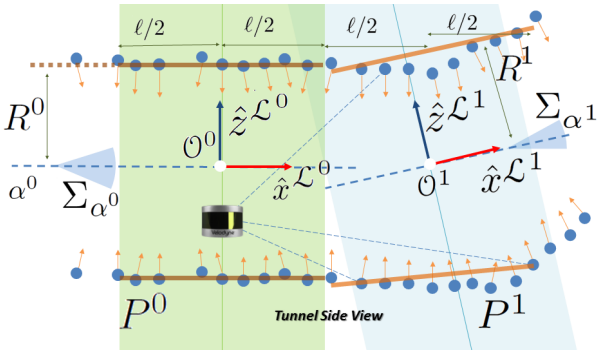
where $\mathcal{I}$ is the identity matrix, $\bullet^{\dagger}$ is the Moore-Penrose inverse, $P_{i,c}^*$ is the c^{th} component of P_i^* with $c \in \{x, y, z\}$ and

$$\frac{\partial \Delta_{P_i^*}}{\partial P_{j,x}^*} = \frac{[\ P_j^* - \mu_{P_i^*},\ \mathbf{0},\ \mathbf{0}\ ]^T + [\ P_j^* - \mu_{P_i^*},\ \mathbf{0},\ \mathbf{0}\ ]}{|\{j_{P_i^*}\}|} \tag{9}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters



(a) This figure illustrates pointcloud data (blue circles) from inside a penstock plotted in side-view and the initialization step of the local frame and segment estimation process.



(b) This figure shows the results of two iterations of local frame and segment estimation.

Fig. 4. Algorithm 4 illustrated. These figures explain the components of the range-based pose estimation and the local mapping algorithm.

$$\frac{\partial \Delta_{P_i^*}}{\partial P^*_{j,y}} = \frac{[\,\mathbf{0},\ P_j^* - \mu_{P_i^*},\ \mathbf{0}\,]^T + [\,\mathbf{0},\ P_j^* - \mu_{P_i^*},\ \mathbf{0}\,]}{|\{j_{P_i^*}\}|} \tag{10}$$

$$\frac{\partial \Delta_{P_i^*}}{\partial P^*_{j,z}} = \frac{[\,\mathbf{0},\ \mathbf{0},\ P_j^* - \mu_{P_i^*}\,]^T + [\,\mathbf{0},\ \mathbf{0},\ P_j^* - \mu_{P_i^*}\,]}{|\{j_{P_i^*}\}|} \tag{11}$$

Finally, the Jacobian matrices

$$\frac{\partial \hat{n}_i^*}{\partial P_j^*} = \frac{1}{|\{j_{P_i^*}\}|}(\lambda_i \mathcal{I} - \Delta_{P_i^*})^\dagger \left[\ \frac{\partial \Delta_{P_i^*}}{\partial P^*_{j,x}}\hat{n}_i^*,\quad \frac{\partial \Delta_{P_i^*}}{\partial P^*_{j,y}}\hat{n}_i^*,\quad \frac{\partial \Delta_{P_i^*}}{\partial P^*_{j,z}}\hat{n}_i^*\ \right] \tag{12}$$

can be plugged into Equ. 5 to propagate point uncertainties into normal uncertainties.

Fig. 4(a) illustrates the definitions introduced in this section. Points farther than r^* meters from the sensor are drawn in light-blue and never used in the subsequent calculations. The inlier data (blue points) is denoted by P^*. Some of the points are overlaid their corresponding uncertainties formula of which is given in Equ. 6. Normal vectors of each point are plotted in orange. Uncertainties of sample normal vectors are also plotted in light-orange formula of which is given in Equ. 5. This figure also illustrates α^* and $\mathcal{L}^*$ further details for which are provided in the following sections. Briefly, α^* is the initial tunnel axis estimate perpendicular to all the normal vectors in the least-squares manner. Its uncertainty is denoted as Σ_{α^*}. An initial local coordinate frame is defined using α^* given by the unit vectors $\mathcal{L}^* := \{\hat{x}^{\mathcal{L}^*}, \hat{y}^{\mathcal{L}^*}, \hat{z}^{\mathcal{L}^*}\}$. The origin of $\mathcal{L}^*$ is coincident with origin of $\mathcal{V}$ at $\mathcal{O}^*$.

C. Pointcloud Segmentation and Surface Fitting

The Field of View (FOV) of the Velodyne Puck along the elevation dimension is very narrow (± 15 degrees), which imposes a significant constraint on sensing and modeling its surroundings. However, using the symmetry of the tunnel, we can extrapolate the pointcloud data in order to get a parametric representation of the tunnel surface. In a practical sense, fitting cylindrical segments to the pointcloud results in a 360 degrees FOV around the tunnel axis. As opposed to direct pointcloud matching algorithms such as [24], [25], this method does not suffer from data association problems due to non-overlapping FOVs between different laser scans. Furthermore, this can also be used to get depth estimates of image features with zero latency.

1) Overview: The segmentation and mapping process is an iterative algorithm with can be summarized as *initialize-refine-recur* until all the points in P^* are consumed. P^* is the pointcloud obtained after radius filtering applied on the raw data, P, as explained in Section III-B. At this point, we also have to provide the definition of a segment S. Segment, S^s, with the index $s \in \{*, 0, \pm 1, \pm 2, ...\}$ consists of a local frame, $\mathcal{L}^s$, origin, $\mathcal{O}^s$ and a pointcloud P^s (i.e. $S^s := \{\mathcal{L}^s, \mathcal{O}^s, P^s\}$).

The first step is to use the pointcloud, P^*, to initialize a rough, but reliable, initial local coordinate frame which consists of $\mathcal{L}^*$ and $\mathcal{O}^* = \mathbf{0}$. In order to define $\mathcal{L}^*$, we first have to estimate the local tunnel axis α^*, which is found as the eigenvector of $M = (WN)^T(WN)$ corresponding to its smallest eigenvalue as explained in Algorithm 1 where N is $|P^*| - by - 3$ matrix with $N_i = \hat{n}_i^T$ and W is the diagonal weighing matrix with $W_{(i,i)} = e^{-(\kappa_i^*/\tau_\kappa)^2}$. Here, κ_i^* is the curvature at P^* which is defined as the ratio of the smallest eigenvalue of $\Delta_{P_i^*}$ to their summation [20] and τ_κ is a constant which we choose to be in the range $[0.1, 0.3]$. W assigns smaller weights to points at high curvature regions. We can then use the procedure explained in Sec. II-A to construct the frame $\mathcal{L}^* := \{\hat{x}^{\mathcal{L}^*}, \hat{y}^{\mathcal{L}^*}, \hat{z}^{\mathcal{L}^*}\}$. This process is also illustrated in Fig. 4. Note that the same procedure can easily be adapted for local axis estimation by assigning s to either $\{0, \pm 1, \pm 2, ...\}$. Then, S^* is used to initialize the segment $S^0 := \{\mathcal{L}^0, \mathcal{O}^0, P^0\}$ (Fig. 4(b)). The initialization is followed by parameter refinement and outlier rejection (Sec. III-C3). These steps are recursed to obtain S^1 and $S^{(-1)}$ where the sign denotes the forward and backward direction with respect to the tunnel axis. The process continues until all the points in P^* are either segmented into tunnel sections or marked as outliers.

2) Segment Initialization: Segment initialization is handled differently for the cases $s = *$, $s = 0$ and $|s| > 0$. The first case, $s = *$, is explained in the previous paragraph. S^0 is initialized with $\mathcal{L}^0 = \mathcal{L}^*$, $\mathcal{O}^0 = \mathcal{O}^*$. P^0 is defined to be the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters

Algorithm 1: $\mathcal{L}$ = EstimateLocalFrame(P, $\hat{n}$, Σ_P)

while $i < |P|$ **do**
  $N_i = \hat{n}_i^T$;
  $evals = eigvals(\Sigma_{P_i})$;
  $\kappa_i = min(evals) / \sum evals$; *// Curvature at P_i [20].*
  $W(i,i) = e^{-(\kappa_i/\tau_\kappa)^2}$;
end
// get the eigenvector for the smallest eigenvalue
$\alpha = min\left(eigvecs\left((WN)^T(WN)\right)\right)$;
return $\mathcal{L} = defineFrame(\alpha)$ *// see Sec. II-A*;

Algorithm 2: P_{out} = SegmentPointcloud(P, $\mathcal{L}$, $\mathcal{O}$, ℓ)

while $i < |P|$ **do**
  $P_i^{\mathcal{L}} = {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}(P_i - \mathcal{O})$;
  if $\left|P_{i,x}^{\mathcal{L}}\right| \leq \ell/2$ **then**
    $P_{out} = P_{out} \bigcup P_i^{\mathcal{L}}$; *// expand pointcloud*
end
return P_{out} ;

set of all points that satisfy $|P_{i,x}^{*\mathcal{L}^0}| \leq \ell/2$ where

$$P_i^{*,\mathcal{L}^0} = {}^{\mathcal{L}^0}\mathbf{R}_{\mathcal{V}}(P_i^* - \mathcal{O}^0). \tag{13}$$

For other cases, $|s| \geq 1$, the coordinate frame is initialized as $\mathcal{L}^s = \mathcal{L}^{s^-}$ and $\mathcal{O}^s = \mathcal{O}^{s^-} + sign(s) \cdot \ell \cdot \hat{x}^{\mathcal{L}^{s^-}}$ (Fig. 4(b)) where $s^- = s - sign(s)$ for $s \neq 0$ and $s^- = *$ for $s = 0$. P^s can be obtained the same way as P^0 through proper coordinate frame substitution in Equ. 13. Pointcloud segmentation is also explained in Algorithm 2. In our tests, we choose ℓ to be in the range of $[0.10 - 2]$ meters.

3) Segment Refinement: The quality of segment initialization is dependent on the estimation quality of the previous segment, S^{s^-}, as well as the noise level of sensory data which is not guaranteed for certain cases. For this reason, initialization is followed by refining the point set P^s as well as the coordinate frame definitions $\mathcal{L}^s - \mathcal{O}^s$ through outlier rejection and refitting the model iteratively until convergence. The two assumptions we use for outlier detection are the agreement of local surface normals, $\hat{n}_i^s$, with the local tunnel axis, α^s, and the cylindrical surface model. The former assumption eliminates points with incompatible surface normals to obtain more accurate local frames, $\mathcal{L}^s$ and the latter removes regions of the pointcloud that do not comply with the cylindrical surface assumption due to objects such as scaffolding and operators.

Outlier rejection based on the surface normal assumes that each normal, $\hat{n}_i^s$, is perpendicular to the local tunnel axis, α^s. This condition is written as $|\hat{n}_i^s \bullet \alpha^s| \leq \tau_{\hat{n}}$. We choose $\tau_{\hat{n}}$ to in the range of $[0.1, 0.25]$. Both the surface normal compliance assumption and the low weight assignment to high curvature regions (Algorithm 1) for tunnel axis estimation, give robust local frame estimates.

We use the points in P^s to fit a cylindrical model, C^s, assuming that the tunnel has parametric cross-section which is circle in this case. C^s is defined with its origin, $\mathcal{O}_c^s$, axis length and radius, R^s. Axis length is always equal to the segment length, ℓ. In order to simplify the model fitting problem, we use the fact that, for a reasonably well estimated local frame, $\mathcal{L}^s$,

Algorithm 3:
S_{out} = RefineSegment(S, $\hat{n}$, Σ_P)

$[P, \mathcal{L}, \mathcal{O}] \Leftarrow S$;
while $outer\_iter \leq outer\_iter_{max}$ **do**
  while $inner\_iter \leq inner\_iter_{max}$ **do**
    $\mathcal{L}$ = EstimateLocalFrame(P, $\hat{n}$, Σ_P) ;
    $\alpha = \mathcal{L}_{\hat{x}}$;
    while $i < |P|$ **do**
      if $|\hat{n}_i \bullet \alpha| \geq \tau_{\hat{n}}$ **then**
        $P = P \backslash P_i$; *// remove P_i from P*
    end
  end
  while $inner\_iter \leq inner\_iter_{max}$ **do**
    while $i < |P|$ **do**
      $p = {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}(P_i - \mathcal{O})$;
      $A_i = [\ p_{i,y},\ \ p_{i,z},\ \ 1\ ]$;
      $b_i = -\left({p_{i,y}}^2 + {p_{i,z}}^2\right)$;
    end
    $f = A^\dagger b$;
    $\mathcal{O}_{c,x} = 0\ ,\ \ \mathcal{O}_{c,y} = -f_1/2\ ,\ \ \mathcal{O}_{c,z} = -f_2/2$;
    $R = \sqrt{(f_1^2 + f_2^2)/16 - f_3}$;
    while $i < |P|$ **do**
      $p = {}^{\mathcal{L}}\mathbf{R}_{\mathcal{V}}(P_i - \mathcal{O})$;
      if $|1 - \|p_{(y,z)}\|_2/R| \geq \tau_R$ **then**
        $P = P \backslash P_i$; *// remove P_i from P*
    end
  end
  $\mathcal{O} = \mathcal{O} + \left({}^{\mathcal{V}}\mathbf{R}_{\mathcal{L}}\right)\mathcal{O}_c$;
end
return $S = [P, \mathcal{L}, \mathcal{O}]$;

the projection of the points P_i^s onto the local $y-z$ plane will form a circle. Then the parameter estimation can be formulated as a linear regression written as

$$A_i = [\ p_{i,y},\ \ p_{i,z},\ \ 1\ ] \tag{14}$$

$$b_i = -\left({p_{i,y}}^2 + {p_{i,z}}^2\right) \tag{15}$$

$$f = A^\dagger b \tag{16}$$

where $p_i := P_i^{s,\mathcal{L}^s}$, f is a 3-vector. The center and the radius of C^s are found as

$$\mathcal{O}_{c,(y,z)}^{s,\mathcal{L}^s} = -\frac{1}{2}[\ f_1,\ f_2\ ]^T \tag{17}$$

$$R^s = \sqrt{\left(f_1^2 + f_2^2\right)/16 - f_3} \tag{18}$$

where $\mathcal{O}_{c,x}^{s,\mathcal{L}^s} = 0$ by definition. Note that the model centers are defined in their corresponding local frames hence the superscript $\mathcal{L}^s$. In order to simplify the notation, this point on, we will drop the frame label from *model centers*. At each refinement step, the model center, $\mathcal{O}_c^s$, is used to update the local frame origin, $\mathcal{O}^s$. The update is written as

$$\mathcal{O}^s \Leftarrow \mathcal{O}^s + \left({}^{\mathcal{V}}\mathbf{R}_{\mathcal{L}^s}\right)\mathcal{O}_c^s. \tag{19}$$

This equation indicates that the model center and the local frame have to be aligned due to the cylindrical cross-section assumption. This always holds in straight sections of the tunnel whereas ℓ should be chosen to be smaller as the curvature of the tunnel increases, such as around bends.

The second outlier rejection uses the cylindrical surface assumption. Points that are far from the surface are marked as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters

outliers. This criterion is given as

$$\left| 1 - \frac{\|P^{s,\mathcal{L}s}_{i,(y,z)}\|_2}{R^s} \right| \leq \tau_R \tag{20}$$

where R^s is the radius of the C^s, $\|P^{s,\mathcal{L}^s}_{i,(y,z)}\|_2$ is the $L2$ norm of the point P^s_i as written in the $\mathcal{L}^s$ frame projected on to the local $y-z$ plane. The latter can also be interpreted as the distance of the corresponding point from the tunnel axis. τ_R is a constant that is in the range $[0.1, 0.2]$. The refinement process is also explained in Algorithm 3.

Algorithm 4: $\{S\}$ = BuildLocalMap(P)

```
P* = RadiusFilter(P) ; // Sec. III-B & Fig. 4(a)
[n̂*, Σ_P*] = EstimateSurfaceNormals(P*) ; // Sec. III-B
L* = EstimateLocalFrame(P*, n̂*, Σ_P*) ; // Algo. 1
O* = 0 ;
s = 0 ;
while P* ≠ ∅ ∧ |s| ≤ s_max do
    // Sec. III-C2
    [L^s, O^s] = InitializeSegment(P*, L^s⁻, O^s⁻) ;
    S^s = RefineSegment(P, L^s, O^s, n̂^s, Σ_Ps); // Algo. 3
    s++ ;
end
return {S} ;
```

Fig. 4(b) illustrates the segment initialization and refinement steps. In this figure, the green and light-blue shaded rectangles designate the regions of each local frame, $\mathcal{L}^0$ and $\mathcal{L}^1$ respectively. Points belonging to each region are denoted as P^0 and P^1. The algorithm starts with the initialization step as illustrated in Fig. 4(a). The points inside the shaded region in Fig. 4(a) are then used to refine the local frame estimate $\mathcal{L}^0$. After the refinement, we obtain $\{\hat{x}^{\mathcal{L}^0}, \hat{y}^{\mathcal{L}^0}, \hat{z}^{\mathcal{L}^0}\}$ and $\mathcal{O}^0$ defined in $\mathcal{V}$. Then, a cylindrical surface is fit to P^0 with all the corresponding parameter uncertainties as explained in the following sections. Once the 0^{th} segment is processed, an initial guess for $\mathcal{O}^1$ and α^1 is made by extrapolating α^0 by $+l$ meters. This guess is then refined to obtain $\{\hat{x}^{\mathcal{L}^1}, \hat{y}^{\mathcal{L}^1}, \hat{z}^{\mathcal{L}^1}\}$ and $\mathcal{O}^1$ using the point set P^1 where P^1 consists of all the unused points that are at most $l/2$ meters away from the $\hat{y}^{\mathcal{L}^1}$-$\hat{z}^{\mathcal{L}^1}$ plane.

D. Uncertainty Estimation of Local Frames

We estimate the uncertainty of α^s by propagating normal uncertainties through the equation

$$\Sigma_{\alpha^s} = \left(\sum_{i \in \{P^s\}} \frac{\partial \alpha^s}{\partial \hat{n}^s_i} \left(\Sigma_{\hat{n}^s_i}\right)^{-1} \frac{\partial \alpha^{s\,T}}{\partial \hat{n}^s_i} \right)^{-1} \tag{21}$$

where the summation is over all surface normals contributing to the estimation of α^s. The partial of α^s with respect to the normal $\hat{n}^s_i$ is calculated in a similar way as the normal vector partials. Let λ_i be the smallest eigenvalue of M with the corresponding eigenvector $\hat{e}_i = \alpha^s$. Each column of the derivative of α^s is given as [23]

$$\frac{\partial \alpha^*}{\partial n^*_{i,c}} = (\lambda_i \mathcal{I} - M)^\dagger \frac{\partial M}{\partial \hat{n}^s_{i,c}} \alpha^s \tag{22}$$

where $\mathcal{I}$ is the identity matrix, $\hat{n}^s_{i,c}$ is the c^{th} component of $\hat{n}^s_i$ with $c \in \{x, y, z\}$ and

$$\frac{\partial M}{\partial \hat{n}^s_{i,x}} = W^2_{(i,i)} \left([\, \hat{n}^s_i,\ \mathbf{0},\ \mathbf{0}\,]^T + [\, \hat{n}^s_i,\ \mathbf{0},\ \mathbf{0}\,] \right) \tag{23}$$

$$\frac{\partial M}{\partial \hat{n}^s_{i,y}} = W^2_{(i,i)} \left([\, \mathbf{0},\ \hat{n}^s_i,\ \mathbf{0}\,]^T + [\, \mathbf{0},\ \hat{n}^s_i,\ \mathbf{0}\,] \right) \tag{24}$$

$$\frac{\partial M}{\partial \hat{n}^s_{i,z}} = W^2_{(i,i)} \left([\, \mathbf{0},\ \mathbf{0},\ \hat{n}^s_i\,]^T + [\, \mathbf{0},\ \mathbf{0},\ \hat{n}^s_i\,] \right). \tag{25}$$

The Jacobian in the above equation may be written

$$\frac{\partial \alpha^*}{\partial \hat{n}^s_i} = (\lambda_i \mathcal{I} - M)^\dagger \left[\begin{array}{ccc} \frac{\partial M}{\partial \hat{n}^s_{i,x}} \alpha^s, & \frac{\partial M}{\partial \hat{n}^s_{i,y}} \alpha^s, & \frac{\partial M}{\partial \hat{n}^s_{i,z}} \alpha^s \end{array} \right]. \tag{26}$$

This can be substituted in Equ. 21 to get Σ_{α^s}.

E. Robot State and Its Uncertainty

Robot orientation can be estimated only with the presence of an IMU since roll and pitch angles, $\{\phi, \theta\}$, are not observable to the range scanner. We integrate the rotational velocity from the IMU in our UKF and also use gravity vector to eliminate possible error accumulation in the orientation for roll and pitch estimation. We use the formulation given in [12] for UKF. This method works very accurately in non-aggressive flights, which is the case in our application. Assuming that IMU and Velodyne frames are aligned, we can use the IMU to infer the transformation between the Velodyne frame, $\mathcal{V}$, and the world frame, $\mathcal{W}$, as ${}^W\mathbf{R}_V = Rot(\theta, \phi)$. Then robot's yaw angle, ψ, can be obtained from $\alpha^{0,\mathcal{W}} = \left({}^W\mathbf{R}_V\right) \alpha^0$ as $\psi = -atan\left(\alpha^{0,\mathcal{W}}_y / \alpha^{0,\mathcal{W}}_x\right)$ defined in the *local* tunnel axis. This equation holds when the robot is heading forward, $\hat{x}^{\mathcal{B}} \bullet \hat{x}^{\mathcal{L}} > 0$, and the tunnel is not vertical, which is consistent with the geometry of most penstocks. The uncertainty in ψ can be estimated by propagating the uncertainty in α^0 as

$$\Sigma^{\mathcal{W}}_\psi = \left(\frac{\partial \psi}{\partial \alpha^{0,\mathcal{W}}} \left(\Sigma^{\mathcal{W}}_{\alpha^0_{x,y}}\right)^{-1} \frac{\partial \psi}{\partial \alpha^{0,\mathcal{W}}}^T \right)^{-1} \tag{27}$$

where

$$\frac{\partial \psi}{\partial \alpha^{0,\mathcal{W}}_{x,y}} = \frac{1}{\|\alpha^{0,\mathcal{W}}_{x,y}\|_2} \left[\begin{array}{cc} \alpha^{0,\mathcal{W}}_y, & -\alpha^{0,\mathcal{W}}_x \end{array} \right] \tag{28}$$

which completes the derivation of $\Sigma^{\mathcal{W}}_\psi$.

We prefer defining the position of the robot with respect to local the origin, $\mathcal{O}^0$. This offers an intuitional way of defining way-points for the controller making the estimator immune to drifts in vertical and lateral positions, $\mathbf{r}_{y,z}$. The lateral and vertical coordinates are found as $\mathbf{r}^W_{y,z} = -\left({}^W\mathbf{R}_V\ \mathcal{O}^0\right)_{y,z}$.

The uncertainty in $\mathbf{r}_{y,z}$ is a linear transformation of the uncertainty in the $y-z$ coordinates of $\mathcal{O}^0$, $\Sigma_{\mathcal{O}^0_{y,z}}$, which writes *(all in $\mathcal{L}^0$)*

$$\Sigma_{\mathcal{O}^0_{y,z}} = \left(\sum_{i \in \{P^0\}} \left(\frac{\partial \mathcal{O}^0_{y,z}}{\partial P^0_i} \right) \left(\Sigma_{P^0_i}\right)^{-1} \left(\frac{\partial \mathcal{O}^0_{y,z}}{\partial P^0_i} \right)^T \right)^{-1} \tag{29}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2017.2699790, IEEE Robotics and Automation Letters

where

$$\frac{\partial \mathcal{O}^0_{y,z}}{\partial P^0_i} = \frac{\partial \mathcal{O}^0_{y,z}}{\partial f} \cdot \frac{\partial f}{\partial P^0_i} \tag{30}$$

$$\frac{\partial f}{\partial P^0_i} = A^\dagger \left(\frac{\partial b}{\partial P^0_i} - \frac{\partial A}{\partial P^0_i} f \right) \tag{31}$$

and

$$\frac{\partial A}{\partial P^0_{i,x}} = \mathbf{0}, \quad \frac{\partial A}{\partial P^0_{i,y}} = \delta_{i,0}, \quad \frac{\partial A}{\partial P^0_{i,z}} = \delta_{i,1},$$

$$\frac{\partial b}{\partial P^0_{i,x}} = \mathbf{0}, \quad \frac{\partial b}{\partial P^0_{i,y}} = -2\delta_{i,0} P^0_{i,y}, \quad \frac{\partial b}{\partial P^0_{i,z}} = -2\delta_{i,0} P^0_{i,z},$$

$$\frac{\partial \mathcal{O}^0_y}{\partial f} = -1/2\,[\,1,\ 0,\ 0\,], \quad \frac{\partial \mathcal{O}^0_y}{\partial f} = -1/2\,[\,0,\ 1,\ 0\,]$$

and $\delta_{i,j}$ is a zero matrix except that it is 1 at (i,j). We can obtain the uncertainty of $\mathcal{O}^0_{y,z}$, $\Sigma^{\mathcal{L}^0}_{\mathcal{O}^0_{y,z}}$, by substituting the above equations. This is transformed into $\mathcal{W}$ as $\Sigma^{\mathcal{W}}_{\mathcal{O}^0_{y,z}} = {}^{\mathcal{W}}\mathbf{R}'_{\mathcal{L}^0}\ \Sigma^{\mathcal{L}^0}_{\mathcal{O}^0_{y,z}}\ {}^{\mathcal{L}^0}\mathbf{R}'_{\mathcal{W}}$ where ${}^{\mathcal{L}^0}\mathbf{R}'_{\mathcal{W}}$ is the 2-by-2 lower-right block of ${}^{\mathcal{L}^0}\mathbf{R}_{\mathcal{W}}$. One should note that we silently assumed that $\mathcal{O}^0_c \equiv \mathcal{O}^0$ which is valid for successful cylinder fitting.

IV. Estimation and Control

In this section, we briefly present the estimation and controller pipelines of our platform. To enable on-board control, a UKF is used to estimate the full *state* $\mathbf{r}$ of the vehicle, defined in Sec. II-A, at the IMU rate. We embrace the formulation given by [12] for the UKF and IMU integration. The prediction step uses the input translational acceleration and angular velocity measurements given by the IMU, whereas the measurement update is the pose provided by the localization information discussed in the previous section. The control of the hex-rotor platform is composed of an attitude and a position controller. In most previous works, a back-stepping approach is used for control because the attitude dynamics can be assumed to be faster than the position dynamics, and linearized controllers are used for both loops [26], [27]. In this work, for robustness, we use a nonlinear controller based on [28], [29]. The control inputs for the system are the thrust τ and the moments around the rigid body axes $\mathbf{M}$. They are chosen as

$$\begin{aligned} \mathbf{M} = &- k_R \mathbf{e}_R - k_\Omega \mathbf{e}_\Omega + \boldsymbol{\Omega} \times J\boldsymbol{\Omega} - \\ &\mathbf{J}\left(\hat{\boldsymbol{\Omega}}\mathbf{R}^\top \mathbf{R}_C \boldsymbol{\Omega}_C - \mathbf{R}^\top \mathbf{R}_C \dot{\boldsymbol{\Omega}}_C\right), \\ \tau = &(-k_x \mathbf{e}_x - k_v \mathbf{e}_v + mg\mathbf{e}_3 + m\ddot{\mathbf{x}}_d) \cdot \mathbf{R}\mathbf{e}_3, \end{aligned} \tag{32}$$

with $\ddot{\mathbf{x}}_d$ the desired acceleration, k_x, k_v, k_R, k_Ω positive definite terms. The quantities $\mathbf{e}_R, \mathbf{e}_\Omega, \mathbf{e}_x, \mathbf{e}_v$ are the orientation, angular rate, and translation position and velocity errors respectively, defined in [28], [29], [30] and obtained using the information from the UKF estimator. The subscript C denotes a commanded value and $\mathbf{J}$ is the inertia matrix. If the initial attitude error is between 90° and 180°, the zero equilibrium of the tracking errors is almost globally exponentially attractive [28]. Our shared control approach allows the user to take over the control along the tunnel axis if required. In this mode, the user directly controls the robot pitch, while the platform motion is constrained along the main axis regardless of the inclination.

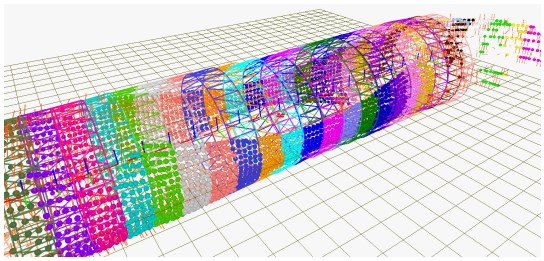


Fig. 5. Screen-shot from the RViz visualization tool showing the robot flying with shared control along the horizontal section of the tunnel. The colored point cloud and their corresponding meshes demonstrate the output of the segmentation and the cylinder fitting algorithms. The robot is shown with a red CAD model at the very center of the meshes.

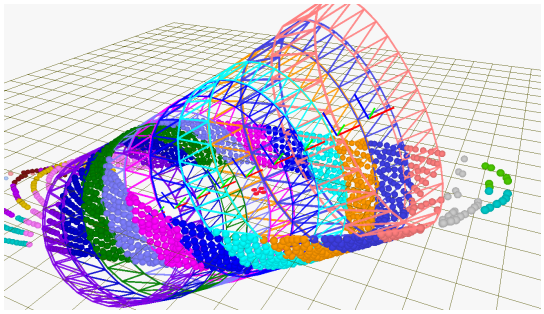


Fig. 6. Screen-shot from the RViz visualization tool showing the robot flying with shared control along the inclined section of the tunnel after ~ 20 seconds after the robot state in Fig. 6. The algorithms does not require any modifications to handle the transition between the horizontal and the inclined section.

V. Experimental Results

In this section, we report on the experiments performed at Center Hill Dam (CHD), TN. The goal of our experiments is to show the robot can autonomously navigate to the end of the tunnel, concurrently reconstructing the local environment. The vehicle starts at an arbitrary point in the horizontal section and traverses the tunnel until the end of the inclined section. The transition between the horizontal and the inclined sections are handled successfully. At CHD, the tunnel is approximately 80 m. long. The local maps can be merged to form the complete map of the tunnel by integrating the optical flow based velocity estimates which we leave as future work.

The flight starts with the operator commanding the robot to take off. In order to reduce vortex formation when in close proximity to walls and to maximize the image brightness, we command the robot to align with the center-line ($\mathbf{r}_{y,z,\psi} = \mathbf{0}$). Then the operator can command the robot to go forward or backward along the tunnel through the GUI. The RViz visualization shows the sensory data such as video stream and pointcloud as in Fig. 5, 6 to the operator in real-time. This interface allows the operator to direct the robot even when the robot is outside the field of view of the operator.

In the experiments, we command the robot to fly along the center-line ($\mathbf{r}_{y,z,\psi} = \mathbf{0}$). The robot closely follows these commands as shown in Fig. 7 except for oscillations and a constant offset in $\mathbf{r}_z$ due to imprecise parameter tuning. This proves the accuracy of the proposed estimator.

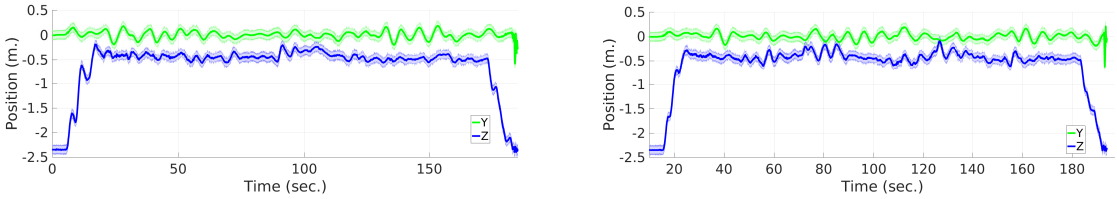


Fig. 7. Vertical and lateral position, $\mathbf{r}_{y,z}$, of the robot flying during two different tests along the entire penstock. Shades around the plots are the corresponding standard deviation scaled by 100.

The diameter of the penstock is 5.5 meters and our algorithm could estimate it within a 5% error. We refer to our previous work, [1], for details of the visual odometry and its results due to brevity and limited space.

VI. Conclusion

In this work, we present a new approach for state estimation, mapping and shared control with MAVs in tunnels. The key components are the range-based localization in a parametric cylindrical environment, local mapping and shared control. The on-board controller uses the feedback from the estimator to drive the robot to the end of the tunnel while allowing shared control with an operator. Our results show the capability of autonomous navigation in penstocks and tunnels, and the capability to concurrently map the local environment.

Future work will consider image analysis algorithms to provide a panoramic visualization of the environment and the capability to automatically identify regions that require maintenance. It is also in our interest to consider navigation in longer tunnels with branch-off and merging sections.

References

[1] T. Özaslan, K. Mohta, J. Keller, Y. Mulgaonkar, C. J. Taylor, V. Kumar, J. M. Wozencraft, and T. Hood, "Towards fully autonomous visual inspection of dark featureless dam penstocks using MAVs," in *2016 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, oct 2016, pp. 4998–5005.

[2] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sept 2012.

[3] T. Özaslan, S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, *Inspection of Penstocks and Featureless Tunnel-like Environments Using Micro UAVs*. Cham: Springer International Publishing, 2015, pp. 123–136.

[4] J. Cacace, A. Finzi, V. Lippiello, G. Loianno, and D. Sanzone, *Aerial Service Vehicles for Industrial Inspection: Task Decomposition and Plan Execution*. Springer Berlin Heidelberg, 2013, pp. 302–311.

[5] G. Loianno, J. Thomas, and V. Kumar, "Cooperative localization and mapping of mavs using rgb-d sensors," in *IEEE International Conference on Robotics and Automation*, May 2015, pp. 4021–4028.

[6] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 57–64, Jan 2016.

[7] J. Das, G. Cross, C. Qu, and A. Makineni, "Devices, Systems, and Methods for Automated Monitoring enabling Precision Agriculture," *... (CASE), vol., no*, 2015.

[8] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.

[9] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, jun 2001.

[10] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proc. 2006 IEEE Int. Conf. Robot. Autom. 2006. ICRA 2006.* IEEE, 2006, pp. 2262–2269.

[11] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, apr 2017.

[12] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV," in *2014 IEEE Int. Conf. Robot. Autom.*, may 2014, pp. 4974–4981.

[13] J. Zhang, M. Kaess, and S. Singh, "A real-time method for depth enhanced visual odometry," *Auton. Robots*, vol. 41, no. 1, 2017.

[14] A. Concha, G. Loianno, V. Kumar, and J. Civera, "Visual-inertial direct SLAM," in *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2016-June, 2016, pp. 1331–1338.

[15] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Springer Tracts Adv. Robot.*, vol. 100, 2017, pp. 235–252.

[16] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *Proc. - IEEE Int. Conf. Robot. Autom.* IEEE, may 2011, pp. 20–25.

[17] P. Hansen, H. Alismail, P. Rander, and B. Browning, "Visual mapping for natural gas pipe inspection," *Int. J. Rob. Res.*, vol. 34(4-5), no. 4-5, pp. 532–558, nov 2015.

[18] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart, "A UAV system for inspection of industrial facilities," in *IEEE Aerosp. Conf. Proc.*, 2013.

[19] M. Burri, J. Nikolic, C. Hürzeler, G. Caprari, C. Hurzeler, G. Caprari, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," *Appl. Robot. Power Ind. (CARPI), 2012 2nd Int. Conf.*, pp. 70–75, 2012.

[20] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[21] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[22] A. Censi, "An accurate closed-form estimate of icp's covariance," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 3167–3172.

[23] J. R. Magnus, "On differentiating eigenvalues and eigenvectors," *Econometric Theory*, vol. 1, no. 02, pp. 179–191, 1985.

[24] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Auton. Robots*, vol. 41, no. 2, pp. 401–416, feb 2017.

[25] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[26] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The Grasp Multiple Micro–UAV Test Bed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[27] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[28] T. Lee, M. Leok, and N. H. McClamroch, "Nonlinear Robust Tracking Control of a Quadrotor UAV on SE(3)," *Asian Journal of Control*, vol. 15, no. 2, pp. 391–408, 2013.

[29] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *IEEE International Conference on Robotics and Automation*, Shangai, China, 2011, pp. 2520–2525.

[30] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "Smartphones power flying robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, Sept 2015, pp. 1256–1263.