NanoMap: Fast, Uncertainty-Aware Proximity Queries with Lazy Search over Local 3D Data

Peter R. Florence[1], John Carter[1], Jake Ware[1], Russ Tedrake[1]

***Abstract*— We would like robots to be able to safely navigate at high speed, efficiently use local 3D information, and robustly plan motions that consider pose uncertainty of measurements in a local map structure. This is hard to do with previously existing mapping approaches, like occupancy grids, that are focused on incrementally fusing 3D data into a common world frame. In particular, both their fragile sensitivity to state estimation errors and computational cost can be limiting. We develop an alternative framework, NanoMap, which alleviates the need for global map fusion and enables a motion planner to efficiently query pose-uncertainty-aware local 3D geometric information. The key idea of NanoMap is to store a history of noisy relative pose transforms and search over a corresponding set of depth sensor measurements for the minimum-uncertainty view of a queried point in space. This approach affords a variety of capabilities not offered by traditional mapping techniques: (a) the pose uncertainty associated with 3D data can be incorporated in motion planning, (b) poses can be updated (i.e., from loop closures) with minimal computational effort, and (c) 3D data can be fused lazily for the purpose of planning. We provide an open-source implementation of NanoMap, and analyze its capabilities and computational efficiency in simulation experiments. Finally, we demonstrate in hardware its effectiveness for fast 3D obstacle avoidance onboard a quadrotor flying up to 10 m/s.**

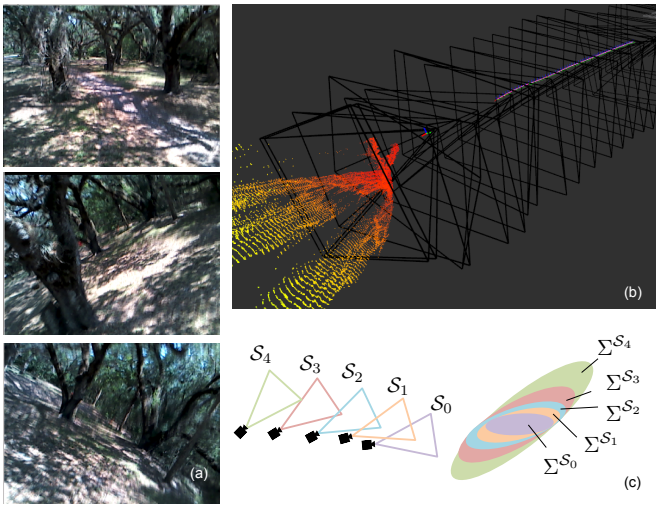

Fig. 1: (a) Onboard images from a quadrotor using NanoMap and flying at 10 m/s in a forest. (b) Visualization of vehicle's depth camera frustums over time, and current point cloud observing a tree. (c) Depiction of frame-specific uncertainty $\Sigma^{\mathcal{S}_i}$ for each depth sensor measurement frame $\mathcal{S}_i$.

I. INTRODUCTION

Robust, fast motion near obstacles is an open problem that is central in robotics, with applications spanning across manipulation, autonomous cars, and UAV navigation in unknown environments. Although many approaches exist for planning obstacle-free motions, mapping errors due to significant state estimation uncertainty can degrade their performance [1], [2]. Accordingly, a notable trend in the state of the art has been to develop memoryless approaches to obstacle avoidance that use only the current depth sensor measurement [1]–[5]. These approaches are less prone to state estimation errors, but fail to capture all available information.

Towards this goal, a primary motivation of this work was to be able to use pose uncertainty to reason about a local history of depth information. NanoMap is an algorithm and data structure that enables uncertainty-aware proximity queries for planning. While traditional mapping approaches rely on fusing a history of depth information into a discretized world frame, we propose an alternative: perform no discretization, and no fusing. Instead, the process for querying local 3D data is a *search over views*. When a query point (i.e. a sample along a motion plan) is provided, the history of depth information is searched for the most-recent and therefore minimum-uncertainty relative to current body frame view of that query point.

In practice, this approach offers a variety of unique capabilities not present in traditional fusion-based mapping algorithms. For one, the pose uncertainty associated with depth sensor measurements can be incorporated into planning, by treating each pose with frame-specific uncertainty relative to the current body frame (Fig. 1, c). Second, since fusion between measurements is not performed, it is trivial to incorporate updated information about previous poses. Third, the build time of the data structure is low, which leads to an improvement in computational efficiency for small amounts of motion planning queries ($< 10,000$).

This paper presents the design of NanoMap and our experiments in quantifying the benefits of its novel properties. We believe this work strongly demonstrates that jointly considering motion planning and perception can improve a system's robustness and computational efficiency. To briefly clarify our scope of work: (a) we focus on a method of incorporating pose uncertainty, but modeling the noise of the depth sensor itself is outside of scope, (b) NanoMap requires nonzero volume depth sensors, i.e. depth cameras or 3D lidars, but not 2D or 1D sensors, (c) adding more sensors to increase the FOV is a hardware route to alleviate

[1]CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA
{peteflo,jcarter,jakeware,russt}@csail.mit.edu

the problem but does not address occlusions, and (d) we are concerned with local obstacle avoidance, rather than global planning, and so short histories of information are sufficient.

The contributions of this work are as follows:

- A novel use of frame-specific uncertainty for planning with depth sensors
- An approach to search a history of depth frustums to enable motion plans to satisfy field of view constraints
- An efficient use of independently spatially partitioned depth measurements for motion planning queries
- Simulation experiments demonstrating the magnitudes of state estimation uncertainty at which frame-specific uncertainty becomes significant
- Hardware validation demonstrating this approach onboard a quadrotor, including flight at up to $8-10\ m/s$ in unknown warehouse and forest environments

II. MOTIVATION

This work seeks a method to reason about local 3D obstacles in the presence of significant state estimation uncertainty. Our approach is guided by our experience with high-speed UAVs, the use of depth sensors for obstacle avoidance, and the planning challenges introduced by imperfect state estimation [2]. In practice, depth sensor data (Fig. 1, b) is often clean enough that fusing many recent observations is not required in order to plan obstacle-free motions. Furthermore, the current or very recent depth measurements frequently contain a view of planned directions of motion (Fig. 1, b). In the case that the planned trajectory does not fall within the current field of view, it is still possible to perform robust trajectory planning by using the history of depth measurements.

To assess the utility of maintaining a history of depth measurements, we consider a mobile robot with fixed orientation, limited field-of-view depth sensors. For this configuration, the robot can can either guarantee safety by limiting its motion plans to within the known region of the environment or accept some risk of collision and allow motion plans to extend into the unknown region of the environment. For a quadrotor with a rigidly mounted depth camera using memoryless obstacle avoidance, we can approximately relate (derived in the Appendix) the maximum allowable safe speed to the depth camera's maximum depth sensing range and its vertical FOV:

$$v_{max} = \sqrt{\frac{2 d_{range} g}{3} \tan\left(\frac{FOV_{vertical}}{2}\right)}$$

Plotting this function (Figure 2, left) for a range of values and annotating with the specifications of common depth sensors, we see that this limits the maximum speed of a quadrotor with a 10 m sensing range and 45 $\deg$ FOV to 5.2 m/s. In this scenario, the FOV limitation is the primary factor limiting the quadrotor's top speed. We would like to relax the FOV constraints such that safe top speeds are limited only by the sensing range and the maximum achievable constant-altitude deceleration. The results of this relaxation can be seen in Figure 2, right. In particular, note that safe top speed with a 10 m sensing range is increased to 10.6 m/s for a vehicle with a 2:1 thrust to weight ratio. These approximations suggest how a small amount of obstacle memory can have significant impact on flight performance. Only a few seconds are needed of memory for an emergency stop for the range of values in Figure 2.

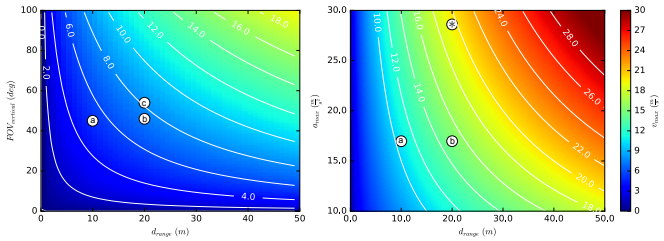

Fig. 2: (Left) Maximum safe top speed (v_{max}) for memoryless, constant-altitude flight with a depth camera. Maximum sensor range (d_{range}) and vertical FOV are superimposed for: (a, ASUS Xtion), (b, Intel RealSense r200), (c, Stereolabs ZED). Range specifications are based on empirical evaluations. (Right) Maximum safe top speed (v_{max}) as a function of maximum sensor range (d_{range}) and maximum horizontal deceleration (a_{max}). The mentioned sensing ranges together with an achievable deceleration for our vehicle (approximately 2:1 thrust ratio) are superimposed (a,b). The $(*)$ represents a 3:1 thrust ratio.

III. FORMULATION

NanoMap is a framework composed of both a local 3D data structure and an algorithm for searching that data structure. Briefly, the algorithm works by reverse searching over time through sensor measurement views until finding a satisfactory view of a subset of space (Figure 3), and then returning the k-nearest-neighbors from that view's sensor measurement. Important components of the framework include: the determination of in-frame views (the IsInFOV() function), the propagation of uncertainty, and efficient data structure design for handling asynchronous data inputs of point clouds, poses, and pose updates. We first describe the query algorithm, which gives insight into efficient data structure design. We then discuss details of handling asynchronous data.

A. Querying Algorithm

The query algorithm (Algorithm 1) iteratively transforms an uncertain query point into the coordinate frames of previous sensor measurements until it finds a view which contains the query. The query point in the original body frame and each of the relative transforms are each modeled with Gaussian translational uncertainty. In each frame associated with a given sensor measurement $\mathcal{S}_i$, the query point $\mathbf{x}_{query}^{\mathcal{S}_i} = \mathcal{N}(\mu^{\mathcal{S}_i}, \Sigma^{\mathcal{S}_i}) \in \mathbb{R}^3$ has uncertainty specific to that frame.

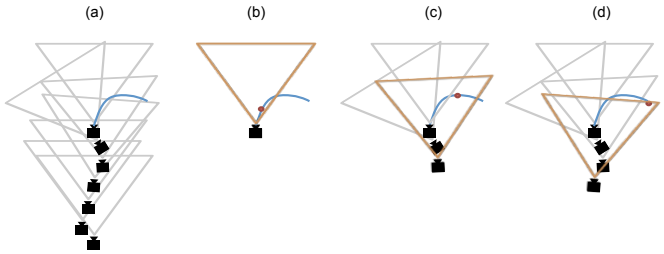

Fig. 3: Depiction of how NanoMap queries can be used to evaluate motion plans, (a, blue), given a series of depth sensor measurements over time (a, camera frustums). For each sample point (b, c, d; red) the history of measurements is searched until a view is found (orange) that contains the sample point. Note that sample points are actually Gaussian sample points (See Figure 5).

Algorithm 1: NanoMap query algorithm. Subroutine IsInFOV() is described in Section 4; Knn() is provided by a single-frame k-d-tree query. N is the number of measurements stored in memory.

function NanoMapQuery ($\mathbf{x}^{\mathcal{B}}_{query}$);
  Input : body frame query point $\mathbf{x}^{\mathcal{B}}_{query} = \mathcal{N}(\mu^{\mathcal{B}}, \Sigma^{\mathcal{B}})$
  Output: i, index of frame containing view
    $\mathbf{x}^{\mathcal{S}_i}_{query} = \mathcal{N}(\mu^{\mathcal{S}_i}, \Sigma^{\mathcal{S}_i})$
    k-nearest-neighbors $\mathbf{x}^{\mathcal{S}_i}_1$, ..., $\mathbf{x}^{\mathcal{S}_i}_k$
Transform query point from body frame into most recent sensor frame:
  $\mathbf{x}^{\mathcal{S}_0}_{query} \leftarrow \mathcal{N}(T^{\mathcal{S}_0}_{\mathcal{B}}\mu^{\mathcal{B}},\ \Sigma^{\mathcal{S}_0}_{\mathcal{B}} + R^{\mathcal{S}_0}_{\mathcal{B}}\Sigma^{\mathcal{B}})$
if *IsInFOV*($\mathbf{x}^{\mathcal{S}_0}_{query}$) **then**
  return 1, $\mathbf{x}^{\mathcal{S}_0}_{query}$, Knn($\mu^{\mathcal{S}_0}$);
end
for $i \leftarrow 1$ **to** N **do**
  Transform query point into previous frame:
    $\mathbf{x}^{\mathcal{S}_i}_{query} \leftarrow \mathcal{N}(T^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}\mu^{\mathcal{S}_{i-1}},\ \Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}} + R^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}\Sigma^{\mathcal{S}_{i-1}})$
  if *IsInFOV*($\mathbf{x}^{\mathcal{S}_i}_{query}$) **then**
    return i, $\mathbf{x}^{\mathcal{S}_i}_{query}$, Knn($\mu^{\mathcal{S}_i}$);
  end
end
return "out of known space", $\mathbf{x}^{\mathcal{S}_0}_{query}$, Knn($\mu^{\mathcal{S}_0}$);

1) IsInFOV(): determining in-frame views: A key challenge is in determining which view contains the uncertain point, referred to as the IsInFOV() function. Projecting the mean of the uncertain point into the depth image, as described in Figure 4, can be used to efficiently check a series of inequalities (inside each of lateral and vertical FOV, occluded, not beyond sensor horizon) to determine if the point is in free space. A challenge, however is represented by Figure 5. If only the mean of the distribution is used to check whether or not a view contains the point, then a large portion of that distribution may lie outside the FOV. With infinite-tail Gaussian distributions, no view fully contains them. NanoMap approximates this problem by using an axis-aligned bounding box (AABB), a familiar concept for fast approximations in the graphics community. The AABB for the 1-σ (1 standard deviation) of the distribution is used. Checking whether or not the AABB is contained can be done efficiently with the same number of inequality evaluations as the single point. To check for occlusions, NanoMap performs a simple occlusion check of the mean point.

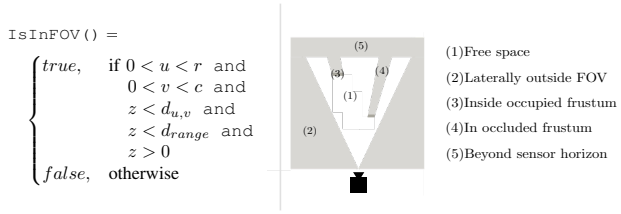

Fig. 4: The IsInFOV function (left) for determining if a query point is in freespace or one of the regions of non-freespace (right). The pixel coordinates u, v can easily be calculated by $(u, v) = (\frac{x}{z}, \frac{y}{z})$ and $(x, y, z) = K\mu$ with K the camera intrinsics matrix, and the query point $\mu \in \mathbb{R}^3$ in the right-down-forward Cartesian frame of the sensor measurement. $r \times c$ is the depth camera resolution.

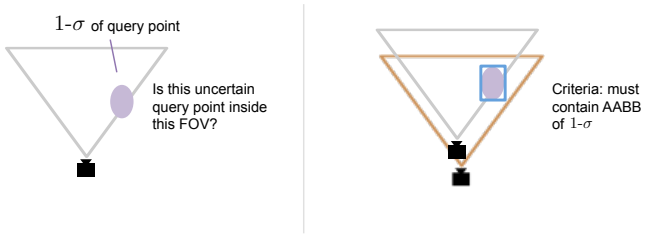

Fig. 5: Which view is sufficient for evaluating an uncertain query point distributed as an infinite-tail Guassian? A more recent measurement (left) may contain the mean of the distribution, but a significant portion may fall outside. Our criteria (right) for an approximate solution is to use the AABB of the 1-σ of the query point distribution. If no view fully contains the AABB, then the most recent frame is used, which may have had a partial view (Algorithm 1).

2) Uncertainty propagation: Accounting for uncertainty is performed as follows. The query is provided as the mean and covariance of a point in the current body frame $\mathcal{B}$ of the robot $\mathbf{x}^{\mathcal{B}}_{query} = \mathcal{N}(\mu^{\mathcal{B}}, \Sigma^{\mathcal{B}})$. The query is first transformed into the frame $\mathcal{S}_0$ of the most recent sensor measurement, $\mu^{\mathcal{S}_0} = T^{\mathcal{S}_0}_{\mathcal{B}}\mu^{\mathcal{B}}$, where $T^{\mathcal{S}_0}_{\mathcal{B}}$ represents the local, relative transform between the current body frame and the recent sensor frame. $T^{\mathcal{S}_0}_{\mathcal{B}}$ is modeled with a noisy translation $\mathcal{T}^{\mathcal{S}_0}_{\mathcal{B}}$ with covariance $\Sigma^{\mathcal{S}_0}_{\mathcal{B}}$, and known rotation $R^{\mathcal{S}_0}_{\mathcal{B}}$. In addition to computational simplification, our choice to model translational uncertainty and not rotational is guided by the practical observation that due to gravity, IMUs provide good observability of roll and pitch, and yaw is only a single integration of a noisy gyrometer (covariance grows $\propto N$ for N measurements), whereas positions are double integration of the accelerometer (covariance grows $\propto N^3$). Under the assumption of independence between body-frame query point uncertainty and each transform covariance, the variance of the query point in frame $\mathcal{S}_0$ is simply the sum

$\Sigma^{\mathcal{S}_0} = \Sigma^{\mathcal{S}_0}_{\mathcal{B}} + R^{\mathcal{S}_0}_{\mathcal{B}} \Sigma^{\mathcal{B}}$. Extending this process to the ith sensor coordinate frame, we have

$$\Sigma^{\mathcal{S}_0} = \Sigma^{\mathcal{S}_0}_{\mathcal{B}} + R^{\mathcal{S}_0}_{\mathcal{B}} \Sigma^{\mathcal{B}}$$

$$\Sigma^{\mathcal{S}_i} = \Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}} + R^{\mathcal{S}_i}_{\mathcal{S}_{i-1}} \Sigma^{\mathcal{S}_{i-1}} \quad \text{for } i = 1, 2, ...N$$

and concatenating transforms for the mean we have

$$\mu^{\mathcal{S}_i} = \prod_{j=1}^{i} \left[T^{\mathcal{S}_j}_{\mathcal{S}_{j-1}} \right] T^{\mathcal{S}_0}_{\mathcal{B}} \mu^{\mathcal{B}}$$

which defines $\mathbf{x}^{\mathcal{S}_i}_{query} = \mathcal{N}(\mu^{\mathcal{S}_i}, \Sigma^{\mathcal{S}_i})$.

B. Data Structure for Asynchronous Data

The data structure (Figure 6) matches the form of the query algorithm and is performant given the requirements of asynchronous data and continuous addition and removal of data. The core data structure is a chain of edge-vertex pairs, where the edge is the transform $T^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}$ and the vertex contains both the raw point cloud data and the previously-processed k-d-tree. The raw point cloud data (row-column-organized) is used to evaluate the IsInFOV() function, whereas the k-d-tree is used to evaluate k-nearest-neighbors if IsInFOV()=true.

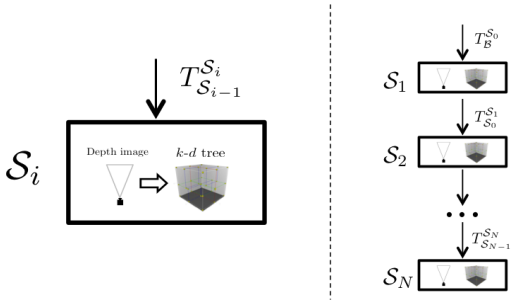

Fig. 6: Core NanoMap data structure: a sequence (right) of edge-vertex pairs (left), where each vertex contains both the raw data for evaluating FOV constraints, and a k-d-tree of the point cloud data. The edge is a relative transform to the coordinate frame of the next vertex.

We briefly highlight some data structure design considerations. By nature NanoMap is never defined in one coordinate frame, and rather has components in many coordinate frames. One implication of this is that NanoMap must constantly be updating $T^{\mathcal{S}_0}_{\mathcal{B}}$ with each new pose. Further, we desired both fast insertion of a new edge-vertex pair, and fast removal of the oldest edge-vertex pair. Since search through the data structure is also always performed linearly, a doubly-linked list of edge-vertex pairs is a good fit for these requirements, efficiently supporting $O(1)$ addition/removal at ends, and $O(1)$ incremental linear step. An additional feature given the separate-frame nature of the framework and asynchronous data is that the k-d-tree of a point cloud can be built even before the pose of the point cloud can be determined, allowing the k-d-tree building to begin before a world-frame map would be capable of starting insertion. Lastly, a key feature of NanoMap is to be able to efficiently handle asynchronous updated recent pose information, which may only cover a subset of its history. Upon receiving a series of updated world-frame poses, NanoMap only updates a transform edge $T^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}$ if it can fully interpolate the updated world frame pose of both vertices. This can be done efficiently by searching through the edge-vertex chain with a time-sequenced list of pose updates.

TABLE I: NanoMap Data Inputs and Parameters

Data Input	**Note**	**Example Rate**
6-DOF poses	timestamped	100 Hz
6-DOF pose corrections	sequence of timestamped poses	1-100 Hz
Organized 3D Point-Clouds	organized (row,column) from depth camera	30-60 Hz
Parameter	**Note**	**Example Values**
Max sensor range		10-20 m
Depth camera resolution, FOV	equivalently, K matrix	320x240, 60 deg V, 90 deg H FOV
N, history length (# point clouds)		150-300
$\Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}$ covariance	between sensor poses	0.005 - 0.02 m

IV. RESULTS

We start by (A) analyzing in simulation how NanoMap is able to provide robust obstacle avoidance depsite significant state estimation uncertainty, and quantify the scale of drift and correction jumps (i.e., from a loop closure) at which this is significant. We then (B) analyze the computational efficiency of NanoMap compared to other available packages for evaluating local 3D data in motion planning. Finally, (C) we demonstrate NanoMap used effectively on a real hardware system.

A. Robustness of NanoMap to State Estimation Uncertainty

A central goal of NanoMap was to increase obstacle avoidance robustness in regimes of significant state estimation uncertainty. There are two separate features we evaluate: the ability to separately model pose uncertainty of each depth measurement, and the ability to efficiently correct recent pose information from a sliding-window state estimator. Our hypothesis was that at some threshold of pose uncertainty, these features become relevant. Here we present our findings.

1) Experimental: Motion Planner and Simulation: In these experiments, NanoMap is used by a stochastic motion planner. This motion planner was as described previously [2], with the following modifications: (a) a full 3D motion primitive library of 125 primitives, (b) a collision-chance-constrained (maximum allowed collision probability of 0.001) rather than mixed-objective described previously, and (c) "early-exit" for subsequent sampling of a primitive that already evaluates below the chance constraint. Our simulation system was also as described in [2], here used with a professional-grade urban environment created in the Unity game engine. The simulated depth camera was 30 Hz, 20 m range, and 46 $deg\ FOV_{vertical}$.

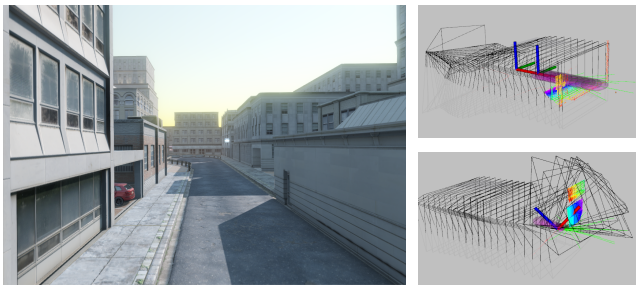

Fig. 7: Simulation environment (left) at beginning of experimental scenario. (Right top) Ground truth pose of vehicle and corrupted pose of vehicle (axes are 10 m, for scale). (Right below) Aggressively stopping when far wall comes into depth sensor range.

2) Scenario: The ability of NanoMap to provide pose-uncertainty-aware queries is most relevant when a motion planner is forced to search deeper into its history of poses. As discussed later with Figure 12, this is most apparent during extreme dodging maneuvers. Accordingly the experiments use the following scenario which is desirable due to its ease of interpretability: a quadrotor, initially at 5 m altitude, is given a desired goal 200 m away, with a desired top speed of 15 m/s, and 100 m along its path there is a large wall of a building with a 3D overhang near its altitude. The vehicle must aggressively decelerate, such that its velocity is outside of its current FOV. Significant pose uncertainty during this aggressive deceleration period would be difficult for other mapping and planning systems to handle.

3) Using Pose-Uncertainty-Aware Queries: To evaluate the magnitude of pose drift at which NanoMap's frame-specific uncertainty capability measurably increases robustness, we experimented with the following controlled experiment. As we increased state estimation noise, we either had NanoMap model the local, relative transforms with no translation covariance, $\Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}} = \mathbf{0}$, or with a covariance corresponding to the noise level, $\Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}} = f(\Sigma_{actual})$. Our noise model was to add noise to each of the x and y acceleration measurements, $\tilde{a} = (a + \eta) \times \xi$, where $\eta \sim \mathcal{N}(0, \Sigma_{actual})$ and $\xi \sim \mathcal{N}(1, \Sigma_{actual})$. Acceleration noise was integrated into the corrupted velocities and positions. Since quadrotors can measure altitude directly with downward-facing lidars and barometers, we did not model noise in z. An intuitive grasp of the scale of the noise model is best described as the standard deviation of drift over the depth measurement history (5 seconds = 150 measurements at 30 Hz) during the final portion of the flight. We term this $\sigma_{\text{drift, 5 seconds}}$, and accordingly used $f(\Sigma_{actual}) = \frac{\Sigma_{\text{drift, 5 seconds}}}{150}$. The singular difference between the two groups of the data (Figure 8) was the value of the $\Sigma^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}$ parameter in NanoMap.

These experiments show (Figure 8) that incorporating pose uncertainty can have a substantial effect, in particular when the drift is on the order of 10 cm per second. At very small drift ($\sigma_{\text{drift, 5 seconds}} = 0.4\ m$), there is little noticeable difference, but at $\sigma_{\text{drift, 5 seconds}} = 0.7,\ 1.5,\ 3.8\ m$, incorporating the uncertainty enables the vehicle to still stay safe 97-98 % of the time, whereas the drift deteriorates the safety of the group that doesn't incorporate pose uncertainty. The ability to stay safe diminishes at massive levels of drift (7.3 m in 5 seconds), where the pose-uncertainty-modeled group only stays safe 10 % of the time, but still more than the unmodeled group. The pose-uncertainty-modeled group on average stays much farther away from obstacles, (γ = distance to closest obstacle), playing it conservative during the aggressive maneuver.

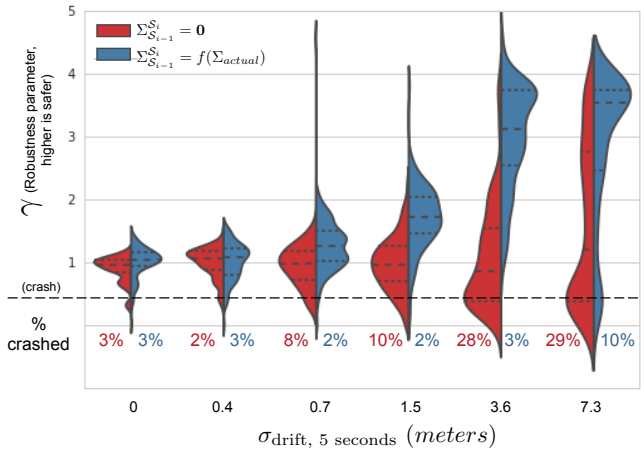

Fig. 8: Robustness (y axis, where the robustness criteria γ is the closest distance to an obstacle during the flight) of either modeling (blue) or not modeling (red) local pose uncertainty. A sampling of noise levels (x axis) are shown, which represent $\Sigma_{\text{actual}} = \{0, 0.05, 0.1, 0.2, 0.5, 1.0\}$. With a vehicle radius of 0.4 m, everything below the dotted line represents a crash. % crash is labeled for each case. 1200 total trials are represented.

4) Incorporating Updated Pose Information: NanoMap has a unique ability to efficiently update recent pose information, which as shown later in Figure 11, is not possible at realtime rates for the other benchmarked packages. This is meaningless, however, without getting a sense of when this capability is useful. Rather than provide a drifting state estimate, as in the previous experiment, we instead provide a deterministic backwards "pose correction" during the deceleration event (triggered at 12 m/s during the deceleration). This is representative of a loop closure occurring in the global state estimator. NanoMap is configured to either use the pose corrections to update and maintain a smooth history of poses, or only add new poses as they come in, and accordingly have a large "jump" in its history.

We find that even at the scale of 0.5 m pose corrections, this size of a pose jump in its history can measurably cause crashes during the aggressive maneuver scenario, causing 18 % crashes. With pose corrections of 2 m or more, these jumps cause crashes more than 50 %. By using NanoMap's capability to trivially update its entire pose history upon receiving a sliding-window correction (orange), there is expectedly little effect for any level of pose jump tested, with crashes occurring less than 5 % at all levels.

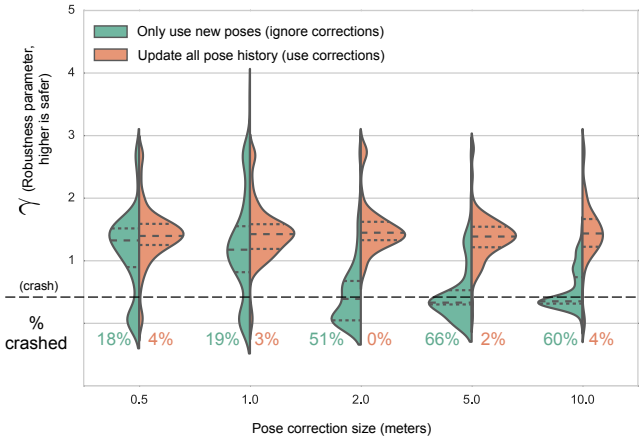

Fig. 9: Robustness (y axis, where the robustness criteria γ is the closest distance to an obstacle during the flight) of either incorporating updated pose history information (orange) after a loop closure or not (green). A sampling of pose correction size (x axis) are shown. As in Fig. 8, crash percentages are noted. 1000 total trials are represented.

B. Computational Efficiency Benchmarking

We compare NanoMap to three other packages: OctoMap [6], Voxblox [7], and Ewok [8]. OctoMap implements an octree occupancy grid, Voxblox builds ESDFs (euclidean signed distance functions) out of projective TSDFs, and Ewok builds its ESDF by iterating over a 3D circular buffer occupancy grid. Each of these can provide nearest-obstacle queries, which makes them efficient for stochastic motion planning, where there is uncertainty in configuration. There are of course many parameters for each of these packages, but we have made best efforts to provide a useful comparison given reasonable parameter choices. For both benchmarking experiments, we used a data log of a quadrotor with a simulated 320×240 depth image with 20 m range traversing an approximately 200 $m \times 200$ m urban environment. This dataset, and the scripts for using each of these packages to generate the benchmarking data, are available[1].

We use two metrics to measure the packages. The first metric (Figure 10) measures total time to incorporate a new sensor measurement and then perform $n_{queries}$ nearest-obstacle queries. The second metric (Figure 11) measures total time to adjust or rebuild a data structure after n_{poses} poses are corrected, i.e. after a loop closure.

There are a number of conclusions to draw from the plots. There is a tradeoff inherent from Figure 10 between the fusion-based packages (OctoMap, Voxblox, Ewok) which spend more time building their data structure, and NanoMap which spends less time building the data structure but has more expensive queries. For small amounts of queries, this tips the computational advantage to NanoMap, whereas for large amounts of queries, the fusion-based packages have an advantage. Figure 10 also demonstrates that unlike the discretized, fused packages, NanoMap has variable query time, based on how deep in history the query searches. We plot both the worst-case (each query searches the full history) and best-case (each query is in current FOV). In practice, our planner on average has approximately 75% best-case queries, but it is important to specify the system to worst-case timing, since as shown in Figure 12, more memory is used during critical dodging maneuvers. In the range of queries of our motion planner (2,500 queries), NanoMap is the fastest, even in the worst-case.

[1]https://hub.docker.com/r/flamitdraper/mapping/

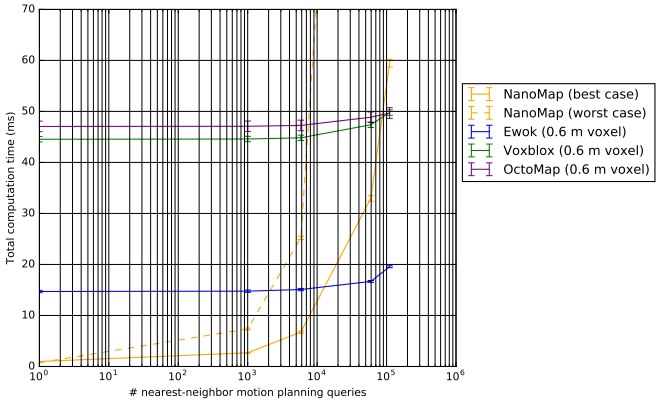

Fig. 10: Total computation time for $n_{queries}$ nearest-neighbor queries. NanoMap depth history is set to 150. Error bars are shown as standard error of the mean.

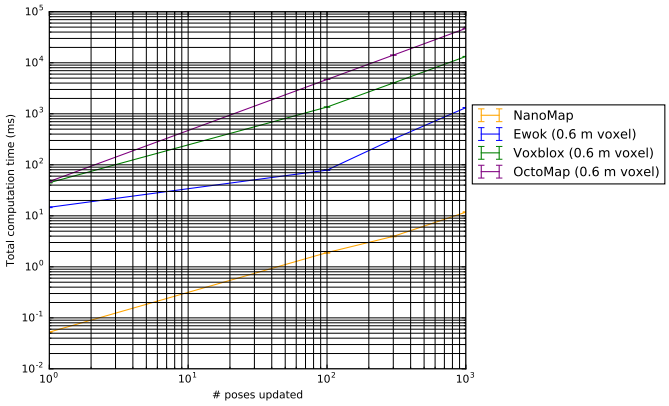

Fig. 11: Total computation time for adapting map structure to update n_{poses}. Error bars are standard error of the mean.

From Figure 11, we see a unique capability of NanoMap – its ability to incorporate updated pose information at realtime rates. NanoMap is two to four orders of magnitude faster than the others – this is not a capability that is feasible at realtime rates for the other packages for more than a handful of n_{poses}. Whereas the only way to incorporate new pose information for the other packages is to rebuild the data structure with new world-frame-registered measurements, NanoMap can adjust by simply updating the relevant sequential transforms ($T^{\mathcal{S}_i}_{\mathcal{S}_{i-1}}$) in its data structure.

We also have been able to empirically validate that for obstacle avoidance motion planning in our flight regimes, a large percentage of NanoMap queries fall within the current or very recent FOV of the depth sensor. For a

representative flight, we measure a very strong sufficiency of recent measurements, with 74.4% of queries falling within the current FOV, and a cumulative 92.3% of queries satisfied with the last 40 measurements. Figure 12 shows a histogram plotted over time during the course of the flight. During aggressive obstacle avoidance maneuvers (between $\sim$ 9 to 11 seconds into flight), there is expectedly more of a need to use memory. Yet even during this period, the last few seconds of flight mostly suffice for satisfying motion planning queries.

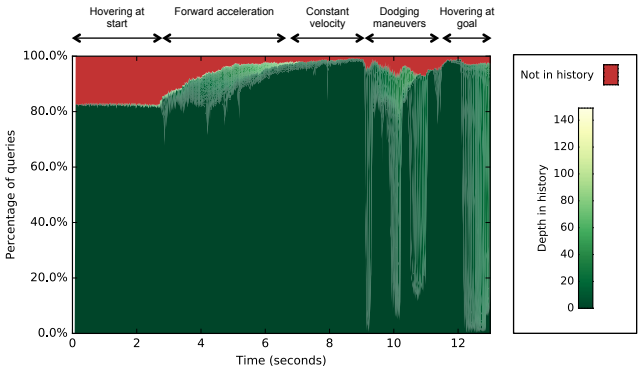

Fig. 12: Histogram over time for depth of history is searched in a representative flight. This data is from a 13 second flight traveling approximately 50 m in low clutter with a 30 Hz depth sensor, 10 m range, and 45 deg $FOV_{vertical}$. Phases of flight are labeled above the time axis.

C. Hardware Experimentation

NanoMap has been extensively used in our hardware system on our MIT-Draper DARPA FLA[2] team. Figure 1, a, shows images from onboard video of a flight. Over the course of a week of experimental testing at the May 2017 FLA event, NanoMap was the local mapping representation used for the majority of flights, with both an Intel RealSense r200 (for outdoor environments) and an ASUS Xtion (for indoor environments) used as the depth camera sensor. A Hokuyo 2D lidar sensor also aided obstacle perception for many of these flights, but it was used in a memoryless fashion, and due to its 0-deg vertical FOV was not useful during aggressive high-attitude maneuvers. We also (see video) demonstrate flight using only the RealSense, with no Hokuyo lidar. A sliding-window visual inertial (VIO) state estimator [9] with 100 Hz low-latency poses and lower-rate, higher-latency pose corrections over a 5-second sliding window. NanoMap incorporated these sliding window pose corrections. The mapping, planning, and hardware systems have been described in the author's Master's Thesis. Notable other vehicle hardware includes: a dual-core Intel NUC i7, a 450 mm Flamewheel DJI frame, and monocular Point Grey Flea3 camera and ADIS 16448 IMU for visual-inertial state estimation.

[2]DARPA Fast Lightweight Autonomy program: https://www.darpa.mil/program/fast-lightweight-autonomy

Our hardware experimentation with NanoMap demonstrates its robustness and applicability to high-speed obstacle avoidance. We have flown at up to 10 m/s in forested canopy environments with the Intel r200 (empirically, we observe 20+ m range in high-texture environments), and 8 m/s in indoor warehouse environments with the ASUS Xtion (empirically, we observe $\sim$ 8-10 m range). Flights in these types of settings can be seen in our video.

V. RELATED WORK

A few related works share some features of using pose estimation uncertainty in planning, but do not address planning around obstacles in unknown environments. Previous works have used directly the uncertainty of a pose graph framework for planning but have a critical limitation that they only plan over graphs of pre-known poses [10], [11]. Other work seeks to develop generalized belief space that includes distributions over worlds, but there are no obstacles in these worlds, only landmarks for navigation [12]. Another related work includes a sampling of depth perception estimates (a discrete probability distribution), but inserts them into a map structure using maximum-likelihood poses [13].

Rather than deal with the belief space of previous poses, the predominant approach for incorporating memory has been to ignore pose uncertainty, and use a maximum-likelihood mapping approach [14], [15]. Mapping-based approaches benefit from extensive decades of research into the robot mapping problem. When they are well constrained, many SLAM approaches are able to create precise maps that are the maximum likelihood estimate map $\hat{\mathcal{M}}_{MLE}$ from the fusion of a variety of noisy depth sensor, RGB, and other sensor data. There are a variety of different ways to formulate a map – the most common version are occupancy grids, which are used ubiquitously [6]. Occupancy grids can probabilistically incorporate depth sensor measurements (multiple measurements can be required for a cell to be occupied), but this doesn't address pose estimation uncertainty. Other forms include polar maps, and for some dense SLAM techniques, surfel maps are used.

A different and popular approach to the obstacle avoidance problem under significant state estimation uncertainty is to essentially cut pose estimation out of the equation, which can be done via a method that uses no memory of depth sensor measurements. In addition to planning-based approaches that exhibit this property [1]–[5], [16], any obstacle avoidance approaches that are considered reactive approaches may inherently have this property as well. Reactive approaches, including optic flow methods [17], reactive imitation-learning [18], and non-planning-based geometric approaches [19] have demonstrated considerable success at obstacle avoidance for UAVs. The limitations of memoryless obstacle avoidance have been well noted, however [16], [18]. Other related approaches have limited map-building to very short time horizons [20], or have used map structures that exponentially decay old depth sensor measurements [13].

VI. CONCLUSION

We have described, implemented, analyzed, and validated NanoMap. NanoMap provides novel features for using local 3D data with pose uncertainty. We have shown that for state estimation drift on the order of tens of cm/s, or state estimate position corrections on the order of 1 m, using NanoMap's uncertainty-aware features can substantially increase robustness. Given these results, we believe NanoMap is a compelling, novel route forward when compared to the traditional, fusion-first paradigm of mapping for planning. We would encourage future work that may draw inspiration from NanoMap and supplement traditional mapping approaches. NanoMap is open source and available at `github.com/peteflorence/nanomap_ros`.

APPENDIX

During constant-altitude high speed flight, in order to keep the horizon inside its field of view, a quadrotor with fixed orientation depth camera cannot pitch more than half of the vertical FOV of the camera: $\psi_{\text{max}} = \frac{FOV_{vertical}}{2}$ and due to the dynamics of a quadrotor [21], a maximum allowable pitch ψ_{max} can easily be related to a maximum allowable horizontal constant-altitude deceleration: $a_{FOV-limited} = g \tan\left(\frac{FOV_{vetical}}{2}\right)$. Given a maximum allowable deceleration, in order to keep the stopping distance d_{stop} inside of the depth sensor's maximum range d_{range}, then the allowable speed must be limited. With a simple double-integrator approximation of the quadrotor dynamics, solving $d(t) = \frac{1}{2}at^2 + v_0 t$ and $v(t) = at + v_0$ for $t = \frac{-v_0}{a}$ gives the maximum allowable speed for a stopping distance d_{stop}: $v_0 = \sqrt{\frac{2d_{stop}a}{3}}$. Plugging in for $a = a_{FOV-limited}$ and setting $d_{stop} = d_{range}$, we now have a function that relates the vertical FOV and range to the maximum allowable speed.

ACKNOWLEDGMENT

The authors would like to thank the rest of the MIT-Draper FLA team, particularly Brett Lopez, Kris Frey, Jonathan How, Ted Steiner, William Nicholas Greene, and Nicholas Roy. This work was supported by the DARPA Fast Lightweight Autonomy (FLA) program, HR0011-15-C-0110.

REFERENCES

[1] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1484–1491.

[2] P. Florence, J. Carter, and R. Tedrake, "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps," in *Algorithmic Foundations of Robotics XII*, 2016.

[3] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3242–3249.

[4] S. Daftry, S. Zeng, A. Khan, D. Dey, N. Melik-Barkhudarov, J. A. Bagnell, and M. Hebert, "Robust monocular flight in cluttered outdoor environments," *arXiv preprint arXiv:1604.04779*, 2016.

[5] B. T. Lopez and J. How, "Aggressive 3-d collision avoidance for high-speed navigation," in *ICRA (Accepted but not published*, 2017.

[6] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0

[7] H. Oleynikova, Z. Taylor, M. Fehr, J. Nieto, and R. Siegwart, "Voxblox: Building 3d signed distance fields for planning," *arXiv preprint arXiv:1611.03631*, 2016.

[8] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and 3d circular buffer," *arXiv preprint arXiv:1703.01416*, 2017.

[9] T. J. Steiner, R. D. Truax, and K. Frey, "A vision-aided inertial navigation system for agile high-speed flight in unmapped environments: Distribution statement a: Approved for public release, distribution unlimited," in *Aerospace Conference, 2017 IEEE*. IEEE, 2017, pp. 1–10.

[10] R. Valencia, M. Morta, J. Andrade-Cetto, and J. M. Porta, "Planning reliable paths with pose slam," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 1050–1059, 2013.

[11] E. H. Teniente, R. Valencia, and J. Andrade-Cetto, "Dense outdoor 3d mapping and navigation with pose slam," 2011.

[12] V. Indelman, L. Carlone, and F. Dellaert, "Towards planning in generalized belief space," in *Robotics Research*. Springer, 2016, pp. 593–609.

[13] D. Dey, K. S. Shankar, S. Zeng, R. Mehta, M. T. Agcayazi, C. Eriksen, S. Daftry, M. Hebert, and J. A. Bagnell, "Vision and learning for deliberative monocular cluttered flight," in *Field and Service Robotics*. Springer, 2016, pp. 391–409.

[14] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1872–1878.

[15] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5332–5339.

[16] M. W. Otte, S. G. Richardson, J. Mulligan, and G. Grudic, "Path planning in image space for autonomous robot navigation in unstructured environments," *Journal of Field Robotics*, vol. 26, no. 2, pp. 212–240, 2009.

[17] A. Beyeler, J.-C. Zufferey, and D. Floreano, "Vision-based control of near-obstacle flight," *Autonomous robots*, vol. 27, no. 3, pp. 201–219, 2009.

[18] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[19] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 50–56.

[20] A. J. Barry, "High-speed autonomous obstacle avoidance with pushbroom stereo." PhD Thesis, MIT, 2016.

[21] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Rob. Res.*, vol. 31, no. 5, pp. 664–674, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1177/0278364911434236