Autonomous Miniature Blimp Navigation with Online Motion Planning and Re-planning

Jörg Müller Norman Kohler Wolfram Burgard

Abstract— **In recent years, there has been an increasing interest in autonomous navigation for lightweight flying robots in indoor environments. Miniature airships, which are an instance of such robots, are especially challenging since they behave nonlinearly, typically are under-actuated, and also are subject to drift. These aspects, paired with their high-dimensional state space, demand efficient planning and control techniques. In this paper, we present a highly effective approach to autonomous navigation of miniature blimps in mapped environments which applies a multi-stage algorithm to accomplish strongly goal-directed tree-based kinodynamic planning. It performs path-guided sampling and optimally selects actions leading the robot towards sampled subgoals. Based on this, our approach can quickly provide a partial trajectory, which is extended and refined in the consecutive planning steps. The navigation system has been implemented and is able to reliably operate a robotic blimp in a real-world setting. Further experiments demonstrate that our approach outperforms a standard tree planner.**

I. Introduction

Recently, unmanned aerial vehicles (UAVs) have become a growing research field because such robots can navigate freely in three-dimensional environments. In this domain, especially airships have a couple of substantial advantages. Their low power consumption, safe navigation capabilities, and robustness to collisions allows for applying them in long-term operation tasks. We envision a wide range of applications including surveillance, disaster scenarios, communication, and advertising even in the presence of people, e.g., in public spaces.

However, these favorable properties come at the cost of some challenges imposed on autonomous navigation for blimps. The very limited acceleration capabilities together with the serious under-actuation make it practically infeasible to neglect second-order dynamics. Due to the nonlinear, non-holonomic, and drift-prone system, kinodynamic motion planning has to be performed in the 12-dimensional state space consisting of the pose and velocity of the robot. Furthermore, the cost of the shortest path in general does not follow any metric [15] and the commonly applied decoupling of trajectory shape and velocities is not applicable.

In this paper, we consider the task of indoor navigation on a round trip in a mapped environment, e.g., in a continuous surveillance task. We present an approach to online autonomous navigation including the state estimation, a multi-stage planner, a mission control module, and a controller.We approximate the planning problem, which in general is PSPACE-hard [22], by first applying A* search to generate a collision-free path on a discretized low-dimensional subspace of the state space. In the second stage, a tree planner quickly generates a trajectory through path-guided sampling which is extended and refined in consecutive planning cycles by re-using the pruned tree. We show the reliability and performance of our navigation system in extensive experiments in simulation and with a real robotic blimp (see Fig. 1). Furthermore, we show that our planning approach outperforms a standard goal-biased RRT planner [12].

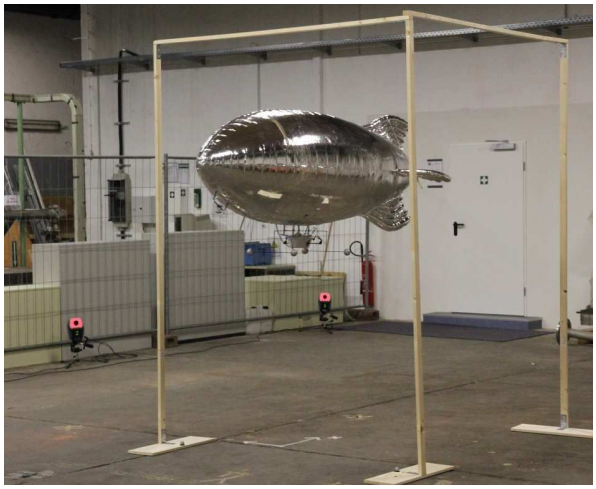

Fig. 1. The robotic indoor blimp [25] operating in the experimental setting observed by cameras of the motion capture system. The blimp is about 2.1 m long, has a payload of about 150 grams, and is actuated by three propellers.

This paper is organized as follows. After discussing related work in the following section, we describe the underlying models of our approach to autonomous navigation in Section III. In Section IV we introduce our multi-stage planner. We will then briefly present our controller implementation in Section V and finally demonstrate the capabilities of our approach in simulation experiments and with a real robotic blimp.

II. Related Work

In the past, several authors considered the problem of autonomous navigation of blimps. The majority of approaches, however, concentrated on control of robotic blimps in the absence of obstacles.

Some authors successfully applied model-free learning to control a single selected degree of freedom of a real indoor

This work has partly been supported by the German Research Foundation (DFG) within the Research Training Group 1103 and by the Gottfried Wilhelm Leibniz Program of the DFG. All authors are members of the Department of Computer Science at the University of Freiburg, Germany muellerj@informatik.uni-freiburg.de

blimp [13], [24]. In contrast to that, Liu *et al.* [18] and Zufferey *et al.* [29] learned controllers for the full state space of the blimp. However, both papers report a large number of iterations when learning a controller specific for a single trajectory or goal configuration. Several model predictive approaches have been proposed namely decoupling of components [11], extending the classic LQR [8], or nonlinear control [19]. As opposed to the LQR controller we designed for our navigation system, the tuning of the control design parameters of those controllers usually is time-consuming.

During the last decades, motion planning for mobile robots has been an area of active research [4], [16] and spawned sampling-based planning techniques such as rapidly-exploring random trees (RRTs) and probabilistic roadmaps (PRMs) which proved to be effective means to solve high-dimensional planning problems. However, controls needed for connecting nearby states cannot be calculated analytically in nonlinear kinodynamic motion planning problems [27] and thus PRM and bidirectional RRT techniques cannot be applied. Instead, sampling-based tree planners [5] like (standard) RRTs [17] were widely applied to such problems. Kim and Ostrowski [12] proposed an RRT with goal biasing for blimp motion planning. However, their planner was designed for an outdoor blimp in an obstacle-free environment. Ladd and Kavraki [14] account for under-actuation and drift of robots most planners are suffering from. Unlike our approach, they aim to explore the full state space which is time-consuming even in not very complex scenarios.

The concept of multi-stage planning was popular for discrete goal-directed online motion planning in real robot applications [3], [9], [26]. Rickert *et al.* [23] and Plaku *et al.* [20], [21] presented tree planners that quickly explore the lower-dimensional workspace of the robot through multiple paths to the goal which are not necessarily collision-free. In contrast to that, our planner is guided by one collision-free low-dimensional path which is additionally augmented by velocity information. This results in a more focused exploration of the state space and enforces that even partial trajectories do not tend to a dead end.

In our approach, we apply a novel combination of many existing techniques in order to build an efficient navigation system which has several desirable properties. It takes into account obstacles, is strongly goal-directed and efficient, and therefore suited for online navigation on a real indoor blimp.

III. State, Control, and Environment Models

We model our blimp which is shown in Fig. 1 as a floating rigid body in a three-dimensional environment. Consequently, its state is described by its pose and velocity in the 12-dimensional state space $\mathcal{X} \subseteq \mathrm{SE}(3) \times \mathbb{R}^6$. The blimp is actuated by three propellers. Two of them are mounted beside the gondola and are pivoted together to provide thrust along the forward- and the up-axis. The third propeller is mounted laterally near the bow of the blimp for yaw rotation. The blimp can be controlled by a three-dimensional vector $\mathbf{u} \in \mathcal{U} = [-1, 1]^3$ defining the relative forward, upward, and rotational thrust about the vertical axis.

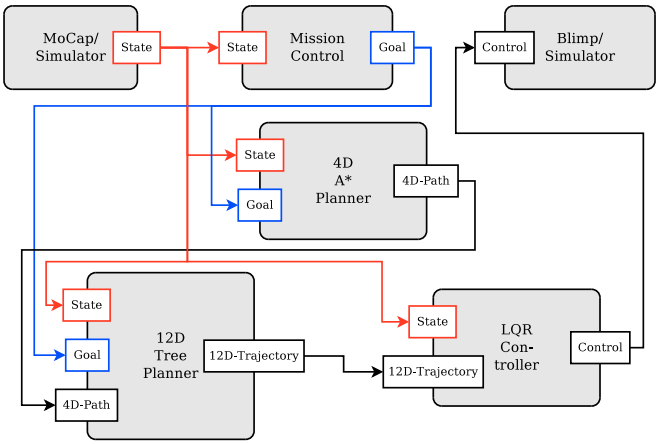

Fig. 2. The interaction of the modules of our approach to autonomous navigation. In the simulation experiments, we replaced the state estimation of the motion capture (MoCap) system and the blimp executing controls by a simulator module.

In our approach, we physically model the dynamics of the blimp according to Zufferey *et al.* [29]. In each time step, we compute the external (gravity, lift, thrust, and air drag) and fictitious forces and torques acting on the blimp. We compute the subsequent state based on the Newton-Euler equation of motion

$$M \frac{d\mathbf{v}}{dt} = \mathbf{F}_{\text{external}}(\mathbf{x}) + \mathbf{F}_{\text{fictitious}}(\mathbf{x}) \tag{1}$$

in a numerical integration where $\mathbf{v}$ is the velocity part of the state space and M is the inertia matrix. In a nutshell, the motion model can be described as a function

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{2}$$

of the state $\mathbf{x}_t$ and the control $\mathbf{u}_t$ at time t.

The previously mapped environment of the blimp is modeled using the OctoMap framework [28], which provides a tree-based map structure representing occupancy of 3D volume elements in a hierarchical fashion. For collision checking, we compute a distance map and conservatively approximate the blimp by a set of spheres which are arranged along its longitudinal axis. Since collision checking in this way is just a lookup to the distance map at the centers of the spheres, it can be done very efficiently.

IV. Planning Algorithm

We specify a certain state $\mathbf{g} \in \mathcal{X}$ as goal which should be reached within a predefined radius. The interaction of the planning modules of our approach to autonomous navigation is shown in Fig. 2. Our planning algorithm works in two stages. First, we apply A* search [10] to compute an optimal path assuming a simplified motion model on a discretized 4-dimensional subspace of the state space. In the second stage, a sampling-based tree planner searches for a trajectory in the full 12-dimensional state space efficiently by utilizing the A* path in order to draw samples in a goal-directed way. This prevents the tree planner from getting trapped, e.g., in a maze.

Algorithm 1 Tree-based re-planning with guided sampling

Input: Previous tree $\mathcal{T}$, planning timeout $t_{\max}$, augmented path $\mathcal{P}$, current state $\mathbf{x}$, goal $\mathbf{g}$
Output: A (partial) solution trajectory

1: Determine node $\mathbf{x}_{\text{start}}$ of $\mathcal{T}$ at time $t_{\max}$
2: Prune everything below $\mathbf{x}_{\text{start}}$ from $\mathcal{T}$
3: $\mathcal{T}$.root $= \langle \mathbf{x}_{\text{start}}, \mathbf{0} \rangle$
4: $\mathbf{x}_{\text{closest}} = \mathbf{x}_{\text{start}}$
5: **while** $t < t_{\max}$ **do**
6: $\quad \mathbf{x}_{\text{rnd}} \leftarrow$ GAUSSIANSAMPLEFROMPATH$(\mathcal{P})$
7: $\quad \mathbf{x}_{\text{near}} \leftarrow$ NEARESTNEIGHBOR$(\mathcal{T}, \mathbf{x}_{\text{rnd}})$
8: $\quad \mathbf{u}^\star \leftarrow$ OPTIMALACTION$(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{rnd}})$
9: $\quad \mathbf{x}_{\text{new}} \leftarrow f(\mathbf{x}_{\text{near}}, \mathbf{u}^\star)$
10: $\quad$ **if** COLLISIONFREE$(\mathbf{x}_{\text{new}})$ **then**
11: $\quad\quad \mathcal{T}$.insert$(\langle \mathbf{x}_{\text{new}}, \mathbf{u}^\star \rangle)$
12: $\quad\quad$ EXTENDSAMPLINGINTERVAL$(\mathcal{P}, \mathbf{x}_{\text{new}})$
13: $\quad\quad \mathbf{x}_{\text{closest}} \leftarrow$ UPDATECLOSEST$(\mathbf{x}_{\text{closest}}, \mathbf{x}_{\text{new}})$
14: $\quad$ **end if**
15: **end while**
16: **return** SOLUTIONTRAJECTORY$(\mathcal{T}, \mathbf{x}_{\text{closest}})$

A. Low-dimensional Optimal Path Generation

The low-dimensional path generation provides a collision-free path. In this planning step we consider a 4-dimensional subspace $\mathcal{X}' = \mathbb{R}^3 \times \mathrm{SO}(2)$ of the state space defined by the translation and the yaw-rotation of the blimp. This reduces the full state space by the velocity and the angles roll and pitch which are not directly controllable. To allow for path generation by A* search [10], we discretize this subspace into a grid and guide the search by a Euclidean distance heuristic. As the blimp can turn approximately on the spot when moving very slowly, we define the set of allowed actions as moving one grid cell forward, backward, upward, downward and rotating to the left and to the right. We define the resulting path computed on the 4D grid as $\mathcal{P}' = (\mathbf{x}'_1, \ldots, \mathbf{x}'_N)$ with $\mathbf{x}'_i \in \mathcal{X}'$ for all $i \in [1, N]$.

B. Path-guided Sampling-based Tree Planning

In a preprocessing step, our tree planner augments each new 4D path $\mathcal{P}'$ with a zero roll and pitch angle and velocity information resulting in the augmented (12D) path $\mathcal{P} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. The velocities contain the forward, upward, and yawing velocity and are determined based on the clearance to obstacles as well as the curvature of the 4D path and a maximum centripetal acceleration. Here, both, a low clearance or a high curvature, lead to a reduced velocity. Note that the augmented path $\mathcal{P}$ is not necessarily dynamically feasible but aims to focus the sampling of the tree planner to reasonable areas of the huge state space.

Algorithm 1 shows the pseudocode for our RRT-based [17] tree planning approach with path-guided sampling. In the first planning cycle, we initialize the tree with the current state propagated to the time $t_{\max}$ at which the planning cycle will be finished. In all subsequent planning cycles, we prepare the tree generated in the previous planning cycle for re-use by searching for the node that will be reached at $t_{\max}$ and pruning (line 1 to 3) similar to Bekris and Kavraki [1].

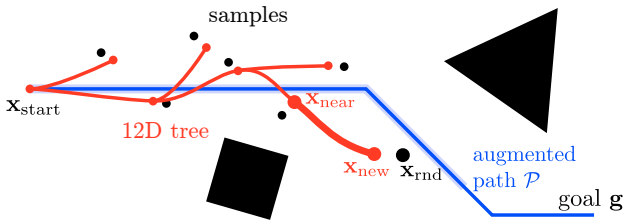

Fig. 3. An example of an extension step of the tree with path-guided sampling. The augmented A* path $\mathcal{P}$ is shown in blue, the current sampling interval is highlighted in light blue. The node $\mathbf{x}_{\text{rnd}}$ is sampled from $\mathcal{P}$. $\mathbf{x}_{\text{near}}$ which is the nearest node to $\mathbf{x}_{\text{rnd}}$ is extended towards $\mathbf{x}_{\text{rnd}}$ resulting in the new node $\mathbf{x}_{\text{new}}$. Obstacles are shown in black.

Fig. 3 shows an example of an extension step of the tree. GAUSSIANSAMPLEFROMPATH (line 6) utilizes the augmented path $\mathcal{P}$ to draw goal-directed samples from the state space. First, a position on the path is sampled from the current sampling interval. Then the sample $\mathbf{x}_{\text{rnd}}$ is drawn from a Gaussian with the chosen augmented path element as mean. This implicitly induces goal-biasing and ensures that a valid partial trajectory is returned if the time available for planning runs out before the tree reaches the goal.

Our tree planner selects the NEARESTNEIGHBOR which will be extended towards the sampled state $\mathbf{x}_{\text{rnd}}$ based on a weighted Euclidean metric [12]

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^T D (\mathbf{x}_1 - \mathbf{x}_2) \tag{3}$$

with a diagonal distance matrix D. Since the distance matrix is diagonal, we can find the nearest state efficiently by utilizing a kd-tree [7] containing all tree nodes scaled by the square root of D.

We optimally select the action leading from the nearest neighbor $\mathbf{x}_{\text{near}}$ towards the sampled state $\mathbf{x}_{\text{rnd}}$ as we will describe in Section IV-C.

Finally, the sampling interval on $\mathcal{P}$ (see Fig. 3) is extended according to the growth of the tree (line 12) and the node $\mathbf{x}_{\text{closest}}$ which is nearest to the goal is determined by UPDATECLOSEST (line 13). This can be done by selecting the node which is nearest to the end of the sampling interval on the path. It ensures a good choice even when only a partial trajectory has been computed and the robot has to veer away from the goal when navigating through a maze.

C. Optimal Action Selection

We select the optimal action $\mathbf{u}^\star$ leading from a state $\mathbf{x}_t$ towards a target state $\mathbf{x}^\star$ with respect to the metric ρ. This means that we want to select the action

$$\mathbf{u}^\star = \operatorname*{argmin}_{\mathbf{u}} \rho(f(\mathbf{x}_t, \mathbf{u}), \mathbf{x}^\star) \tag{4}$$

that minimizes the metric distance to the target state $\mathbf{x}^\star$ after executing one motion step starting from $\mathbf{x}_t$.

By linearizing the motion model with respect to the control around the state $\mathbf{x}_t$ and the neutral control $\mathbf{0}$, we obtain

$$f(\mathbf{x}_t, \mathbf{u}) \approx f(\mathbf{x}_t, \mathbf{0}) + \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_t, \mathbf{0})\, \mathbf{u}\ . \tag{5}$$

Plugging Eq. (5) and Eq. (3) into Eq. (4) results in

$$\mathbf{u}^\star = \underset{\mathbf{u}}{\operatorname{argmin}} (C_t\mathbf{u} + \mathbf{y}_t)^T D (C_t\mathbf{u} + \mathbf{y}_t) \tag{6}$$

$$= \underset{\mathbf{u}}{\operatorname{argmin}} \, \mathbf{u}^T C_t^T D \, C_t\mathbf{u} + \mathbf{u}^T C_t^T D \, \mathbf{y}_t + \mathbf{y}_t^T D \, C_t\mathbf{u} + \mathbf{y}_t^T D \, \mathbf{y}_t \tag{7}$$

$$= \underset{\mathbf{u}}{\operatorname{argmin}} \, \frac{1}{2}\mathbf{u}^T C_t^T D \, C_t\mathbf{u} + \mathbf{u}^T C_t^T D \, \mathbf{y}_t \tag{8}$$

with $C_t := \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_t, \mathbf{0})$ and $\mathbf{y}_t := f(\mathbf{x}_t, \mathbf{0}) - \mathbf{x}^\star$ by omitting the constant term and exploiting the symmetry of D.

In an unbounded control space this can be solved in closed form. For robots with bounded controls $\mathbf{u}_{\text{low}} \leq \mathbf{u} \leq \mathbf{u}_{\text{high}}$ like our blimp this problem can be solved efficiently, e.g., using a quadratic programming-based solver [6].

V. Controller

The trajectory computed by the planning algorithm consists of the full state and control information $(\mathbf{x}_t^\star, \mathbf{u}_t^\star)$ at discrete time steps $t \in [1, T]$. In order to keep the robot on this trajectory, we apply finite-horizon discrete-time linear-quadratic regulation (LQR) control [2].

The LQR controller aims to minimize a cost function

$$\mathrm{E}\left[\sum_{\ell=t}^{T} \left((\mathbf{x}_\ell - \mathbf{x}_\ell^\star)^T P (\mathbf{x}_\ell - \mathbf{x}_\ell^\star) + (\mathbf{u}_\ell - \mathbf{u}_\ell^\star)^T Q (\mathbf{u}_\ell - \mathbf{u}_\ell^\star)\right)\right] \tag{9}$$

which quadratically penalizes the expected deviation of the real state and control from those defined by the trajectory. Linearizing the motion model along the trajectory

$$A_t = \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_t^\star, \mathbf{u}_t^\star), \quad B_t = \frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}_t^\star, \mathbf{u}_t^\star) \tag{10}$$

so that

$$(\mathbf{x}_{t+1} - \mathbf{x}_{t+1}^\star) \approx A_t(\mathbf{x}_t - \mathbf{x}_t^\star) + B_t(\mathbf{u}_t - \mathbf{u}_t^\star), \tag{11}$$

gives the optimal control $\mathbf{u}_t$ at time t as

$$(\mathbf{u}_t - \mathbf{u}_t^\star) = L_t(\mathbf{x}_t - \mathbf{x}_t^\star). \tag{12}$$

The corresponding feedback matrix L_t is computed recursively for $M_T = P$ and $\forall \ell \in [t, T-1]$:

$$L_\ell = -(B_\ell^T M_{\ell+1} B_\ell + Q)^{-1} B_\ell^T M_{\ell+1} A_\ell \tag{13}$$

$$M_\ell = P + A_\ell^T M_{\ell+1} A_\ell + A_\ell^T M_{\ell+1} B_\ell L_\ell \tag{14}$$

in linear time with respect to the length of the trajectory.

VI. Experimental Evaluation

We implemented and evaluated the approach described above in simulation and with a real robotic blimp operating in a large indoor environment with two rooms. In our experiments, we consider the task of continuously navigating on a round trip specified in advance by an ordered set of goals shown in Fig. 4.

In the run-up to the experiment, we learned the parameters of the motion model described in Section III from about 10 minutes of manually operated flight observed by a Vicon motion capture system.

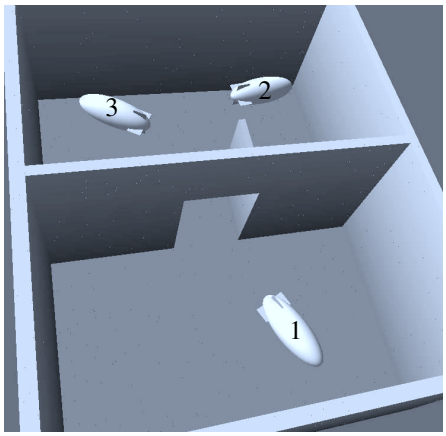

Fig. 4. The experimental environment consists of two rooms sized 8 m×6 m connected by an open door. The round trip navigation task is defined by three goals, which are set sequentially by the mission control module.

We chose a 0.25 m and 45 ° resolution for the low-dimensional A* path generation. For speedup, we precalculated the distance map of the environment and the numerically derived Jacobians of the motion model for the typical range of velocities. In all experiments, we set the planning timeout $t_{\max}$ to 1 sec. With this setting, the tree-based planner extended the tree by 150 to 500 nodes in each period depending on the initial tree size and the number of collisions.

The mission control module (see Fig. 2) continuously checks whether the blimp is approaching the current goal. If the distance to this goal drops below a threshold, it switches to the next goal and provides it to the planner modules. All experiments were run on an Intel® Core™ 2 Duo processor running at 2.53 GHz.

A. Simulation

Our simulation module handles control commands and simulates the motion of the blimp according to the parametric motion model learned from real recorded data of the blimp. Additionally, it provides the simulated position and velocity as state information. Due to the difference in time discretization used in the individual modules, the simulation deviates slightly from the prediction of the planner.

In an extensive experiment the online simulated blimp traveled for 110 min on a round trip and reached each of the 3 goals 70 times. For that, it calculated 6628 trajectories including all re-planning steps. The controller executed all trajectories without any collision. The calculation of the control feedback matrices for a new trajectory took 1.1 msec on average with a maximum value of 6.8 msec. The 4D A* planner took 14.4 msec on average with a maximum value of 81.7 msec for calculating a full path.

We compared our planning algorithm to an RRT planner with goal-biased sampling as proposed by Kim and Ostrowski [12] for airship navigation. We experimentally found that drawing 10% of the samples from a Gaussian around the goal was a good trade-off between exploration and exploitation. We also ran the goal-biased planner for

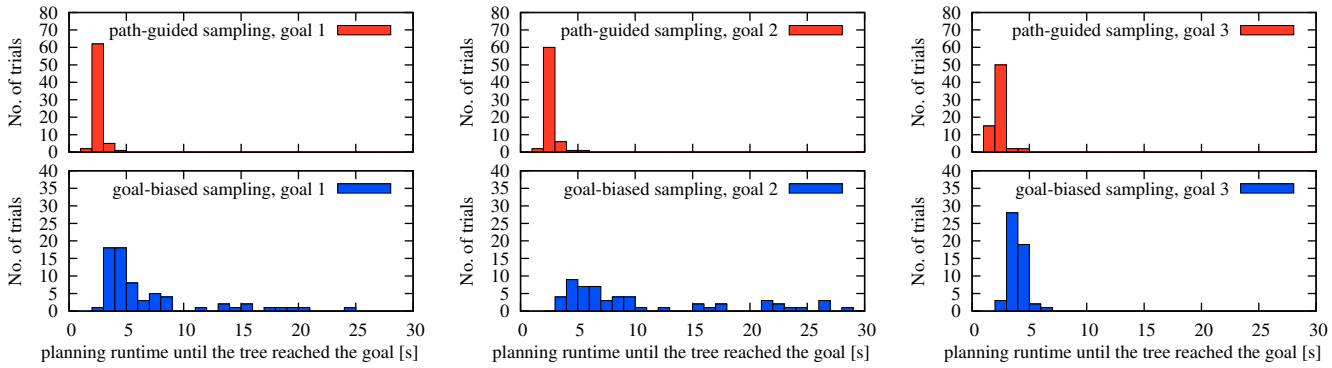

Fig. 5. Comparison of the planning times for three different goals of our path-guided and the goal-biased planner.

Goal	Path-guided		Goal-biased	
	Mean	StdDev	Mean	StdDev
1	35.2	13.6	35.1	19.1
2	51.4	10.0	62.7	18.2
3	18.4	6.9	22.4	11.7

Table 1. Comparison of travel times of the planned trajectories using path-guided and goal-biased sampling. All units are seconds.

110 min of online operation on the same round trip during which it failed in 14 of 191 attempts to plan a trajectory to a goal. In case of failure it created a partial trajectory during traveling which ran into a dead end and resulted in a collision. The planning times until the trajectory provided by the planner reached the goal are compared in Fig. 5. While our planning algorithm never exceeded 6 seconds, the goal-biased sampling resulted in a wide-spread distribution which caused the planner to fail in 14 attempts.

As shown in Table 1, the average travel time for the trajectories planned by our algorithm had a considerably lower standard deviation as the path-guided sampling appeared to be more goal-directed. In fact, the travel times resulting from our planning algorithm proved to be significantly lower in a paired t-test with a p-value of 0.6%.

B. Real Blimp

We conducted an extensive experiment with our real robotic blimp showing that the models used for planning and control are realistic and that our approach is able to deal with real noise and moderate modeling approximations. Our blimp [25] is about 2.1 m long and has an effective payload of about 150 grams used for a Gumstix computer communicating via WiFi with a standard laptop computer. Additionally, the blimp is equipped with a web-cam, an IMU, and sonar sensors. The blimp is actuated by two main propellers that are mounted beside the gondola and pivot together. The main propellers provide thrust in the forward/backward and upward/downward directions. A third propeller is mounted laterally near the bow of the blimp for yaw rotation (see Fig. 1).

For our experiments we did not use the on-board sensors to localize the blimp. Instead, we localized the blimp using a MotionAnalysis motion capture (MoCap) system with eight digital Raptor-E cameras tracking 4 reflective markers mounted around the gondola of the blimp (see Fig. 1). Due to practical reasons we only built up the door frame and the contour of the door as building up the all obstacles and walls would prevent from tracking the blimp with a reasonable number of MoCap cameras. Since the pose estimate provided by the MoCap system is very accurate (in our setting the error is typically below 3 mm), we additionally applied online collision checking based on MoCap pose estimates and the map.

The attached video shows an extract of the experiment in which the blimp autonomously traveled on the round trip for about 20 minutes and passed 28 goals without any collision. In this setup, the problem of Inevitable Collision States (ICS) [1] didn't arise due to the quick planning and the comparably low velocity of the goal states. Three exemplary trajectories are shown in Fig. 6. In our experiment, the root mean square (RMS) translational deviation from the trajectory was 0.24 m and the RMS deviation in yaw was 8.6°. This is due to air motion caused by the air conditioning system of the adjacent clean rooms and the moderate approximations of the motion model. The RMS deviation from zero roll was 0.35°. Since the velocity profile on the 4D path guides the sampler to move at low velocities in the vicinity of obstacles, the blimp passed the door slowly in the majority of those safety-critical situations.

VII. Conclusions

In this paper, we presented an approach to autonomous navigation of a blimp in a known indoor environment including motion capture state estimation. To efficiently approximate the high-dimensional nonlinear kinodynamic motion planning problem, we apply a multi-stage planning technique. In the first stage a collision-free path is generated through A* search on a low-dimensional subspace of the state space. We utilize this path to efficiently generate kinematically feasible trajectories by goal-directed, path-guided tree planning in the full 12-dimensional state space. In contrast to other approaches, it optimally selects actions towards sampled subgoals and is able to quickly provide a possibly partial trajectory which is extended in the consecutive planning steps. Including a motion capture state estimate, a mission control module, and an LQR controller,

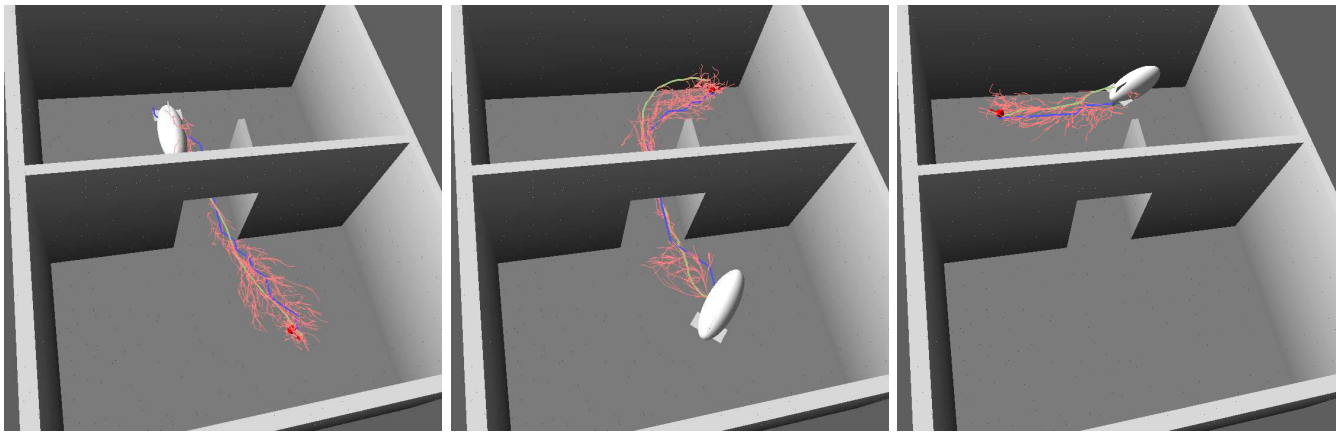

Fig. 6. Three exemplary trajectories generated by our path-guided planner during the experiment with the real robotic blimp. The goals are shown as a big red ball with an arrow indicating the desired orientation of the blimp. The four-dimensional A* path is shown as a thick blue line, the tree built by the tree planner is shown in red, and the chosen branch is marked by a thick yellow line.

our approach successfully controlled a robotic blimp. We performed extensive experiments in simulation and with a real robotic blimp. In all experiments, our navigation system efficiently and reliably operated the blimp and outperformed a standard goal-biased RRT planner. In future work we would like to consider autonomous navigation of the blimp with self-localization using the on-board sensors.

REFERENCES

[1] K.E. Bekris and L.E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[2] D.P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific, 2005.

[3] J. Chestnutt and J. Kuffner. A tiered planning strategy for biped navigation. In *Proc. of the IEEE - RAS / RSJ Conference on Humanoid Robots*, 2004.

[4] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavarki, and S. Thrun. *Principles of Robot Motion Planning*. MIT-Press, 2005.

[5] I.A. Şucan and L.E. Kavraki. On the implementation of single-query sampling-based motion planners. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

[6] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[7] J. H. Freidman, J. L. Bentley, and R. A Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[8] H. Fukushima, S. Ryosuke, M. Fumitoshi, Y. Hada, K. Kawabata, and H. Asama. Model predictive control of an autonomous blimp with input and output constraints. In *Proc. of the International Conference on Control Applications*, pages 2184–2189, 2006.

[9] J. Garimort, A. Hornung, and M. Bennewitz. Humanoid navigation with dynamic footstep plans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011. To appear.

[10] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. In *IEEE Transactions on Systems Science and Cybernetics*, pages 100–107, 1968.

[11] E. Hygounenc, I-K. Jung, P. Soueres, and S. Lacroix. The autonomous blimp project at LAAS/CNRS: Achievements in flight control and terrain mapping. *Int. Journal of Robotics Research*, 23(4), 2004.

[12] J. Kim and J.P. Ostrowski. Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[13] J. Ko, D.J. Klein, D. Fox, and D. Hahnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[14] A.M. Ladd and L.E. Kavraki. Fast tree-based expoloration of state space for robots with dynamics. In *Algorithmic Foundations of Robotics VI*, pages 297–312. Springer, STAR 17, 2005.

[15] A.M. Ladd and L.E. Kavraki. Motion planning in the presence of drift, underactuation and discrete systems changes. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[16] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[17] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In K.M. Lynch B.R. Donald and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, 2001.

[18] Y. Liu, Z. Pan, D. Stirling, and F. Naghdy. Q-learning for navigation control of an autonomous blimp. In *Proc. of the Australasian Conf. on Robotics & Automation (ACRA)*, 2009.

[19] A.B. Moutinho. *Modeling and Nonlinear Control for Airships Autonomous Flight*. PhD thesis, 2007.

[20] E. Plaku, L.E. Kavraki, and M.Y. Vardi. Discrete search leading continuous exploration for kinodynamic motion planning. In *Proc. of Robotics: Science and Systems (RSS)*, 2007.

[21] E. Plaku, L.E. Kavraki, and M.Y. Vardi. Impact of workspace decompositions on discrete search leading continuous exploration (DSLX) motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[22] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of the Symposium on Foundations of Computer Science*, 1979.

[23] M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

[24] A. Rottmann, C. Plagemann, P Hilgers, and W. Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[25] A. Rottmann, M. Sippel, T. Ziterell, W. Burgard, L. Reindl, and C. Scholl. Towards an experimental autonomous blimp platform. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2007.

[26] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 508–513, Lausanne, Switzerland, 2002.

[27] K.I. Tsianos, I.A. Şucan, and L.E. Kavraki. Sampling-based robot motion planning: Towards realistic applications. *Computer Science Review*, 1(1), 2007.

[28] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[29] J.C. Zufferey, A. Guanella, A. Beyeler, and D. Floreano. Flying over the reality gap: From simulated to real indoor airships. *Autonomous Robots*, 21(3), 2006.