Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles

David H. Shim[*] and Hoam Chung[†]
University of California, Berkeley, CA, 94720

H. Jin Kim[‡]
Seoul National University, Seoul, Korea

and

Shankar Sastry[§]
University of California, Berkeley, CA, 94720

In this paper, we present an autonomous exploration method for unmanned aerial vehicles in unknown urban environment. We address two major aspects of exploration- map building and obstacle avoidance- by combining model predictive control (MPC) with a local obstacle map builder. An onboard laser scanner is used to build the online map of obstacles around the vehicle during the flight. A real-time MPC algorithm with a cost function that penalizes the distance to the nearest obstacle replans the path. The adjusted trajectory is sent to the position tracking layer in the Berkeley UAV avionics. The proposed approach is implemented on Berkeley rotorcraft UAVs and successfully tested in urban flight experiment setup.

I. Introduction

UNMANNED aerial vehicles (UAVs) have become an indispensable platform for many applications where manned operation is considered unnecessary or too risky. As UAVs find their way into more demanding applications such as ground support or urban warfare, they are expected to fly autonomously without colliding into obstacles. So far, those situations have been avoided by operating UAVs at higher altitudes where chances of running into obstacles are very slim, or, if really necessary, by controlling UAVs with human operators in the loop. However, as UAVs are required to operate in cluttered or dynamically changing environments with more authority, the need for autonomous exploration capability is greatly increasing. Exploring autonomously in an unknown environment requires two crucial component technologies: map building and online trajectory replanning.

Map building and obstacle avoidance have been intensively covered by robotics society since late 70s. As a result of these efforts, various algorithms and implementations are currently available for guiding mobile robots safely in unknown or partially known two-dimensional world. Although some algorithms can be extended to the modeling of three-dimensional spaces, these are still computationally expensive, and have limitations in outdoor applications. In 1990s, researchers introduced probability theories into map building techniques,[1] enhancing robustness and performance of those algorithms even with cheaper and inaccurate sensors.[2] Naturally, intensive computations are required for these enhancements, and the situation becomes worse in three-dimensional problems. UAVs also pose more challenges than the ground robots do in the development and implementation. UAVs typically fly at a much faster speed than the ground robots, and therefore demand faster and more accurate decision making. The payload allocated to the avionics and onboard sensors is no less limiting than ground robots. Finally, during the

[*] Principal Research Engineer, Department of Electrical Engineering and Computer Science, Corresponding Author (hcshim@eecs.berkeley.edu)
[†] Ph.D. Candidate, Department of Mechanical Engineering (hachung@eecs.berkeley.edu).
[‡] Assistant Professor, School of Mechanical and Aerospace Engineering (hjinkim@snu.ac.kr).
[§] Professor, Department of Electrical Engineering and Computer Science (sastry@eecs.berkeley.edu).

development stage, trial-and-error approaches are strictly prohibited because any failure to avoid obstacles inescapably leads to a costly and very dangerous accident.

Model predictive control has been found highly attractive for addressing control problems in dynamic environments. The online optimization[3] with preview enables a control system more responsive to the changes in the system dynamics and the surroundings. Further, it has been suggested a variety of performance goals, in addition to the feedback stabilization, can be incorporated into the cost function. Shim, Kim, and Sastry[4] proposed MPC-based flight control algorithms by introducing a set of cost functions for decentralized collision avoidance and aerial pursuit-evasion.[5] Particularly, it is shown that MPC is capable of obstacle avoidance using a cost function that penalizes the distance to the nearest obstacle.

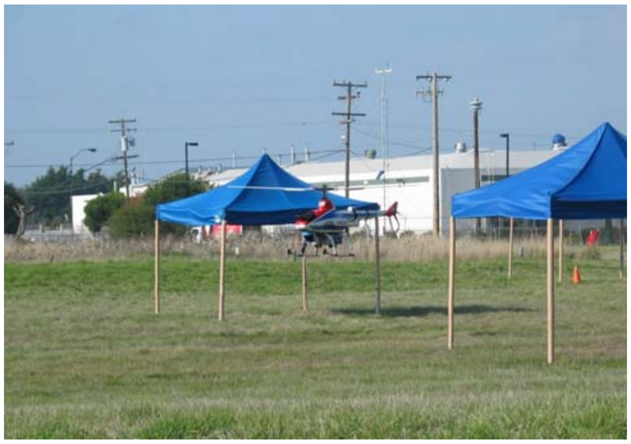
Figure 1. Berkeley UAV flying autonomously in a simulated urban environment

The obstacles in the flight path can be made known to the path planner by a pre-programmed map or a dynamically built obstacle map. Whereas the former approach does not suffer from any sensor-induced errors, the map itself may be inaccurate or outdated. Therefore, we favor onboard sensors, especially, the laser scanner due to its accuracy and long detection range.

In this paper, we propose an autonomous exploration algorithm suitable for, but not limited to, urban navigation by combining the MPC-based obstacle avoidance with local obstacle map building using onboard laser scanning. Starting from the given trajectory, the MPC layer solves for a collision-free trajectory by the real-time gradient-search based algorithm. The proposed algorithm is validated in simulations, and in experiments using a simulated urban environment as shown in Figure 1.

II. Formulation

In this section, we provide some background in the system dynamics, MPC formulation, and the coordinate transformation for laser scanning.

A. System Dynamics and Path Generation using MPC

A model for a given system dynamics can be written as a discrete-time dynamic equation such that:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) + \mathbf{g}(\mathbf{x}(k))\mathbf{u}(k) \tag{1}$$

We are interested in solving a *discrete-time optimal control* problem for the system in Eq. (1) to find the optimal input sequence $\left\{\mathbf{u}^*(k)\right\}_{k=1}^{T}$ such that

$$\left\{\mathbf{u}^*(k)\right\}_{k=1}^{T} = \arg\min \sum_{k=1}^{T} q\left(\mathbf{x}(k),\mathbf{u}(k)\right) + q_f\left(\mathbf{x}(T+1)\right) \tag{2}$$

subject to the dynamic equation (1).

Nonlinear model predictive control (NMPC) problem solves for the optimal control law $\left\{\mathbf{u}^*(k)\right\}_{k=1}^{T}$ for the nonlinear dynamic equation in Eq. (1), starting from $\mathbf{x}(1)$ and implement the optimal input $\left\{\mathbf{u}^*(k)\right\}_{k=1}^{\tau}$ for $1 \le \tau \le T$ and then repeat these steps from the state $\mathbf{x}(\tau+1)$ at $k = \tau + 1$.

Our model-predictive path planning strategy combines the potential field concept with the online optimization with preview. The cost function in Eq. (2) is formulated to reflect the aspect of a potential function for path planning in the environment with stationary or moving obstacles. This allows the trajectory generation and vehicle stabilization to be combined into a single problem, and the look-ahead feature of the MPC framework makes this approach less vulnerable to local minima.[4] In this scenario, we assume that each vehicle is aware of nearby obstacles by means of onboard sensors or a pre-programmed map.

The following potential function term is added to the cost function $q(\mathbf{x}(k),\mathbf{u}(k))$:

$$q^{obst}(\mathbf{x}(k)) = \sum_{i=1}^{N} \frac{K_i}{a_i(x^S(k)-x_i(k))^2 + b_i(y^S(k)-y_i(k))^2 + c_i(z^S(k)-z_i(k))^2 + \varepsilon}, \tag{3}$$

where (x^S, y^S, z^S) denotes the position of the vehicle, and (x_i, y_i, z_i) denotes the position of the i-th nearest obstacle (or the position of other vehicles) in local Cartesian coordinate frame. Eq. (3) introduces a potential field near N obstacle points into the MPC framework. (a_i, b_i, c_i) is a set of scaling factors in *x, y, z* directions, respectively. Note ε is a positive constant to prevent Eq. (3) from being singular when $(x^S, y^S, z^S) = (x_i, y_i, z_i)$.

It has been shown in Ref. 4, along with the detailed description on algorithms, that the proposed MPC framework allows the trajectory generation and the vehicle stabilization problems to be combined into a single problem. In Section III, we will present an obstacle avoidance algorithm using the MPC framework shown above for urban exploration problems.

B. Coordinate Transformation for Laser Scan Data

The laser scanner we adopted in our research consists of a laser source, a rotating mirror for planar scanning, and a laser receptor. The mirror reflects the laser beam in a plane. At each scanning, the sensor reports a stream of measurements that supplies the following information:

$$Y_L = \{(d_n, \beta_n), n = 1, \ldots, N_{meas}\}, \tag{4}$$

where d_n, β_n, and N_{meas} represent the distance from an obstacle, the angle in the scanning plane, and the total number of measurements per scan, respectively. Each measurement can be written into a vector form such that

$$\mathbf{X}_{D/L}^{L}\big|_n = d_n\left(\cos\beta_n \mathbf{i}^L + \sin\beta_n \mathbf{j}^L\right), \tag{5}$$

where $\mathbf{i}^L$ and $\mathbf{j}^L$ are orthonormal unit vectors lying in the scanning plane. The subscripts D and L represent scanned data and laser scanner, respectively.

In order to find the spatial coordinates of each detected point, we need a few coordinate transformations among three coordinate systems: body coordinate systems attached to the laser scanner and to the host vehicle, respectively, and the spatial coordinate system, to which the vehicle location and attitude are referred.

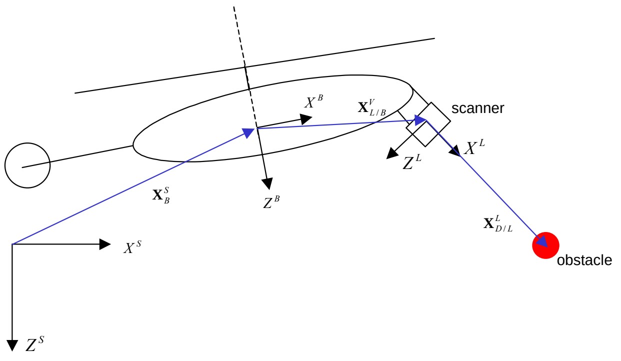

Figure 2. Coordinate transformations for laser scan data

In order to fly around obstacles in the surrounding environment, the laser scanner should be able to scan the area large enough to cover the space that the entire vehicle body may pass through with some clearance. For example, if the laser scanner is installed to scan the area horizontally, an actuation in the pitch axis is added so that the scanner can cover the area higher than the rotor plane and lower than the landing gear.

Each laser measurement vector in laser scanner-attached body coordinates is first transformed into the vehicle-attached body coordinates and then the spatial coordinate system as following:

$$\begin{aligned}\mathbf{X}_{D/L}^{S} &= \mathbf{R}_{S/L}\mathbf{X}_{D/L}^{L} \\ &= \mathbf{R}_{S/B}\mathbf{R}_{B/L}(\alpha)\mathbf{X}_{D/L}^{L}\end{aligned}, \tag{6}$$

where the subscripts S and B represent spatial- and body-coordinate system, respectively. $\mathbf{R}_{B/L}(\alpha)$ is the transformation matrix from the laser body coordinate L to vehicle body coordinate B where α is the tilt angle with respect to the vehicle body. If the laser scanner is mounted without additional tilting motion, $\mathbf{R}_{B/L}(\alpha)$ reduces to a constant matrix. $\mathbf{R}_{S/B}$ denotes the transformation matrix from vehicle body coordinates to spatial coordinates.

Finally, the spatial coordinate of the obstacle is found by:

$$\begin{aligned}\mathbf{X}_D^S &= \mathbf{X}_{D/L}^S + \mathbf{X}_{L/B}^S + \mathbf{X}_B^S \\ &= \mathbf{R}_{S/B}\mathbf{R}_{B/L}(\alpha)\mathbf{X}_{D/L}^L + \mathbf{R}_{S/B}\mathbf{X}_{L/B}^B + \mathbf{X}_B^S\end{aligned} \tag{7}$$

Using Eq. (7), one can find the spatial coordinate of a detected obstacle point by combining the raw measurement vector with the position and attitude of the vehicle, which is available from the INS and the GPS onboard. Note that the accuracy of the detection in the spatial coordinate system not only depends on the laser scanner itself, but also the accuracy of the vehicle position and attitude.

Figure 3 shows fixed and actuated types of laser scanner mountings on Berkeley UAVs. The scanner shown in the left is a fixed mounting and the one on the right is installed on a tilt actuator with a tilt angle encoder. It is noted the laser scanner on a fixed mounting can provide vertical scanning with a limited range due to the small body motion caused by the main rotor gyration.

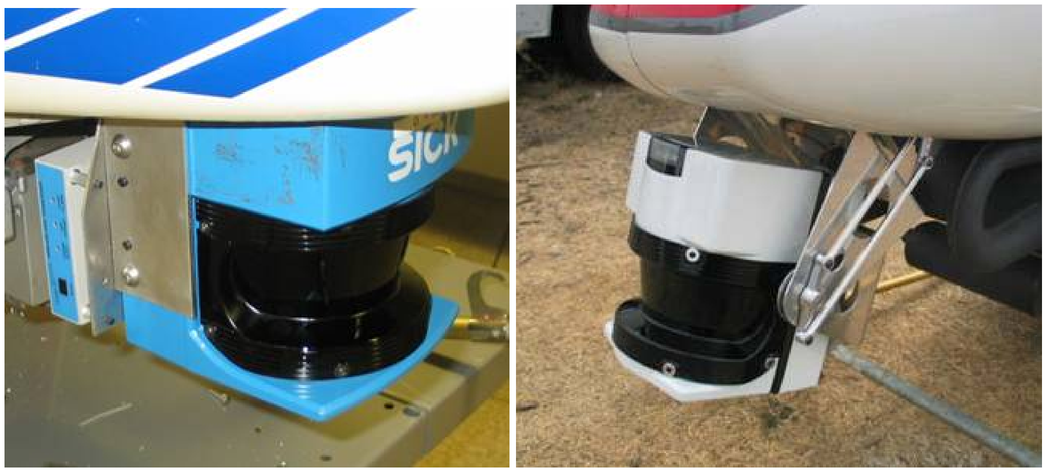

Figure 3. Laser scanning devices mounted on Berkeley UAVs (left: fixed, right: actuated mounting)

III. Autonomous Exploration using MPC with Local Maps

In this section, we present a MPC-based trajectory generation for autonomous exploration in an unknown environment with obstacles. Particularly, we are interested in addressing safe navigation of UAV in urban environments with no prior information available on the obstacles. We begin with the following statement:

Problem Statement

Find a trajectory that allows the vehicle to navigate from the given starting point A to the destination point B with safe distance from obstacles in the environment.

We address the problem with an integral approach of MPC-based trajectory planner with local obstacle map generation using onboard sensors.

A. Trajectory Replanning with MPC

In this section, we consider a navigation problem from point A to point B, connected by a reference trajectory. Without loss of generality, the trajectory is assumed as a straight line. The MPC approach in Section II-A can be formulated as a tracking control problem with a cost function term for Eq. (2) such that

$$q^{trk}(\mathbf{x}(k)) = \frac{1}{2}\left(\mathbf{y}_{ref}(k) - \mathbf{x}(k)\right)^T Q\left(\mathbf{y}_{ref}(k) - \mathbf{x}(k)\right), \tag{8}$$

where $q(\mathbf{x}(k)) = q^{trk}(\mathbf{x}(k)) + q^{obst}(\mathbf{x}(k))$ in Eq. (2).

In Ref. 4, Eq. (1) is chosen to be the full vehicle dynamic model so that the optimized variable $\mathbf{u}(k)$ is the stabilizing control input that also minimizes other penalties for tracking, obstacle avoidance, or aerial pursuit-evasion games. Although the MPC can be formulated either for direct stabilization or reference trajectory generation, we opt for the latter due to the safety during flight experiments using our UAVs. By separating the trajectory layer from the stabilization layer, any failure of the optimization routine to converge to a solution does not directly affect the stability of the overall vehicle. However, the difference between the reference position and the physical position due to the tracking error should be considered in the obstacle map building process in Section III-B. Therefore, in this paper, we choose a simplified dynamic model to generate a reference trajectory in order to lower the numerical load of the optimization process for experiment.

$$\mathbf{x}(k+1) = \mathbf{x}(k) + T_s \mathbf{u}(k), \tag{9}$$

where $\mathbf{x} \triangleq [x^S \quad y^S \quad z^S]^T, \mathbf{u} \triangleq \left[v_x^S \quad v_y^S \quad v_z^S\right]^T$ and T_s is the sampling time of the discretized model. In this setup, the optimization results in the reference velocity in the spatial coordinates. The optimization output is fed back into Eq. (9) to obtain the reference trajectory, which is then sent to the tracking layer in Berkeley UAV avionics.[6]

As described in Section II, an MPC-based trajectory generator can be formulated for obstacle avoidance as well. In Ref. 4, it was shown that the cost term (Eq. (3)) with N=1 is sufficient although Eq. (3) with $N > 1$ is expected to result in a smoother cost function surface and thus allow a better convergence in the gradient-search based optimization. In favor of efficiency of algorithms and computation, we choose the *nearest-point method,* i.e., N=1 so that only the nearest point is considered in the optimization(Figure 4). The cost term in Eq. (3) for urban navigation is set to

$$q^{obst}(\mathbf{x}(k)) = K_{obs}\left((x^S(k) - x_{\min}(k))^2 + (y^S(k) - y_{\min}(k))^2 + (z^S(k) - z_{\min}(k))^2 + \varepsilon\right)^{-1}, \tag{10}$$

where the cost function is chosen to decay uniformly in every direction from the obstacle point at $(x_{\min}, y_{\min}, z_{\min})$. K_{obs} is a tuning parameter to balance the tendency to stay on the original given path and to break away from the given path to go around obstacles.

B. Local Obstacle Map Building

For the MPC-based trajectory generation with obstacle avoidance, we need to find $\mathbf{X}_O^{\min}$, the relative vector with minimum length from the *reference position* to a point on an obstacle such that

$$\mathbf{X}_O^{\min}(\mathbf{X}_{ref}) = \underset{\mathbf{X}_O^i \in S_{obs}}{\arg\min} \left\|\mathbf{X}_O^i - \mathbf{X}_{ref}\right\|_2, \tag{11}$$

where $\|\cdot\|_2$ is Euclidian norm in three-dimensional space and S_{obs} is the set of all points on the obstacles in the surrounding three-dimensional space.

Theoretically, Eq. (11) demands a perfect knowledge on all obstacles in the surrounding environment, which would require an ideal sensor capable of omni-directional scanning with infinite detection range. Further, if the MPC is for reference trajectory generation, the ideal sensor should be moving along the trajectory of the reference points during the state propagation over a finite horizon at every optimization stage. Obviously, such a scenario is impractical. Therefore, in order to provide $\mathbf{X}_O^{\min}$ to the MPC-based trajectory

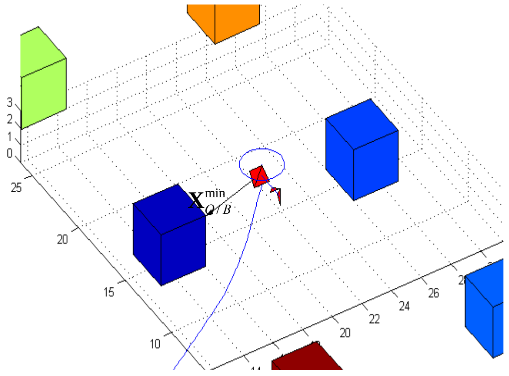

Figure 4. Nearest-point method

generator with these restrictions, it is necessary to maintain a local obstacle map consisting of recent measurements from onboard sensors.

At each sample time, the sensor provides N_{meas} measurements of scan points on obstacles nearby. Due to the imperfect coverage of the surroundings with possible measurement error, each measurement set $\mathbf{X}_O^i$ is first filtered, transformed into local Cartesian coordinates, and *cached* in the local obstacle map. A first-in first-out (FIFO) buffer is chosen for the data structure for such a map, whose update rate depends on the types of obstacles nearby. If the surrounding is known to be static, the caching time is desired to be as long as the memory and processing overhead permits. On the other hand, a more dynamic environment would require shorter caching to reduce the chance to detect an obstacle that may not exist any more.

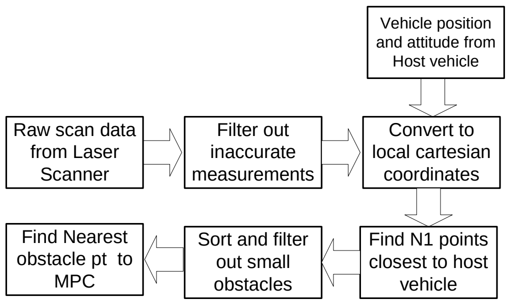
Vehicle position and attitude from Host vehicle
Raw scan data from Laser Scanner
Filter out inaccurate measurements
Convert to local cartesian coordinates
Find N1 points closest to host vehicle
Sort and filter out small obstacles
Find Nearest obstacle pt to MPC

Figure 5. Local partial map building method for nearest-point approach using MPC

In order to solve Eq. (11), the measurement set in the FIFO should be sorted in an ascending order of $\left\|\mathbf{X}_O^i - \mathbf{X}_{ref}\right\|_2$ for all $\mathbf{X}_O^i$ in the local obstacle map. Prior to be registered in the database, any anomalies such as *salt-and-pepper noise* are discarded. In case that the sensor detects small debris, such as grass blades or leaves blown by the vehicle, those small-size objects, not being a serious threat for safety, should be ignored by the trajectory planner. We apply an algorithm that computes a bounding box of the minimum volume that contains a series of subsequent points in FIFO where the distance between adjacent points in the sorted sequence is less than a predefined length. Then, if the volume of the box is larger than a threshold so that it is considered as a safety threat, the coordinates of the nearest point in the bounding box is found and used for computing Eq. (3). The procedure of the local obstacle map building method proposed above is illustrated in Figure 5.

IV. Experiment Results

In this section, we present the simulation and experiment results of autonomous exploration in an urban environment. The experiment design is strongly affected by the safety regulations: it is performed in a field with portable canopies simulating buildings, not with real buildings. The canopies, measuring 3 meter × 3 meter × 3 meter each, are arranged as shown in Figure 6. The distance between canopies is set to 10 meters in the north-south direction and 12 meters in the east-west direction so that the UAV with 3.5 meter long fuselage can pass between the canopies with minimal safe clearance, about 3 meters from the tip to a canopy when staying on course. For validation, the MPC engine developed in Ref. 4 is applied to the proposed urban experiment setup. A simulation

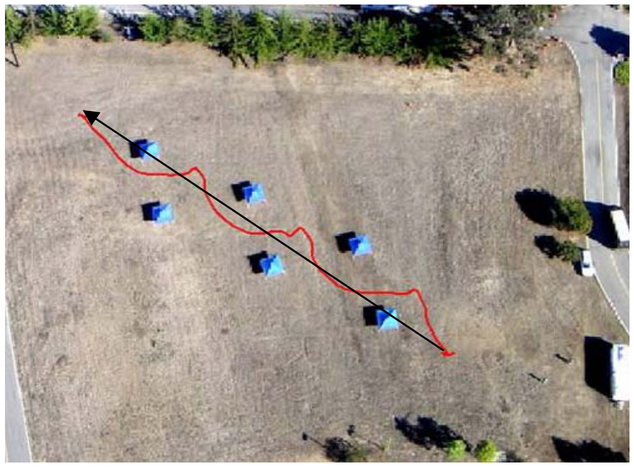

Figure 6. Aerial view of urban experiment (black: given straight path, red: actual flight path of UAV during experiment)

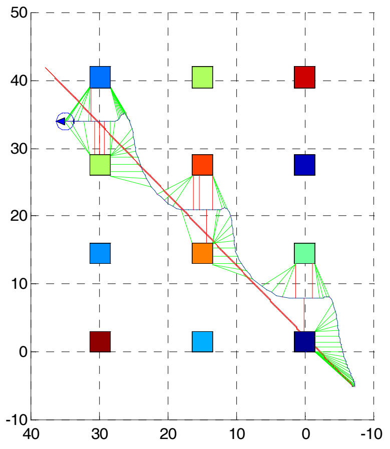

Figure 7. Simulation of MPC-based path planning in the proposed urban experiment setup

model is constructed in MATLAB^{TM}/Simulink^{TM}, where the local map building with a laser scanner is replaced with a pre-programmed map to avoid building a laser sensing model. The MPC algorithm with the local map building algorithm is implemented in C language for speed and portability.

As shown in Figure 7, the MPC path planner is capable of generating a collision-free trajectory around the buildings from the original trajectory with intentional intersection with buildings. The green and red lines pointing to the buildings are $\mathbf{X}_O^{\min}$ at each sample time. For experiments, the Simulink model is modified to function as the online trajectory generator. Although Simulink was not designed for a real-time controller in the loop, it can be forced to run for soft real-time control by adding a real-time enforcing block. Using the behavior of TCP/IP communication, a custom TCP/IP transport block is configured to enforce soft real-time operation of the Simulink model at 10Hz in this case.

Table 1. Specification of a Berkeley UAV

Base platform	Yamaha R-50 Industrial Helicopter
Dimension	0.7 m(W) × 3.5 m (L) × 1.08 m (H)
Rotor	Diameter 3.070 m
Weight	44 kg (dry weight) 20 kg (payload including avionics)
Engine	2 cycle gasoline engine 12 hp
Operation Time	Fuel: 40 minutes Avionics: 200 minutes
Onboard System	Two PC104-based computers Boeing DQI-NP INS NovAtel GPS MillenRT-2 IEEE 802.11b Wireless Ethernet Ultrasonic altimeters SICK laser range finder (LMS-200) Pan-tilt-zoom Camera
Capabilities	Waypoint navigation Position tracking Interactive operation mode

Urban exploration experiments were performed using a Berkeley UAV, whose detailed specification is given in Table 1. For obstacle detection, the vehicle is equipped with an LMS-200 from Sick AG (Figure 3), a two-dimensional laser range finder, which is capable of 80 m scanning range with 10 mm resolution and weighs 4.5 kg. The measurement is sent to the flight computer via RS-232 and then relayed to the ground station running the MPC-based trajectory generator in Simulink and the ground station software with a three-dimensional rendering window. The laser scanner data stream is then processed following the method proposed in Section III-B. In Figure 8, a three-dimensional rendering from the ground station software is given. The display shows the location of the UAV, the reference point marker, $\mathbf{X}_{O/B}^{\min}$ to a point in the local obstacle map at that moment, and laser-scanned points as blue

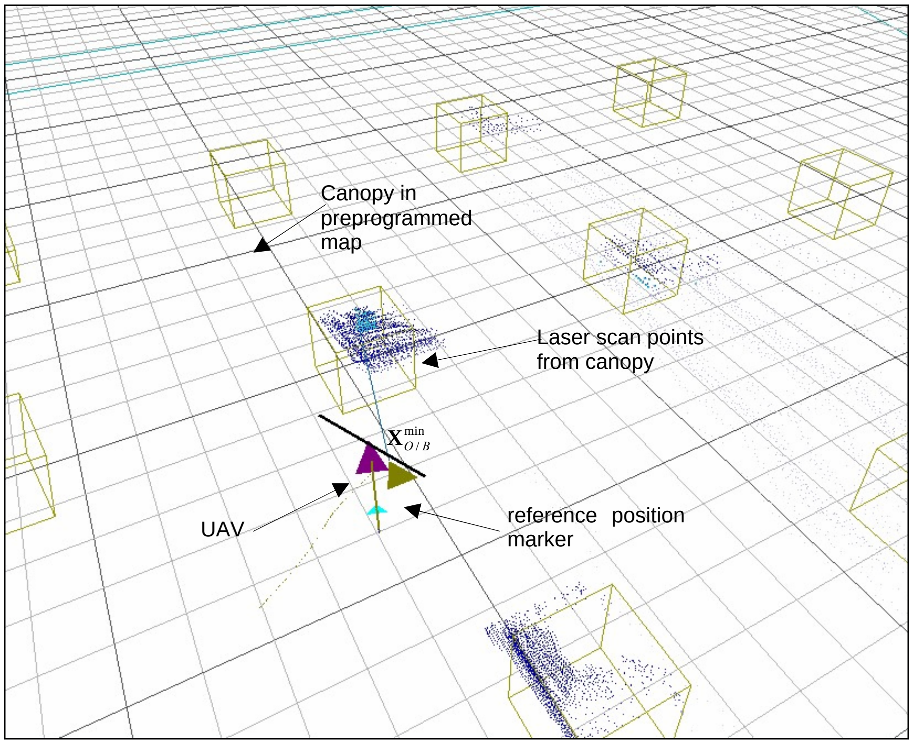
Canopy in preprogrammed map
Laser scan points from canopy
$\mathbf{X}_{O/B}^{\min}$
UAV
reference position marker

Figure 8. A snapshot of three-dimensional rendering during an urban exploration experiment

dots. During the experiments, the laser scanner used in our experiment demonstrated its capability to detect the canopies within the line of sight with great accuracy, and other surrounding natural and artificial objects including buildings, trees and power lines.

The processed laser scanned data in a form of local obstacle map is used to generate a trajectory using the algorithm in Section III-A. The trajectory is then sent via IEEE 802.11b to the avionics system with a dedicated process running to enforce the command update rate at 10Hz. The tracking control layer enables the host vehicle to follow the given trajectory with sufficient distance. In the repeated experiments, the vehicle was able to fly around the obstacles with sufficient distance to reach the destination as shown in Figure 6 (red line).

V. Conclusion

This paper presented an autonomous exploration method for unmanned aerial vehicles in unknown urban environments. An onboard laser scanner is used to build an online map of obstacles around the vehicle. This local map is combined with a real-time MPC algorithm that generates a safe vehicle path, which uses a cost function that penalizes the distance to the nearest obstacle. The adjusted trajectory is then sent to a position tracking layer in the hierarchical UAV avionics architecture. In a series of experiments using a Berkeley UAV, the proposed approach successfully guided the vehicle safely through the urban canyon.

VI. Acknowledgment

This research was supported by DARPA (F33615-98-C-3614) and ARO MURI (DAAD190210383).

References

[1]S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.

[2]M. C. Martin and H. P. Moravec, Robot Evidence Grids, CMU Robotics Institute Technical Report, CMU-RI-TR- 96-06, Mar. 1996.

[3]G. J. Sutton and R. R. Bitmead, “Computational Implementation of NMPC to Nonlinear Submarine,” in F. Allgöwer and A. Zheng, editors, Nonlinear Model Predictive Control, volume 26, pages 461-471, Birkhäuser, 2000.

[4]D. H. Shim, H. J. Kim, and S. Sastry, “Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots,” IEEE Conference on Decision and Control, Maui, HI, December 2003.

[5]J. Sprinkle, J. M. Eklund, H. J. Kim and S. Sastry, “Encoding Aerial Pursuit/Evasion Games with Fixed Wing Aircraft into a Nonlinear Model Predictive Tracking Controller,” pages 2609-2614, *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December, 2004

[6]D. H. Shim, *Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles,* Ph. D. thesis, University of California, Berkeley, 2000.