Shortest Paths Through 3-Dimensional Cluttered Environments

Jun Lu[1], Yancy Diaz-Mercado[2], Magnus Egerstedt[2], Haomin Zhou[1] & Shui-Nee Chow[1]

***Abstract*— This paper investigates the problem of finding shortest paths through 3-dimensional cluttered environments. In particular, an algorithm is presented that determines the shortest path between two points in an environment with obstacles which can be implemented on robots with capabilities of detecting obstacles in the environment. As knowledge of the environment is increasing while the vehicle moves around, the algorithm provides not only the global minimizer – or shortest path – with increasing probability as time goes by, but also provides a series of local minimizers. The feasibility of the algorithm is demonstrated on a quadrotor robot flying in an environment with obstacles.**

I. Introduction

Path planning is one of the enabling technologies making it possible for robots to successfully traverse cluttered environments and, as such, it has received considerable attention in the robotics community. (See for example the textbooks [1], [2], [3] and the references therein for a comprehensive treatment on robotic path planning.)

Since globally optimal path planning is hard, the computational solutions that have been proposed typically fall into three different camps, namely (i) grid-based planners, such as A* [4] and D* [5], [6], [7] , (ii) sample-based planners, such as RRTs [8] or (iii) continuous deformations of locally optimal plans, e.g., [9], [10], [11]. This does not mean that global planners have not been developed, as was done for example in [11], [12], [13], but what it means is that it is in general quite hard to transition global planners from an off-line setting to an on-line robotics implementation.

In this paper we focus on the problem of making an aerial robot traverse an environment populated by obstacles following the shortest path. Optimal planning for aerial vehicles maneuvering environments with obstacles where multiple waypoints must be achieved has been studied with respect to the vehicle dynamics and other tactical constraints [14], [15], however this often results in local minimizers or it is constraint to two dimensions. In addition, many existing planning algorithms either focus on finding a feasible path (collision-free) [10] or optimizing a different objective function rather than the length of the path [11], [16]. In fact, finding the globally shortest path has been proven to be NP-hard in 3-D with polyhedral obstacles in the framework known as configuration space by Canny and Rief [17]. Approximation methods, which are the only practical approaches, seek paths whose length is $1+\epsilon$ times the actual minimal path [18], [19], [20], [21]. However, those methods are either only theoretical and only works for polyhedral obstacles, or have high complexity which are unsuitable for on-line implementation. In this paper, we will take advantage of the recently developed method in [12], where intermittent diffusion techniques are employed to get increasingly good (short) paths in arbitrary dimensions. The resulting algorithm has the following advantages over existing methods: (i) it is able to find the globally shortest path with probability $1-\delta$ with complexity $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$ for arbitrarily small δ, to the best of our knowledge, this is the best complexity occurred in the literature; (ii) the main computation in the algorithm lies in solving initial value ODE's or stochastic ODE's, which is very easy to implement; (iii) the algorithm can be adapted to handle dynamic environments, in which some obstacles may appear or disappear. These three features will allow us to have an aerial quadrotor negotiate a cluttered environment effectively and the main contribution of the paper should be understood in terms of solving the 3-D shortest path problem in an implementable, on-line manner, that is actually deployed on a real robotic vehicle.

This work was supported by NSF Faculty Early Career Development (CAREER) Award DMS-0645266 and DMS-1042998, and ONR Award N000141310408

[1] School of Mathematics, Georgia Institute of Technology, 686 Cherry Street, Atlanta, GA 30332-0160 USA jlu39@math.gatech.edu, hmzhou@math.gatech.edu, chow@math.gatech.edu

[2] Department of Electrical & Computer Engineering, Georgia Institute of Technology, 777 Atlantic Drive NW, Atlanta, GA 30332-0250, USA yancy.diaz@gatech.edu, magnus@gatech.edu

II. Planning Approach

A. Problem Statement

Consider N obstacles in $\mathbf{R}^3$, each of which is represented by a connected and compact set $P_k \subset \mathbf{R}^3$ $(1 \leq k \leq N)$. We assume that all of the obstacles are pairwise disjoint and their boundaries have certain regularities, for example being piecewise C^2. For convenience of computation, we will represent each obstacle through the signed distance function $\phi_k(x)$,

$$\phi_k(x) = \begin{cases} -p_k(x), & x \in P_k \\ p_k(x), & x \in P_k^c \end{cases} \tag{1}$$

where P_k^c denotes the complement of P_k and $p_k(x)$ is the Euclidean distance from x to ∂P_k (the boundary of P_k),

$$p_k(x) = \min_{y \in \partial P_k} \|x - y\|.$$

The signed distance function can be computed efficiently through level set methods [22].

For any two points $x, y \in \partial P_k$, let $d_k(x, y)$ be the length of the shortest geodesic (shortest path) connecting x and y that remains entirely on the boundary. For special shapes of obstacles such as cubes, cylinders and cones, $d_k(x, y)$ can be computed analytically. For polyhedral obstacles, $d_k(x, y)$

can be computed efficiently by the algorithms in [23], or see Remark 2.2 for more information. The general, more complicated cases can be handled by fast marching [24] or fast sweeping [25], or approximated by polyhedrons, with a trade off between accuracy and complexity. In this paper, the planner will assume that $d_k(x, y)$ is given a priori. The environment, however, is allowed to be adaptive, in the sense that in the experimental implementation of the planning algorithm, the environmental knowledge will be incrementally updated.

The path we are interested in is a curve γ in $\mathbf{R}^3$, which is a continuous map $\gamma\colon [0,1] \to \mathbf{R}^3$. We let $L(\gamma)$ be the length of γ under the Euclidean metric.

A path with finite length is said to be rectifiable, and we say a path is feasible, in a geometric sense, if γ does not intersect with the interior of any obstacles. Formally, we require

$$\phi_k(\gamma(t)) \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{2}$$

As the quadrotors have a spatial footprint, we need to be able to accommodate this situation as well. The problem of how to consider the geometry of the robot is out of the scope of this paper and has been addressed before [1]. However, since the spatial footprint of the quadrotor is quite symmetrical, at least along certain dimensions, the robot's footprint can be approximated by a ball of radius r, such that the robot can be treated as a ball robot. This is exemplified in Fig. 8. The level set representation of obstacles in (1) enables us to deal with ball robots in a straightforward manner. One only needs to enlarge all obstacles by this radius r, which boils down to a uniform decrease in the level set function. We may restate the definition of feasibility to: a path is said to be feasible for a ball robot with radius r if

$$\phi_k(\gamma(t)) - r \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{3}$$

It is noteworthy that this approach of wrapping the robot inside a ball can be done even when the robot's footprint is not symmetrical. Even though it is possible to produce better approximations for a given robot footprint, we use this approximation for convenience.

Let F be the set of all feasible rectifiable paths for a ball robot with radius r which starts from X and ends at Y ($\gamma(0) = X, \gamma(1) = Y$). Our problem is to find a path in F such that its length is minimized:

$$\gamma_{opt} = \operatorname*{argmin}_{\gamma \in F} L(\gamma). \tag{4}$$

B. Intermittent Diffusion Algorithm

As observed in [12], [26], the shortest path possesses a special structure that can be harnessed for the purpose of path planning: the optimal path γ_{opt} consists only of line segments and geodesics on the boundaries of obstacles. Therefore we can represent γ_{opt} by a sequence of junctions (special points) that connect different line segments or obstacle boundary

$$(x_0, x_1, \ldots, x_n, x_{n+1}), \quad x_0 = X, \quad x_{n+1} = Y. \tag{5}$$

Each x_i connects to its neighbors x_{i-1} and x_{i+1} either by a line segment or a curve on the boundary. Let x_i^c be the adjacent junction that connects to x_i by a curve and x_i^s the junction that connects to x_i by line segment ($x_0^c = x_0$, $x_{n+1}^c = x_{n+1}$), as shown in Fig. 1.

This structure enables us to search for the optimal path through a sequence of junctions $(x_0, x_1, \ldots, x_n, x_{n+1})$. Note that the number of junctions n may vary in the algorithm. Each x_i (except x_0 and x_n) is the ending point of exactly one line segment and one geodesic on the boundary. The sum of their length is given by

$$J(x_i) = \|x_i - x_i^s\| + d_{n_i}(x_i, x_i^c), \tag{6}$$

where n_i is the index such that $x_i \in P_{n_i}$. The length of γ is then

$$L(\gamma) = L(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=0}^{n+1} J(x_i). \tag{7}$$

The factor $\frac{1}{2}$ comes from the fact that each line segment or geodesic is counted exactly twice.

The above formulation converts the original infinite dimensional problem of finding a path into the a finite dimensional problem of finding the optimal junctions. Therefore, we can apply gradient descent to find the minimizer. One thing to notice is that the number of junctions may vary following the gradient flow. This will be taken care of by our algorithm automatically, in essence by inserting the intersection points when a line segment intersects an obstacle or removing the common junction when two straight segments share a common junction.

To overcome the challenge that gradient descent is only able to find the local minimizer, we adopt a recently developed global optimization strategy called intermittent diffusion. The key idea of intermittent diffusion is that when the gradient flow gets stuck at a local minimizer, we add certain amount of noise to the flow which will kick the flow out of the local trap. The perturbed gradient flow becomes the following stochastic differential equation

$$dx_i = -\nabla J(x_i)dt + \sigma(t)\mathbf{T}(x_i, x_i^c)dW(t), \tag{8}$$

where $W(t)$ is the standard Brownian motion and $\sigma(t)$ is a step function representing the magnitude of the noise we add. Here

$$\mathbf{T}(x_i, x_i^c) = -\nabla d_{n_i}(x_i, x_i^c), \tag{9}$$

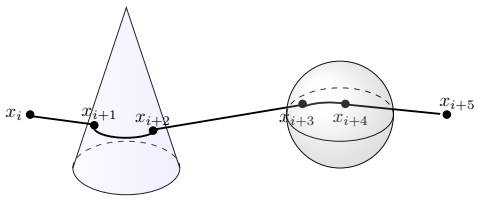

Fig. 1. $x_{i+1}^s = x_i, x_{i+1}^c = x_{i+2}, x_{i+2}^s = x_{i+3}, x_{i+2}^c = x_{i+1}$.

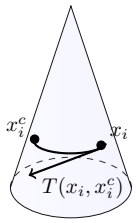

Fig. 2. An illustration of $\mathbf{T}(x_i, x_i^c)$.

i.e. $\mathbf{T}(x_i, x_i^c)$ is the tangent direction at x_i on the curve belonging to the shortest path connecting x_i and x_i^c. The tangent on this curve exists, even when x_i is a critical point of ∂P_{n_i}, for example, at the tip of the cylinder in Fig. 1, as long as $x_i \neq x_i^c$.

We also have that $\sigma(t)$ is given by

$$\sigma(t) = \sum_{i=1}^{m} \sigma_i 1_{[S_i,T_i]}(t). \tag{10}$$

Here $t \in [0,T]$ and $1_{[S_i,T_i]}(t)$ is the indicator function of the interval $[S_i,T_i]$, i.e. one if $t \in [S_i,T_i]$, zero otherwise. The parameters $0 = S_1 < T_1 < S_2 < T_2 < \cdots < S_n < T_n < S_{n+1} = T$ and σ_i are randomly generated by the algorithm and they represent the length of the nonzero intervals and the magnitude of the noise to be inserted at the time interval $[S_i,T_i]$. Given a desired minimum probability of obtaining the global optimizer, the parameters m and T are then selected in order to comply with this probability.

The idea behind the choice of σ is that when $\sigma(t) = 0$, x_i converges to a local minimizer following the negative gradient flow and jumps out of a local trap when $\sigma(t) > 0$ with certain probability. Moreover, we will be able to obtain not only the global optimizer but also a series of local minimizers. This is useful when limited time is allowed for the algorithm to run.

The following theorem captures how the intermittent diffusion algorithm works:

Theorem 1: Given any real number $\delta > 0$, there exist $\tau > 0$, $\sigma_0 > 0$ and integer $m_0 > 0$ such that if $T_i - S_i > \tau$, $\sigma_i < \sigma_0$ (for $i = 1, 2, \cdots, m$), then equation (8) converges to the global minimizer of equation (7) with probability at least $1 - \delta$.

This theorem is a direct application of intermittent diffusion and for a detailed description, including details on how to select τ, σ_0 and m_0, we refer readers to [27].

Now, one challenge associated with the theorem is that it is not particularly user-friendly. It deals with a stochastic differential equation which needs to be solved. However, this can be alleviated by performing some manipulation to obtain a difference equation, which we can then used to obtain a solution numerically. In fact, equation (8) can be discretized by many well established schemes. In this paper, we use the Euler scheme in which the noise term $dW(t)$ is discretized in time as

$$dW(t) = \sqrt{\Delta t}\xi, \tag{11}$$

where $\xi \sim N(0,1)$ is a standard, normal random variable and Δt is the step size. By introducing (6) into (8) we have that

$$dx_i = -\frac{x_i - x_i^s}{\|x_i - x_i^s\|}dt + (\sigma(t)dW(t) + dt)\mathbf{T}(x_i, x_i^c). \tag{12}$$

Combining all the terms together, the temporal discretization is

$$\frac{x_i^{k+1} - x_i^k}{\Delta t} = -\frac{x_i^k - (x_i^k)^s}{\|x_i^k - (x_i^k)^s\|} + (\frac{\sigma(k\Delta t)\xi}{\sqrt{\Delta t}} + 1)\mathbf{T}(x_i^k, (x_i^k)^c). \tag{13}$$

When solving (13), we need to deal with appearing and disappearing junctions. New junctions are generated when a straight line segment of the path intersects with an obstacle in the flow (8). The proposed algorithm simply inserts the intersections points into the sequence of junctions, as in Fig. 3. On the other hand, when two straight components $\overline{x_i z}, \overline{x_j z}$ share a common junction z, the path can be shortened by connecting $x_i x_j$ directly, as long as this straight line does not intersect another obstacle. We hence remove z from the set of junctions. If $\overline{x_i x_j}$ does intersect another obstacle, then new junctions must be added as previously described. See Fig. 4.

Remark 2.1 (Dynamic Environments): The approach described above can also be used when the environment is not fixed but changes over time. When obstacles appear, it is only necessary to introduce the new junctions as described above if an obstacle instersects with the current path. Likewise, when obstacles disappear, the path can be shortened as described above.

Remark 2.2 (Polyhedral Obstacles): For polyhedral obstacles, the proposed method can be equally applied to computing the geodesics on the surface. More specifically, for any two points $x, y \in P_k$, there exists a sequence of junctions

$$(z_0, z_1, \cdots, z_l), \quad z_0 = x, \quad z_l = y, \tag{14}$$

such that each z_i is on some edge of P_k and

$$d_k(x,y) = \sum_{i=0}^{l-1} \|z_i - z_{i+1}\|. \tag{15}$$

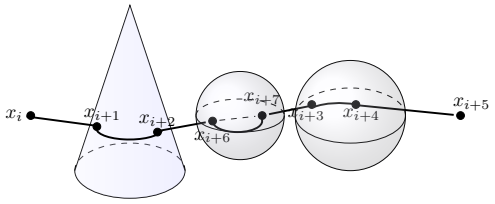

Fig. 3. The initial path is $(\cdots, x_i, x_{i+1}, x_{i+2}, x_{i+3}, \cdots)$. As the path changes, $\overline{x_{i+2}x_{i+3}}$ intersects with the obstacle in between. We add the intersections points as junctions and the new path becomes $(\cdots, x_i, x_{i+1}, x_{i+2}, x_{i+6}, x_{i+7}, x_{i+3}, \cdots)$.

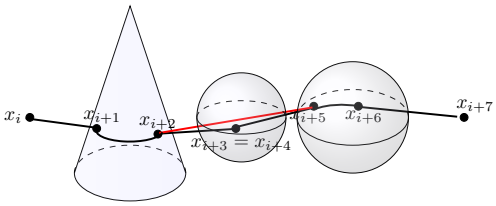

Fig. 4. The initial path is $(\cdots, x_i, x_{i+1}, x_{i+2}, x_{i+3}, x_{i+4}, x_{i+5}, \cdots)$. At some time during the path change, $x_{i+3} = x_{i+4}$. The path can be shortened by connecting x_{i+2} and x_{i+5}. The new path becomes $(\cdots, x_i, x_{i+1}, x_{i+2}, x_{i+5}, x_{i+6}, \cdots)$.

To determine the z_i's, it suffices to optimize (15) over all possible (14). For more details, especially how to handle the topological changes of the path during evolution, we refer readers to [28].

III. Robotic implementation

In this section an algorithm is proposed to find the set of junctions that represents the shortest path between two points in an environment with obstacles. The feasibility, in a dynamical sense, of the algorithm is then demonstrated when it is implemented on a Parrot AR.Drone quadrotor robot, seen on Fig. 11 and Fig. 12.

A. Algorithm

We present our algorithm below.

Input: level set function $\phi_k(x)$
the distance function $d_k(x, y)$,
starting and ending points X and Y,
number of intermittent diffusion intervals m.
Output: The optimal set U_{opt} of junctions.

1 **Initialization.** Find the initial set U of junction points.
2 Select duration of diffusion $\Delta T_l, l \leq m$
3 Select diffusion coefficients $\sigma_l, l \leq m$.
4 **for** $l = 1 : m$
5 $U_l = U$;
6 **for** $x_i \in U_l$
7 **for** $j = 1 : \Delta T_l$
8 Update x_i according to (13)
9 with $\sigma(t) = \sigma_i$
10 Update set U_l, i.e. remove junctions
11 from or add junctions to U_l;
12 **end**
13 **while** $\|x_i^{k+1} - x_i^k\| > \epsilon$
14 Update x according to (13)
15 with $\sigma(t) = 0$
16 Update set U_l;
17 **end**
18 **end**
19 **end**
20 Compare U_l's and set $U_{opt} = \underset{l \leq m}{\operatorname{argmin}}\ L(U_l)$.

The initial set U of junctions consists all the intersections points of line segment $\overline{XY}$ and the obstacles. This initialization gives initial path similar to those generated by bug algorithms [10], and in many cases is already close enough to the global optimizer. It should be noted that good initialization is not required for the proposed algorithm. Given enough time to run the algorithm, the global minimizer will still be obtained even starting from a far initial path.

Both the duration of diffusion ΔT_l and diffusion coefficients σ_l are randomly selected in intervals $[0, T_{\max}]$ and $[0, \sigma_{\max}]$ respectively. Experiments show that $T_{\max} = 20$ and $\sigma_{\max} = 2$ are often adequate. Depending on whether one wants to record local minimizers, line 20 can be replaced by keeping track only of the best minimizers at current realization. This will save storage dramatically.

We now give a brief analysis of the algorithm

1) Completeness: Since we assume all the obstacles are bounded and disjoint, and we start from a feasible path, Theorem 1 guarantees the proposed algorithm is complete.

2) Complexity: Following [20], instead of discussing the algebraic complexity of the algorithm, we will consider the running time in order to achieve certain relative error ϵ. We will compare our result with other approaches only for polyhedral obstacles since most of the literature focus on them.

1) The initialization can be done by a bisection line search, which can be achieved in $O(\log \frac{1}{\epsilon})$ time..
2) Line 2-3 takes $O(m)$ time.
3) Inner loop line 7-12 takes $O(\Delta T_l)$ time. This is because equation (13) takes constant time, and so does adding or removing junctions.
4) Inner loop line 13-17 takes $T(\epsilon)$ time where $T(\epsilon)$ denotes the number of iterations required until the error is less than ϵ. If we assume the Hessian matrix of the gradient is nondegenerate, which is the case for all polyhedral obstacles [18], then $T(\epsilon) = O(\log \frac{1}{\epsilon})$.

Let $\Delta T = \max_{i \leq l} \Delta T_i$. Then the total running time is $O(m(\Delta T + \log \frac{1}{\epsilon}))$. From [27], it can be shown that in order to obtain the desired successful probability $1 - \delta$, the number of realizations must be of order $O(\log \frac{1}{\delta})$. Therefore, the complexity is $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$. Table I shows a complexity comparison with some existing methods.

TABLE I
Complexity Comparison to Other Algorithms

Algorithm	Complexity
A^*	$O((\frac{1}{\epsilon})^3 \log \frac{1}{\epsilon})$
Papadimitriou [20]	$O(\frac{1}{\epsilon})$
Choi, et. al. [18] (When the shortest path is not unique.)	$O(\frac{1}{\epsilon})$
Choi, et. al. [18] (When the shortest path is unique.)	$O(\log \frac{1}{\epsilon})$

TABLE II

IMPROVEMENT ON PROBABILITY OF OBTAINING SHORTEST PATH, ENVIRONMENT A.

Minimizer	Length (m)	Times Obtained Out of		
		100	200	300
1	5.8660	48	103	159
2	5.9527	50	91	128
3	6.0403	1	4	6
4	6.0594	0	1	1
5	6.0919	0	0	1
6	6.2286	0	0	3
7	6.2305	1	1	2

B. Algorithm Implementation

The algorithm described above was implemented for several environment definitions, as seen in Fig. 5 for environment A, and Fig. 6 and Fig. 7 for environment B. Fig. 5 presents several of the local minimizers found by the algorithm for environment A. Table II shows how, out of seven local minimizers found in environment A, the number of times the shortest path was found improves as more time is allowed for the algorithm to run, i.e. when more realizations (outer-most loop of the algortihm) are permitted. The sorting effect is also evident where the second and third shortest minimizers also improve in probability of being obtained by the algorithm. Fig. 6 and Fig. 7 are minimizers found for environment B. In this case, the number of minimizers is less than in environment A, however the total distance for each minimizer is much closer together. The algorithm is still able to discern between the local and global minimizer, as can be seen in table III. It is noteworthy that even though these paths are very close to each other in total length, their geometry is very different. Depending on the scope of the application, one may choose to follow a local minimizer rather than the global minimizer, trading off distance for alternate path geometry.

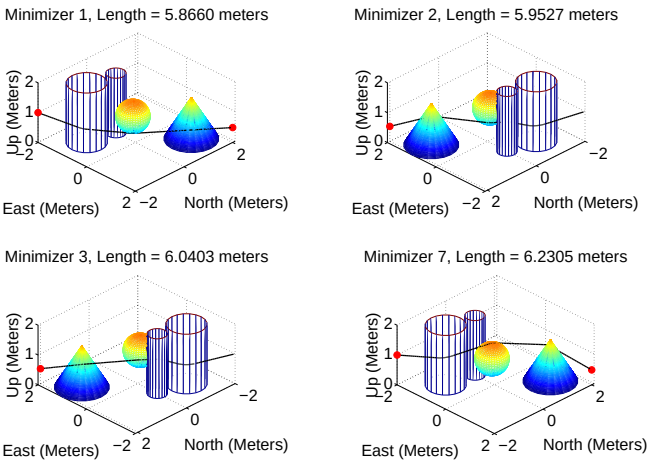

Fig. 5. Three shortest and longest minimizers for environment A

TABLE III

IMPROVEMENT ON PROBABILITY OF OBTAINING SHORTEST PATH, ENVIRONMENT B.

Minimizer	Length (m)	Times Obtained Out of		
		100	200	300
1	5.9204	52	104	158
2	5.9279	48	96	142

C. Environment Definition

Environment B is considered for the robotic implementation and experimental setup that will be the focus of the next sections. As seen in Fig. 6 and Fig. 7, three obstacles are considered, one cone with base center $(0, 0.8, 0)$, base radius $0.8m$ and height $0.8\sqrt{3}m$; one cylinder with base center $(-1, -1, 0)$, radius $0.6m$ and height $2m$; one cylinder with base center $(0.5, -0.5, 0)$, radius $0.5m$ and height $2m$. The starting and ending points are located at $(-2, -2, 1)$ and $(2, 2, 0.1)$ respectively. The proposed algorithm is able to find two minimizers as shown in Fig. 6 and Fig. 7. Fig. 7 is the globally shortest path whose length is approximately $5.9204m$. This is in contrast to the distance between the initial and final position which is approximately $5.728m$ and the local minimizer length which is approximately $5.9279m$, with a difference of only $0.0075m$ between the local and global minimizer.

D. Experimental Setup

For the environment described above the algorithm is run on a quadrotor robot as seen in Fig. 12. The environment was translated by $0.5m$ in the south direction to have the origin coincide with the state coordinate frame. As described in (3), the actual obstacles radii were originally increased in order to accommodate for the spatial footprint of the robot r. The footprint was increased further to r' to compensate for modeling errors, as can be seen in Fig. 8. Six-dimensional state information, including the robot's position in $\mathbf{R}^3$, its pitch, yaw and roll orientations, is available through the use of an Optitrack motion capturing system. This system includes set of 10 Optitrack S250e motion capture cameras and a central computer with tracking software, and is capable

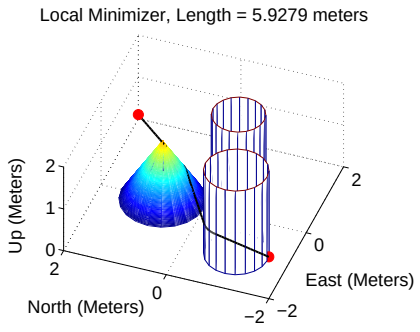

Fig. 6. Local minimizer path with length of approximately $5.9279m$ for Environment B.

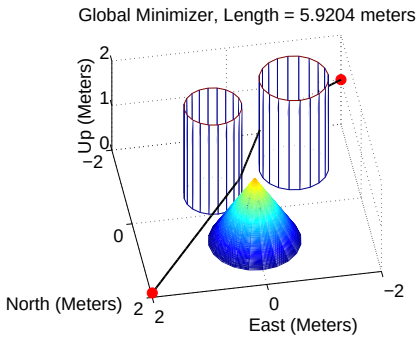

Fig. 7. Global minimizer path with length of approximately $5.9204\ m$ for Environment B.

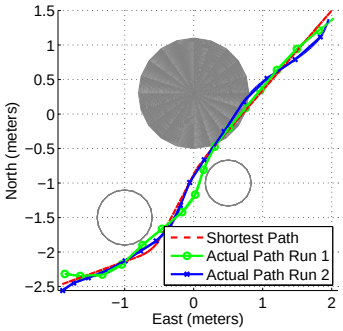

Fig. 9. Shortest Reference Path and Actual Robot Paths, top view. The clear circles are the cylinder whereas the darker circle is the cone.

of providing state information at a rate of up to $250Hz$ with millimeter accuracy.

Optitrack data is sent via TCP/IP over a 10/100 Ethernet switch to a central control computer, a Dell Optiplex 745 with an Intel dual core $2.13GHz$ CPU and 4 GB RAM running Ubuntu 11.04 Natty Narwhal, where every computation takes place. The Robot Operating System (ROS), Diamondback version, is used as the framework to: compute the shortest path with the algorithm, to compute the control signal and to send this control signal to the Parrot AR.Drone quadrotor robot via a wireless router. The Parrot AR.Drone quadrotor, which is equipped with an on-board ARM GNU/Linux board running at $600MHz$, front- and downwards facing VGA cameras, a controller for a stabilized hover and is capable of achieving 12-15 minutes of flight from its $1000mAh$ battery, receives the control command wirelessly and executes it.

As the produced path is purely geometric in that it does not take into account the dynamics of the vehicle, we assume that the quadrotor can follow the planned path sufficiently well. In other words, single integrator dynamics are assumed for the robot's model, i.e. $\dot{x} = u$. This means that we have direct control of the linear and angular velocities where $x \in \mathbf{R}^6$ corresponds to the state information available from the Optitrack system and $u \in \mathbf{R}^6$ is the control signal to be computed.

The control signal is computed as follows: the shortest path is first found for an environment with obstacles where previous knowledge of obstacle sizes, shapes and locations is assumed. The path is then discretized into a collection of equally spaced points. A PID controller is used to minimize the difference between a point in the set and the robot's position on the XY-plane whereas a proportional controller is used for height control.

Once the robot is within a predefined small distance from the point in the path a new point is selected and the new difference is then minimized until the last point in the set is reached. The actual path for multiple experiment runs of the robot are shown in Fig. 9 and Fig. 10. Fig. 9 shows the top view of the actual path followed by the robot whereas Fig. 10 shows an angled view of the data.

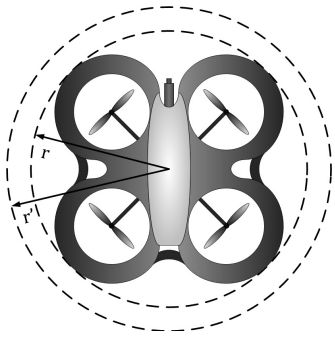

Fig. 8. A quadrotor robot taken as a ball robot of radius r. The robot may be taken to be a ball robot of radius r' instead to compensate for modeling errors.

IV. Conclusion

The problem of having a robot traverse a 3-dimensional environment with obstacles was considered. An algorithm was presented that utilizes intermittent diffusion techniques

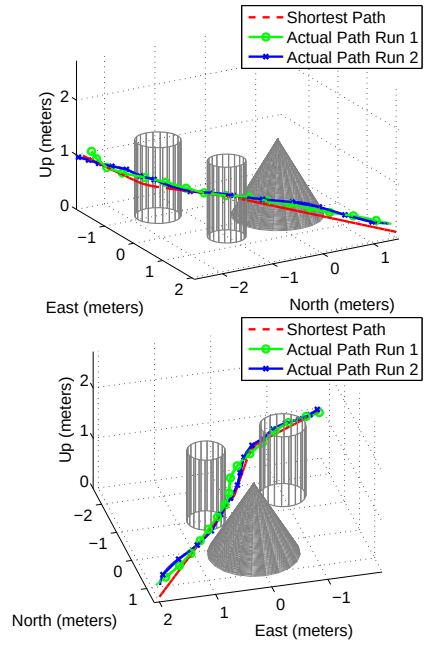

Fig. 10. Shortest Reference Path and Actual Robot Path, angled views.

Fig. 11. A third of the way: front, profile and back view.

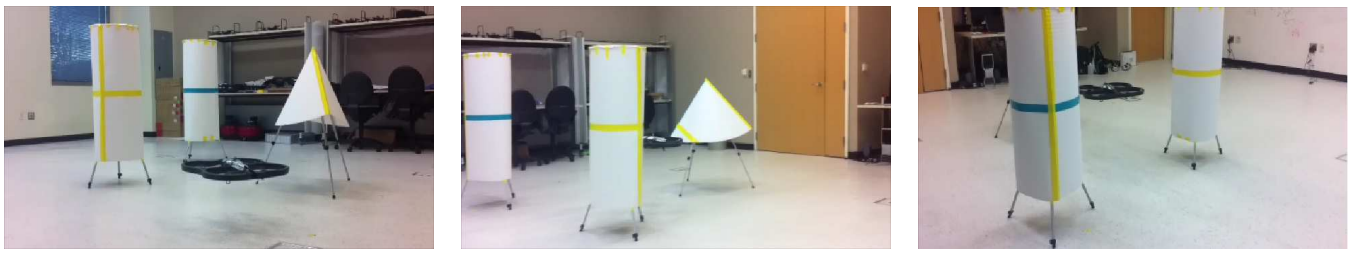

Fig. 12. Two thirds of the way: front, profile and back view.

to provide the global distance minimizer, or shortest path, with increasing probability as more time is allowed for the algorithm to run. A series of local minimizers are also provided for situations where more time is not possible. A brief completeness and complexity analysis is provided for the algorithm together with a comparison to other existing algorithms. Furthermore, the presented algorithm is implementable in an on-line manner for robots with the capability of detecting obstacles and approximating complex shapes to composition of simpler shapes. The algorithm is validated when it is successfully implemented in a quadrotor robot.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.

[2] J.-C. Latombe, *Robot motion planning*. Springer, 1990.

[3] F. Fahimi, *Autonomous robots: modeling, path planning, and control*. Springer, 2008.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[5] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 3310–3317.

[6] A. Stentz *et al.*, "The focussed D* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, vol. 14. Lawrence Erlbaum Associates LTD, 1995, pp. 1652–1659.

[7] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 354–363, 2005.

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

[9] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 802–807.

[10] V. J. Lumelsky, "Algorithmic and complexity issues of robot motion in an uncertain environment," *Journal of Complexity*, vol. 3, no. 2, pp. 146–182, 1987.

[11] J. F. Canny and M. C. Lin, "An opportunistic global path planner," in *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, 1990, pp. 1554–1559.

[12] S.-N. Chow, J. Lu, and H.-M. Zhou, "Finding the shortest path by evolving junctions on obstacle boundaries (E-JOB): An initial value ODE's approach," *Applied and Computational Harmonic Analysis*, 2012.

[13] J. T. Schwartz, *Planning, geometry, and complexity of robot motion*. Ablex Pub, 1987.

[14] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3, 2002, pp. 1936–1941 vol.3.

[15] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, vol. 2, 2002, pp. 1301–1306 vol.2.

[16] J. S. B. Mitchell and M. Sharir, "New results on shortest paths in three dimensions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 124–133.

[17] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *28th Annual Symposium on Foundations of Computer Science*. IEEE, 1987, pp. 49–60.

[18] J. Choi, J. Sellen, and C. Yap, "Precision-sensitive Euclidean shortest path in 3-space," in *Proceedings of the eleventh annual symposium on Computational geometry*. ACM, 1995, pp. 350–359.

[19] J. Choi, J. Sellen, and C.-K. Yap, "Approximate Euclidean shortest paths in 3-space," *International Journal of Computational Geometry & Applications*, vol. 7, no. 04, pp. 271–295, 1997.

[20] C. Papadimitriou, "An algorithm for shortest-path motion in three dimensions," *Information Processing Letters*, vol. 20, no. 5, pp. 259–263, 1985.

[21] K. Jiang, L. Seneviratne, and S. Earles, "Finding the 3D shortest path with visibility graph and minimum potential energy," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1993, pp. 679–684.

[22] S. Osher and R. Fedkiw, *Level set methods and dynamic implicit surfaces*. Springer, 2003, vol. 153.

[23] Y. Schreiber, "Euclidean shortest paths on polyhedra in three dimensions," Ph.D. dissertation, Tel Aviv University, 2007.

[24] R. Kimmel and J. A. Sethian, "Computing geodesic paths on manifolds," *Proceedings of the National Academy of Sciences*, vol. 95, no. 15, p. 8431, 1998.

[25] H. Zhao, "A fast sweeping method for eikonal equations," *Mathematics of computation*, vol. 74, no. 250, pp. 603–628, 2005.

[26] J. Hopcroft and G. Wilfong, "Motion of objects in contact," *The International journal of robotics research*, vol. 4, no. 4, pp. 32–46, 1986.

[27] S.-N. Chow, T.-S. Yang, and H. Zhou, "Global Optimizations by Intermittent Diffusion," *Chaos, CNN, Memristors and Beyond*, 2013.

[28] S.-N. Chow, J. Lu, and H. Zhou, "Shortest path amid 3-D polyhedral obstacles," *submitted to SIAM multiscale modeling and simulation*, 2013.