2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Motion Planning with Satisfiability Modulo Theories

William N. N. Hung[1], Xiaoyu Song[2], Jindong Tan[3], Xiaojuan Li[4], Jie Zhang[5], Rui Wang[4] and Peng Gao[2]

***Abstract*— Motion planning is an important problem with many applications in robotics. In this paper, we focus on motion planning with rectangular obstacles parallel to the X, Y or Z axis. We formulate motion planning using Satisfiability Modulo Theories (SMT) and use SMT solvers to find a feasible path from the source to the goal. Our formulation decompose the robotic path into N path segments where the two ends of each path segment can be constrained using difference logic. Our SMT approach will find a solution if and only if a feasible path exists for the given constraints. We present extensive experimental results to demonstrate the scalability of our approach.**

I. Introduction

Motion planning [1], [2] is an important problem with many applications in robotics. From the iRobot vacuum cleaner to the Mars Exploration Rovers, motion planning is a key ingredient to robots in the real world. In this paper, we investigate the motion planning problem for robots with rectangular obstacles parallel to the X, Y, or Z axis. Irregular shaped obstacles can be approximated using one or more rectangular obstacles to cover their space. The basic problem is to produce a feasible path that connects a start configuration S and a goal configuration G. Our formulation breaks down the robot path into several connected path segments. The two ends of each path segment are constrained using difference logic. The entire path planning problem is formulated using satisfiability modulo theories (SMT) [3]. Our formulation is applicable to two dimensional and three dimensional space, and it can handle grid-based or gridless paths. If there is a feasible path from start S to goal G, our SMT approach will be able to find that path. If there is no feasible path, our SMT approach will also be able to reach such a conclusion.

A SMT problem is a decision problem on a set of logical formulas. Each formula is a combination of various background theories expressed in classical first-order logic with equality. The problem is to find an assignment to the variables that satisfy the given set of logical formulas. The typical theories used are the theory of integers, the theory of real numbers, and the theories of many data structures such as lists and arrays. Difference logic can be considered as a subtheory under the theory of integers or the theory of real numbers. Many SMT solvers use SAT solvers for their core logic reasoning. Over the past decade, both SAT solvers [4], [5], [6] and SMT solvers [7], [8], [9] have significantly improved their runtime performance and capacity and they have been applied to large and complex problems [10], [11], [3].

Many sampling-based algorithms [1], [2] have been used for motion planning. These algorithms typically relax strong completeness guarantees to optimize for speed. Timed Automata was applied to motion planning in [12]. Linear Temporal Logic (LTL) was used for motion planning in [13], [14], [15], [16]. Although they provide powerful expressive capabilities, both Timed Automata and LTL suffer from the state-explosion problem. On the other hand, SMT solvers are a lot more scalable than temporal logic approaches. In this paper, we conduct a wide range of experiments to demonstrate the scalability of our SMT based approach.

II. Background

Our core formulation can be expressed in terms of difference logic. There are typically two variants of difference logic: DLZ (difference logic over integers) and DLR (difference logic over reals). In this section, we summarize the problem definition presented in [17], [18].

Definition 1 (Difference Logic): Let $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ be a set of Boolean variables and $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ be a set of numerical variables. The set of atomic formulas of $DL(\mathcal{B}, \mathcal{X})$ consists of the Boolean variables in $\mathcal{B}$ and properties on the numerical variables in $\mathcal{X}$ in the following forms:

$$x_i - x_j \geq c \quad \text{and} \quad x_i - x_j > c$$

where $c \in \mathbb{Z}$ for DLZ and $c \in \mathbb{R}$ for DLR). The set $\mathcal{F}$ of all DL formulas is the smallest set containing the atomic formula which is closed under negation and conjuction:

- $\phi \in \mathcal{F}$ implies $\neg\phi \in \mathcal{F}$
- $\phi \in \mathcal{F}$ and $\varphi \in \mathcal{F}$ implies $\phi \wedge \varphi \in \mathcal{F}$

Other Boolean operators $\vee, \wedge, \rightarrow, \ldots$ can be defined in terms of negation and conjuntion.

A valuation of $(\mathcal{B}, \mathcal{X})$ consists of two functions (overloaded with the name v) $v : \mathcal{B} \rightarrow \{\mathrm{T}, \mathrm{F}\}$ and $v : \mathcal{X} \rightarrow \mathbb{Z}$ for DLZ or $v : \mathcal{X} \rightarrow \mathbb{R}$ for DLR. The valuation v is extended to all $DL(\mathcal{B}, \mathcal{X})$ formulas by letting

$$v(x_i - x_j \leq c) = \mathrm{T} \quad \text{iff} \quad v(x_i) - v(x_j) \leq c$$

[1]William N. N. Hung is with Synopsys Inc., Mountain View, California, USA william_hung@alumni.utexas.net

[2]Xiaoyu Song and Peng Gao are with Portland State University, Portland, Oregon, USA

[3]Jindong Tan is with the University of Tennessee, Knoxville, Tennessee, USA

[4]Xiaojuan Li and Rui Wang are with the Beijing Engineering Research Center of High Reliable Embedded System, College of Information Engineering, Capital Normal University, Beijing 100048, China

[5]Jie Zhang is with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China. She is now a visiting scholar at the University of Tennessee, Knoxville, Tennessee, USA

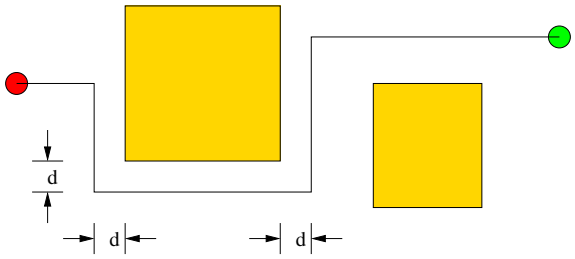

Fig. 1. Rectilinear path with distance d from obstacles

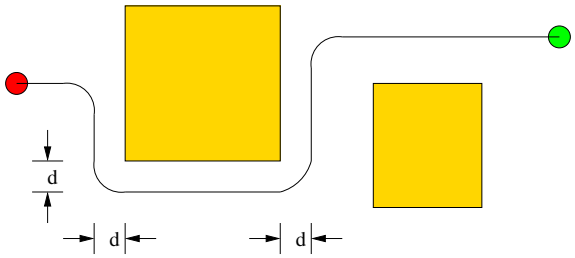

Fig. 2. Smooth path with turns of radius d

and applying similar rules for the Boolean operators. The difference logic over other domains can be defined in similar fashion. A formula ϕ is satisfied by a valuation v iff $v(\phi) = \text{T}$. The satisfiability problem for $DL(\mathcal{B}, \mathcal{X})$ is NP-complete [17].

For the rest of this paper, we use $:=$ to denote a definition and $=$ to denote equality.

III. Theory

In this section, we formulate an motion planning problem as a set of constraints, then map it onto a difference logic satisfiability problem, and finally add a path length constraint to restrict the travel distance for the robot path.

We are given a three dimensional space $W \times L \times H$. The robot needs to move from the source (s^x, s^y, s^z) to the destination (d^x, d^y, d^z). There are M rectangular obstacles in the three dimensional space. Each obstacle j can be defined using their three dimensional span $[x_L^j, x_H^j], [y_L^j, y_H^j], [z_L^j, z_H^j]$, where $x_L^j \leq x_H^j$, $y_L^j \leq y_H^j$, $z_L^j \leq z_H^j$. Since the obstacles are all rectangular in nature, the robot can move in rectilinear directions, i.e. parallel to the X, Y, or Z axis. We require the robot paths to maintain a minimum separation distance d from all obstacles. Since all paths are rectilinear in nature, the robot turns are all 90 degree turns. These 90 degree turns can be smoothed out with curves of radius d. By controlling the size of d, we can prevent sharp turns for the robot. An example rectilinear path is shown in Figure 1. The corresponding smooth path is shown in Figure 2.

A. N-Segment 3D Path Planning

Suppose we want to solve a path planning problem on a three dimensional space $W \times L \times H$, we can divide the path into N path segments: $p_1, p_2, \ldots, p_N$. Each path segment p_i is a straight line from a source vertex (x_i^s, y_i^s, z_i^s) to a destination vertex (x_i^d, y_i^d, z_i^d). The (x, y, z) coordinates of each source and destination must be within the region for $i = 1, \ldots, N$:

$$\begin{aligned} 0 \leq \quad x_i^s \quad &\leq W \\ 0 \leq \quad y_i^s \quad &\leq L \\ 0 \leq \quad z_i^s \quad &\leq H \\ 0 \leq \quad x_i^d \quad &\leq W \\ 0 \leq \quad y_i^d \quad &\leq L \\ 0 \leq \quad z_i^d \quad &\leq H \end{aligned} \tag{1}$$

We define three boolean expressions ξ_i^x, ξ_i^y and ξ_i^z such that for $i = 1, \ldots, N$:

$$\begin{aligned} \xi_i^x \quad &:= \quad (x_i^s = x_i^d) \\ \xi_i^y \quad &:= \quad (y_i^s = y_i^d) \\ \xi_i^z \quad &:= \quad (z_i^s = z_i^d) \end{aligned}$$

Since each path segment p_i go along the X, Y or Z direction, we have the following property for $i = 1, \ldots, N$:

$$(\xi_i^x \wedge \xi_i^y) \vee (\xi_i^y \wedge \xi_i^z) \vee (\xi_i^x \wedge \xi_i^z) \tag{2}$$

Since we have only one source and one goal, the source of each path segment i is the same as the destination of its preceding path segment n_{i-1}. Hence for $i = 2, \ldots, N$, we have:

$$\begin{aligned} x_i^s \quad &= \quad x_{i-1}^d \\ y_i^s \quad &= \quad y_{i-1}^d \\ z_i^s \quad &= \quad z_{i-1}^d \end{aligned} \tag{3}$$

The robot path should be connected to the source and destination:

$$\begin{aligned} x_1^s &= s^x \\ y_1^s &= s^y \\ z_1^s &= s^z \\ x_N^d &= d^x \\ y_N^d &= d^y \\ z_N^d &= d^z \end{aligned} \tag{4}$$

Each path segment i must be separated from the obstacle j by a minimum separation distance d for $i = 1, \ldots, N$ and $j = 1, \ldots, M$:

$$\begin{aligned} &(x_L^j - x_i^s \geq d \wedge x_L^j - x_i^d \geq d) \vee \\ &(x_i^s - x_H^j \geq d \wedge x_i^d - x_H^j \geq d) \vee \\ &(y_L^j - y_i^s \geq d \wedge y_L^j - y_i^d \geq d) \vee \\ &(y_i^s - y_H^j \geq d \wedge y_i^d - y_H^j \geq d) \vee \\ &(z_L^j - z_i^s \geq d \wedge z_L^j - z_i^d \geq d) \vee \\ &(z_i^s - z_H^j \geq d \wedge z_i^d - z_H^j \geq d) \end{aligned} \tag{5}$$

The path planning problem is essentially the conjunction of formulas (1)–(5). with the source and destination variables for each path segment. Our formulation is applicable to both discrete and continuous path planning. We can use SMT

solvers to find solutions to these formulas. If there is no feasible path, SMT solvers can also confirm there is no solution.

B. N-Segment Plannar Path Planning

The single layer path planning or plannar path planning is a special case of the 3D path planning problem where $z = 0$ for all coordinates. All the formulas in the previous subsection can be greatly simplified for this problem. Formula (2) becomes:

$$\xi^x_{n_i} \vee \xi^y_{n_i}$$

The rest of the formulas: (1), (3), (4) and (5) can also be trivially simplified.

C. Mapping to Difference Logic

Given an N-segment path planning problem, we can declare integer variables for the (X,Y,Z) coordinates of the source and destination vertices of each path segment i:

$$x^s_i, y^s_i, z^s_i, x^d_i, y^d_i, z^d_i$$

The constraints in (1)–(5) are essentially formulas using numeric relation operators $\leq$, $\geq$ and $=$ on the variables (and constants) as well as Boolean operators $\vee$, $\wedge$ to connect the results of the above relation opertors. Since $a = b$ is equivalent to $\neg((a < b) \vee (a > b))$, all of the above formulas can be easily expressed in terms of difference logic.

D. Path Length Constraints

One consideration for path planning is to minimize or limit the length of the robot path. In general, this can become an optimization problem. However, we can set a limit L to the length of the robot path:

$$\sum_{i=1}^{N} |x^d_i - x^s_i| + |y^d_i - y^s_i| + |z^d_i - z^s_i| \leq L \qquad (6)$$

Notice formula (6) is not a difference logic formuation, but it can still be solved by SMT solvers.

IV. Experiments

In this section, we present experiments on random benchmarks to evaluate our motion planner. Each random benchmark is generated using the following scheme: We are given three dimensional space $W \times L \times H$ and the number of obstacles M. The source (s^x, s^y, s^z) and destination (d^x, d^y, d^z) are chosen randomly in the three dimensional space. For each obstacle j, we randomly generate its three dimensional span $[x^j_L, x^j_H], [y^j_L, y^j_H], [z^j_L, z^j_H]$, where $x^j_L \leq x^j_H$, $y^j_L \leq y^j_H$, $z^j_L \leq z^j_H$. for $j = 1, \ldots, M$, such that the obstacle lies within the three dimensional space $W \times L \times H$. We use the motion planning described in Section III with the Z3 [7] SMT solver. All experiments are conducted on Linux with an Intel i7 processor and 2GB memory.

Table I shows experiments on a variety of random three dimensional benchmarks. Each row in the table shows a different size $W \times L \times H$ or a different number of obstacles M. The number of path segments N for each experiment is set to $4 \times M$. The minimum separation distance between the robot motion path and the obstables is $d = 1$. For each row in the table, we generate 100 different benchmarks using different random seeds for the same $W \times L \times H$ and M. Such large number of different random seeds will enable the result to be easily reproduced. Each one of those 100 random benchmarks is either reachable (from source to destination) or not reachable. The "count" columns under "Reachable" (or "Unreachable") shows how many of those benchmarks can (or cannot) be routed from source to destination. The "time" columns shows the avarage time for the Z3 SMT solver on these benchmarks. Given the same number of obstacles, such as $M = 8$, Table I shows that our approach takes roughly the same amount of time (on average) to solve the random benchmarks even if the size of the three dimensional space increase from $10 \times 10 \times 10$ to $640 \times 640 \times 640$.

To understand how many path segments N are needed for motion planning, we took the randomly generated benchmarks from 3 rows of Table I: $80 \times 80 \times 80$ with $M = 32$,

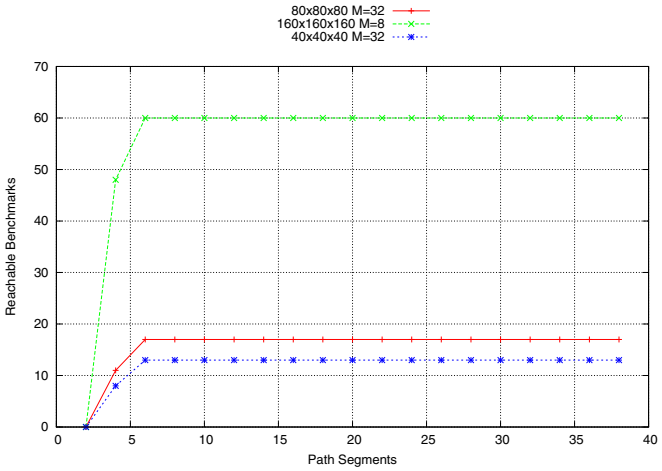

Fig. 3. Number of reachable benchmarks vs. number of path segments N

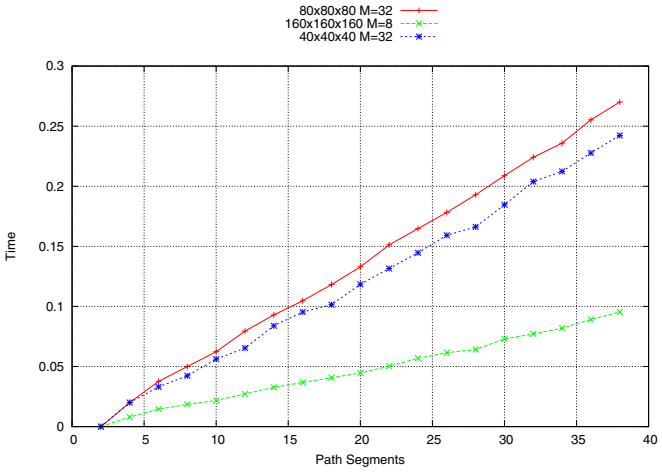

Fig. 4. Average time for reachable benchmarks vs. N

TABLE I

RANDOM 3D BENCHMARKS WITH $d = 1$

$W \times L \times H$	M	N	Reachable		Unreachable		Total	
			count	time	count	time	count	time
$10 \times 10 \times 10$	2	6	70	0.008	30	0.000	100	0.005
$10 \times 10 \times 10$	4	8	59	0.017	41	0.000	100	0.010
$10 \times 10 \times 10$	8	10	34	0.023	66	0.006	100	0.012
$20 \times 20 \times 20$	2	6	75	0.004	25	0.000	100	0.003
$20 \times 20 \times 20$	4	8	66	0.012	34	0.000	100	0.008
$20 \times 20 \times 20$	8	10	44	0.020	56	0.008	100	0.013
$20 \times 20 \times 20$	16	10	30	0.030	70	0.016	100	0.020
$40 \times 40 \times 40$	2	6	80	0.005	20	0.000	100	0.004
$40 \times 40 \times 40$	4	8	68	0.014	32	0.000	100	0.009
$40 \times 40 \times 40$	8	10	52	0.021	48	0.009	100	0.015
$40 \times 40 \times 40$	16	10	33	0.035	67	0.019	100	0.024
$40 \times 40 \times 40$	32	12	13	0.065	87	0.028	100	0.033
$80 \times 80 \times 80$	2	6	90	0.005	10	0.000	100	0.005
$80 \times 80 \times 80$	4	8	68	0.012	32	0.000	100	0.008
$80 \times 80 \times 80$	8	10	48	0.021	52	0.008	100	0.014
$80 \times 80 \times 80$	16	10	30	0.038	70	0.019	100	0.025
$80 \times 80 \times 80$	32	12	17	0.079	83	0.036	100	0.043
$80 \times 80 \times 80$	64	14	7	0.157	93	0.064	100	0.070
$160 \times 160 \times 160$	2	6	86	0.004	14	0.000	100	0.004
$160 \times 160 \times 160$	4	8	77	0.013	23	0.000	100	0.010
$160 \times 160 \times 160$	8	10	60	0.022	40	0.010	100	0.017
$160 \times 160 \times 160$	16	10	39	0.038	61	0.020	100	0.027
$160 \times 160 \times 160$	32	12	16	0.084	84	0.039	100	0.046
$160 \times 160 \times 160$	64	14	8	0.175	92	0.076	100	0.084
$160 \times 160 \times 160$	128	14	3	0.313	97	0.128	100	0.134
$320 \times 320 \times 320$	2	6	85	0.005	15	0.000	100	0.004
$320 \times 320 \times 320$	4	8	69	0.013	31	0.000	100	0.009
$320 \times 320 \times 320$	8	10	60	0.023	40	0.010	100	0.018
$320 \times 320 \times 320$	16	10	35	0.040	65	0.020	100	0.027
$320 \times 320 \times 320$	32	12	18	0.084	82	0.039	100	0.047
$320 \times 320 \times 320$	64	14	9	0.193	91	0.081	100	0.091
$320 \times 320 \times 320$	128	14	3	0.347	97	0.147	100	0.153
$320 \times 320 \times 320$	256	16	3	0.743	97	0.315	100	0.328
$640 \times 640 \times 640$	2	6	87	0.005	13	0.000	100	0.004
$640 \times 640 \times 640$	4	8	70	0.013	30	0.000	100	0.009
$640 \times 640 \times 640$	8	10	56	0.022	44	0.011	100	0.017
$640 \times 640 \times 640$	16	10	37	0.039	63	0.019	100	0.027
$640 \times 640 \times 640$	32	12	21	0.085	79	0.039	100	0.049
$640 \times 640 \times 640$	64	14	13	0.193	87	0.084	100	0.098
$640 \times 640 \times 640$	128	14	5	0.388	95	0.161	100	0.173
$640 \times 640 \times 640$	256	16	1	0.870	99	0.357	100	0.363
$640 \times 640 \times 640$	512	18	0	0.000	100	0.743	100	0.743

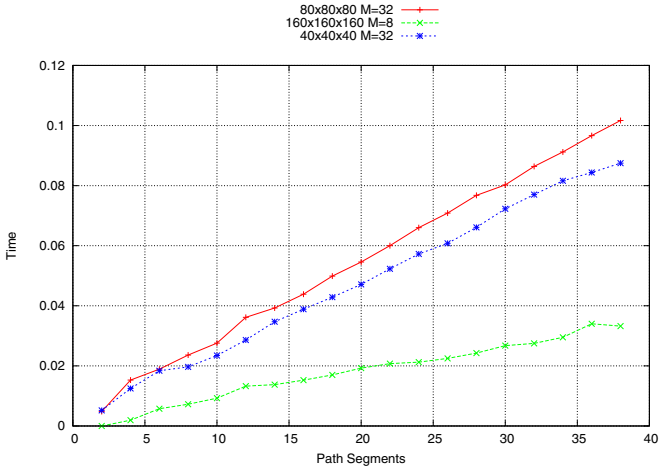

Fig. 5. Average time for unreachable benchmarks vs. N

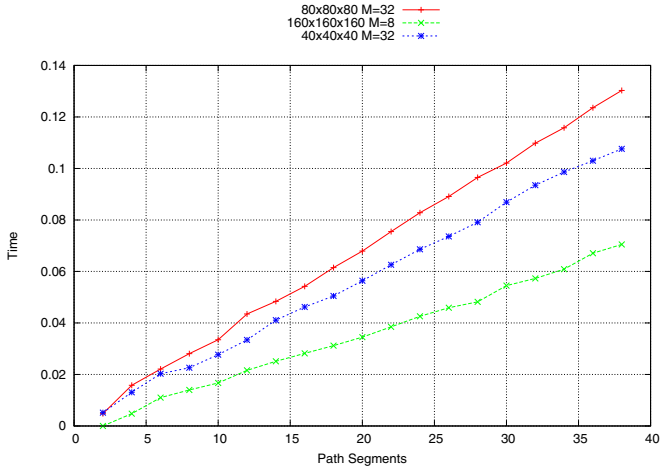

Fig. 6. Average time for all benchmarks vs. N

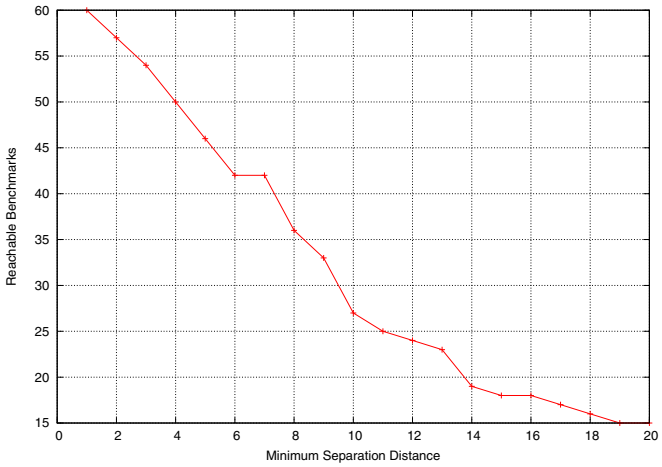

Fig. 7. Number of reachable benchmarks vs. minimum separation distance d

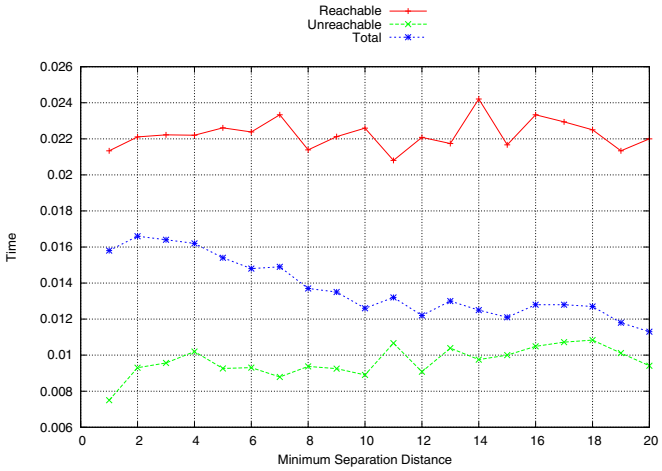

Fig. 8. Average time for benchmarks vs. minimum separation distance d

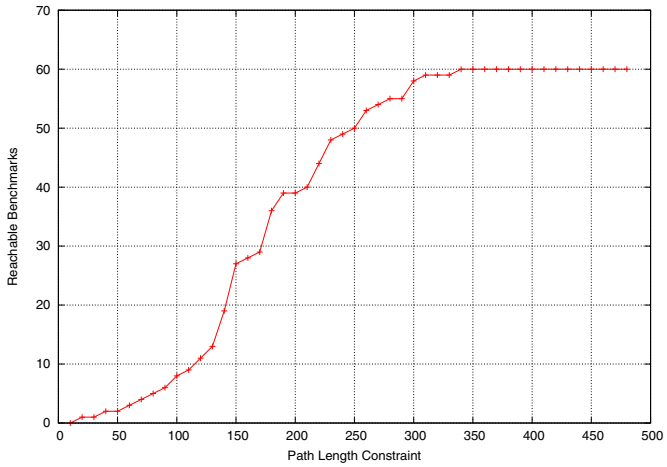

Fig. 9. Number of reachable benchmarks vs. path length L

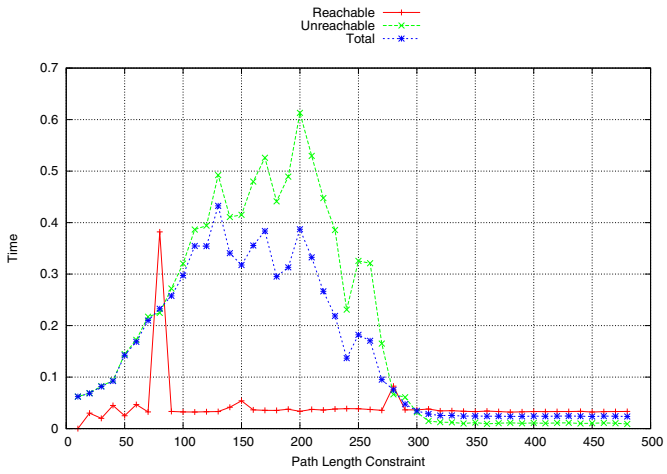

Fig. 10. Average time for benchmarks vs. path length L

$160 \times 160 \times 160$ with $M = 8$, and $40 \times 40 \times 40$ with $M = 32$. There are 100 randomly generated benchmarks from each of those three rows. We change the number of path segments N from 2 to 20. The number of benchmarks that are reachable (from source to destination) is shown in Figure 3. The average time to solve these benchmarks using Z3 is shown in Figure 4 for reachable benchmarks, Figure 5 for unreachable benchmarks, and Figure 6 for all benchmarks, As we increase the number of path segments N, we can see that the number of reachable benchmarks quickly saturates, but the average time for Z3 continues to increase. Note that average time for reachable benchmarks tend to be larger than unreachable benchmarks.

To evaluate the impact of the minimum separation distance d between motion path and the obstacles, we took the 100 randomly generated benchmarks from Table I where the three dimensional space is $160 \times 160 \times 160$, the number of obstacles $M = 8$, the number of path segments $N = 10$, and we change d from 1 to 20. The number of benchmarks that are reachable (from source to destination) is shown in Figure 7. It is obvious that when the minimum separation distance d increases, the number of reachable benchmarks decreases. The average time to solve these benchmarks using Z3 is shown in Figure 8. Note that average time for reachable benchmarks tend to be larger than unreachable benchmarks. Figure 8 suggests that increasing minimum separation distance d does not cause the average time to increase. In fact, the average total time dropped slightly as d increased.

The above experiments were conducted without path lengh constraints. To evalute the impact of a path length constraint, we took the 100 randomly generated benchmarks from Table I where the three dimensional space is $160 \times 160 \times 160$, the number of obstacles $M = 8$, the number of path segments $N = 10$, and we add a path length restriction L from 10 to 480. The number of benchmarks that are reachable (from source to destination) is shown in Figure 9. It is obvious that when the path length L increases, the number of reachable benchmarks increases. The average time to solve

these benchmarks using Z3 is shown in Figure 10.

V. Conclusions

In this paper, we focus on motion planning for robots with rectangular obstacles parallel to the X, Y or Z axis. We formulate motion planning as a SMT problem and use SMT solvers to find a feasible path from the source S configuration to the goal G configuration. Our formulation decompose the robotic path into N path segments along the X, Y or Z direction. The two ends of each path segment can be constrained using difference logic. If a feasible path exists for the given constraints, our SMT approach will find the solution. Otherwise, the SMT solvers will prove there is no feasible solution. Experimental results show that our approach is scalable to handle large problems.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005.

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

[3] L. De Moura and N. Bjørner, "Satisfiability modulo theories: introduction and applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, September 2011.

[4] N. Een, A. Mishchenko, and N. Amla, "A single-instance incremental SAT formulation of proof- and counterexample-based abstraction," in *Proc. Formal Methods in Computer-Aided Design*, ser. FMCAD '10, 2010, pp. 181–188.

[5] L. Yin, F. He, W. N. N. Hung, X. Song, and M. Gu, "Maxterm covering for satisfiability," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 420–426, March 2012.

[6] A. Biere, "Lingeling, Plingeling and Treengeling Entering the SAT Competition 2013," in *Proc. SAT Competition 2013: Solver and Benchmark Descriptions*, ser. Department of Computer Sciences Series of Publications B, A. Balint, A. Belov, M. Heule, and M. Järvisalo, Eds., vol. B-2013-1. University of Helsinki, 2013, pp. 51–52.

[7] L. De Moura and N. Bjørner, "Z3: an efficient SMT solver," in *Proc. International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.

[8] A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani, "The MathSAT5 SMT Solver," in *Proceedings of TACAS*, ser. LNCS, N. Piterman and S. Smolka, Eds., vol. 7795. Springer, 2013.

[9] J. Christ, J. Hoenicke, and A. Nutz, "SMTInterpol: An Interpolating SMT Solver," in *Model Checking Software: Proc. SPIN*, ser. Lecture Notes in Computer Science, A. Donaldson and D. Parker, Eds. Berlin, Heidelberg: Springer, 2012, vol. 7385, pp. 248–254.

[10] C. Flanagan, K. R. M. Leino, M. Lillibridge, G. Nelson, J. B. Saxe, and R. Stata, "Extended static checking for java," in *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation*, ser. PLDI '02, 2002, pp. 234–245.

[11] P. Godefroid, P. de Halleux, A. V. Nori, S. K. Rajamani, W. Schulte, N. Tillmann, and M. Y. Levin, "Automating software testing using program analysis," *IEEE Software*, vol. 25, no. 5, pp. 30–37, Sep. 2008.

[12] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot planning: a timed automata approach," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, ser. ICRA, 2004, pp. 4417–4422.

[13] G. E. Fainekos, H. Kress-gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, Apr. 2005.

[14] ——, "Hybrid controllers for path planning: A temporal logic approach," in *Proc. IEEE Conference on Decision and Control*, Dec. 2005, pp. 4885–4890.

[15] A. Bhatia, M. R. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 55–64, Sep. 2011.

[16] H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control: Mitigating the state explosion problem of temporal logic synthesis," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 65–74, Sep. 2011.

[17] M. Mahfoudh, P. Niebert, E. Asarin, and O. Maler, "A satisfiability checker for difference logic," in *5th International Symposium on the Theory and Applications of Satisfiability Testing*, 2002.

[18] S. Cotton, E. Asarin, O. Maler, and P. Niebert, "Some progress in satisfiability checking for difference logic," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, vol. 3253. Springer, 2004, pp. 263–276.