第 54 回自動制御連合講演会
2011 年 11 月 19 日，20 日 豊橋技術科学大学

到達時間場を利用したランダム探索に基づく移動ロボットのオンライン経路計画

Igi Ardiyanto ○三浦 純 （豊橋技術科学大学）

On-line Path Planning for Mobile Robots using an Arrival Time Field-Biased Randomized Search

I. Ardiyanto and ∗J. Miura (Toyohashi University of Technology)

Abstract− This paper presents a new path planning algorithm for mobile robot in dynamic environments. We calculate the *arrival time field* as a bias which gives larger weights for shorter and safer path towards a goal. We apply a randomzed path search guided by the arrival time field for constructing the path considering kinodynamic constraints. We also consider path quality by adding heuristic constraints such as directing the initial heading of the robot and reducing unstable movements of the robot by using a heading criterion. The algorithm has been tested in simulation and experiments using an actual robot to show its effectiveness.

Key Words: Path planning, Mobile robot, RRT, Arrival time field

1 はじめに

自律移動ロボットにおいて経路計画は基本的な機能のうちの一つである．その満たすべき性能として，目的地に至る安全かつ効率的な経路を生成できること，静的障害物，動的障害物の両方を扱えること，オンライン実行可能であることなどが挙げられる．

これまでに多くの経路計画手法が提案されているが，その中でも RRT（Rapidly-exploring Random Tree）[1] のようにランダム化アルゴリズムに基づく方法は，複雑な経路計画問題に有効であることが示されている．基本的な RRT はその性質上空間を幅広く探索するので，探索をより限定した領域に誘導するための方法がいくつか考えられている [2, 3, 4]．また，幾何学的・動力学的拘束（kinodynamic constraints，以降，KD 拘束と記す）を考慮した方法も提案されている [5, 7]．一方，ランダム化アルゴリズムを用いない方法では，レベルセット法（Level Set Method, LSM）を用いて大域的なポテンシャル場を計算し，最適経路を求める方法 [8] などが提案されている．これらの従来手法は，静的な障害物のみを対象としている（例えば，[1, 4, 5, 8]），計算量が多く実ロボットへの適用が困難である（例えば，[7]），あるいは KD 拘束を考慮していない（例えば，[8]）といった問題点がある．

本論文では，到着時間場（*arrival time field*）を用いて，RRT 的ランダム探索を制御することにより，十分によい経路を効率よく生成する，新たな経路計画アルゴリズムを提案する．このアルゴリズムは移動ロボットの KD 拘束を考慮し，また十分に短い時間で経路生成が可能なので，実ロボットのオンライン経路計画に適用可能である．

2 到達時間場

到達時間場は各地点における目的地への予測到達時間を記述したものであり，目的地へ向かう方向に探索木の枝をより伸ばすためのバイアスを与える．到達時間場を計算するために，まずグリッド地図上の自由空間に距離変換を適用し，各セル x_1 の障害物からの距離関数 $d(x_1)$ を以下のように計算する．

$$d(x_1) = \begin{cases} ||x_1 - x_2|| & \text{for } x_1 \in \boldsymbol{F} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

ここで $\boldsymbol{F}$ は自由空間セルの集合，x_2 は x_1 に最も近い障害物セルの位置である．この距離関数を用いて，速度関数を以下のように計算する．

$$f(x_1) \leftarrow n^{d(x_1)} \tag{2}$$

安全速度は障害物からの距離に依存し，その影響度をパラメータ n で制御する．速度関数の逆数は各セル上を移動する時間を示し，ゴールから時間を伝播させることにより到達時間場を計算する．レベルセット法を用いた実装 [8] では，以下の式を用いる．

$$\sqrt{\begin{matrix}\max\left(D_{i,j}^{-x}T, -D_{i,j}^{+x}T, 0\right)^2 \\ +\max\left(D_{i,j}^{-y}T, -D_{i,j}^{+y}T, 0\right)^2\end{matrix}} = \frac{1}{f_{i,j}};\ (i,j) \in \boldsymbol{F} \tag{3}$$

ここで

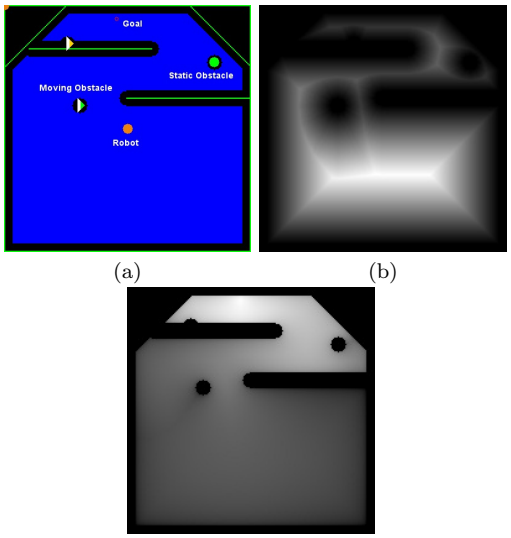
$$\begin{aligned} D_{i,j}^{+x} &= T_{i+1,j} - T_{i,j}, \\ D_{i,j}^{-x} &= T_{i,j} - T_{i-1,j}, \\ D_{i,j}^{+y} &= T_{i,j+1} - T_{i,j}, and \\ D_{i,j}^{-y} &= T_{i,j} - T_{i,j-1}. \end{aligned} \tag{4}$$

であり，$T_{i,j}$，$f_{i,j}$ はそれぞれセル (i,j) の到達時間，速度関数である．

計算された各セルにおける到達時間は正規化し，その後逆数を取ることにより，目的地が最も高い値を持つようにする．これを到達時間場と呼ぶ．この到達時間場の各セルの値を利用して，ランダムサンプリングのバイアスとする．Fig. 1 に到達時間場の計算例を示す．

3 到達時間場をバイアスとするランダム探索アルゴリズム: HeAT Random Tree

任意方向に自由に移動できる点ロボットでは，到達時間場をゴールから最速方向へたどることにより最適経路が計算できる [8]．しかし，KD 拘束を考慮する場合に



(a) (b)

(c)

Fig. 1: (a) Environment map, where blue area denotes free space, green line is wall, the orange circle denotes robot position, the red circle is destination, the triangles are moving obstacles, and the black area is extending space for obstacles. (b) Distance transform of (a). (c) Arrival time field, brighter area means higher value.

は，そのような方法は適用できない．そこで，KD 拘束を考慮したランダム探索アルゴリズムを到達時間場を用いて制御する方法（Heuristically Arrival Time Field-Biased (HeAT) Random Tree 法，以下 HeAT RT 法と呼ぶ）を開発する．

3.1 木の表現

HeAT RT 法では，現在位置から KD 拘束を満たす木を順に成長させ，得られた複数の経路候補からヒューリスティック評価関数により最もよいものを選択する．木の各ノードは 6 つ組 $N = \{x, y, \theta, v, w, t\}$ で定義され，それぞれロボットの位置姿勢 (x, y, θ)，並進・回転速度 (v, w)，および時間 t を表す．

3.2 動作セットと枝の伸長

各動作は並進・回転速度の組で表され，あらかじめ有限個の動作セットを与えておく．ある動作 $u_i = \{v_i, w_i\}$ による，時間 t における状態 N_t から時間 $t+1$ における状態 N_{t+1} への遷移を以下の関数で示すこととし

$$N_{t+1,u_i} \leftarrow g(N_t, u_i), \tag{5}$$

新たな状態 N_{t+1} におけるロボットの位置姿勢は次式で与えられる [6]:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \frac{v_i}{w_i} \begin{pmatrix} -\sin\theta_t + \sin(\theta_t + w_i\Delta t) \\ \cos\theta_t - \cos(\theta_t + w_i\Delta t) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ w_i\Delta t \end{pmatrix} \tag{6}$$

ここで Δt は 1 ステップ分の時間を表す．

各動作は KD 拘束を満たすだけでなく，障害物との衝突を避けなければならず，特に移動障害物との衝突判定が重要である．移動障害物の検出・追跡システム（例えば，[11]）を仮定し，各時点でのすべての移動障害物の位置と速度およびそれらの誤差が得られるものとする．未来の時間においては，そのまま現在の速度で移動し続けるとして衝突判定を行う．

3.3 最良優先探索による経路生成

ロボットと目的地の間に障害物がほとんどないような，簡単な状況では最良優先探索により経路を生成することにより，高速に経路を生成する．探索の各ステップでは状態 N_t から N_{t+1} を以下のように選択する．

$$N_{t+1,u_{best}} \leftarrow g(N_t, u_{best}), \tag{7}$$

ここで最良の動作 u_{best} は N_t で可能なすべての動作を調べて決定する:

$$N_{t+1,u} \leftarrow g(N_t, u), \text{for } u \in \{u_1, u_2, u_3, \ldots, u_{K_u}\} \tag{8}$$

$$u_{best} = \arg\min_u cost(N_{t+1,u}) \tag{9}$$

ここで K_u は KD 拘束を満たす動作の総数．$cost(N_{t+1,u})$ はヒューリスティックなコスト関数であり，以下の値を用いる:

$$\alpha M_1 + \beta M_2 + \delta M_3, \tag{10}$$

$$M_1 = bias(x_{t+1}, y_{t+1}) \tag{11}$$

$$M_2 = dist((x_{goal}, y_{goal}) - (x_{t+1}, y_{t+1})) \tag{12}$$

$$M_3 = |\theta_{t+1} - \theta_t| \tag{13}$$

ここで $bias(x_{t+1}, y_{t+1})$ は動作 u によって到達すると予測される地点 (x_{t+1}, y_{t+1}) の到達時間場の値，$dist((x_{goal}, y_{goal}) - (x_{t+1}, y_{t+1}))$ はその地点と目的地 (x_{goal}, y_{goal}) との距離，$|\theta_{t+1} - \theta_t|$ はその地点でのロボットの向きと目的地でのロボットの向きの差，α，β，δ は重みづけの係数である．

この最良優先探索に基づく経路生成は，それが可能な場合に極めて高速に実行可能解を生成できる．そこで，時間制限を設け，その時間内に経路が生成できない場合には，次に述べるランダム探索に基づく経路生成を行う．

3.4 ランダム探索に基づく経路生成

ランダム探索に基づく経路生成では，空間を広く探索するという利点がある RRT[1] を基本とするアルゴリズムを提案する．従来の RRT 法との違いは，到達時間場を利用することにより，探索をより有効な領域に集中させる点にある．ランダム点の選択領域を制限するために，しきい値を用いる．このしきい値は，最初はロボットの現在位置の到着時間場の値とし，探索途中ではこれまでに生成されたノードのうち最も高い値になるように繰り返し更新する（Fig. 2 参照）．ランダム探索木生成アルゴリズムを Algorithm 1 に示す．

提案するアルゴリズムでは，一定以上の値を持つ到達時間場内の領域でランダムに選択された点に対して，前項の最良優先探索と同様の処理を行って新たなノードを生成する．すなわち，ランダム点を仮の目的地と考え，そこへ至る最良の動作を選択する．

上記の処理を繰り返して探索木が目的地へ到達したら，すべてをリセットして再び初期位置・姿勢から同じ処理を繰り返す．あらかじめ決められた時間（現在 200 $[ms]$）が過ぎるか，あらかじめ決められた数のノード（現在 3,000 個）が生成されたら，繰り返しを終了する．

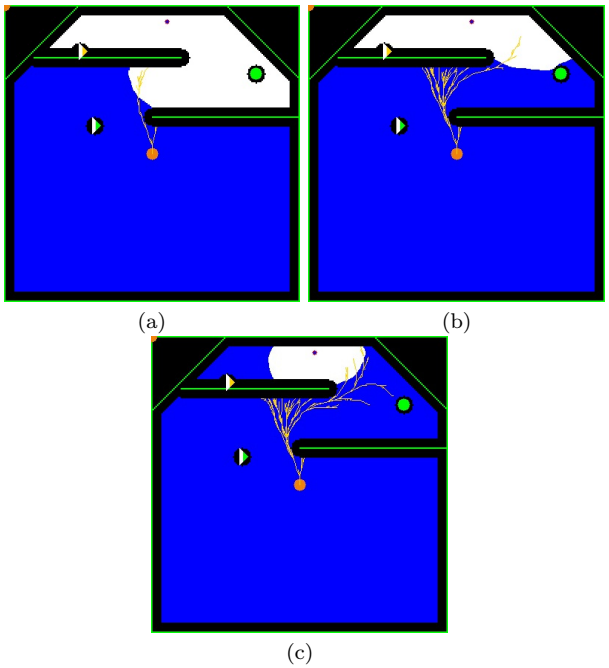


Fig. 2: Expanding the tree using bias (see algorithm (1)). The white area is the region for choosing a random point.

Algorithm 1: HeAT Random Tree

Properties :
N = collection of nodes
u = motion control
$bias(x)$ = arrival time value of point x

Function :
HeAT_RANDOM_TREE_PLANNER()
$N \Leftarrow x_{init}$
$temp\_bias \Leftarrow bias(x_{init})$
while *time_is_available* **do**
  $x_{near} \Leftarrow$ CHOOSE_STATE(x_{rand}, N)
  $N \Leftarrow N \cup$ EXTEND_TREE(x_{near})[1]
  if $bias_(x_{new}) \geq temp\_bias$ **then**
    $temp\_bias \Leftarrow bias(x_{new})$
  end if
end while

Function : CHOOSE_STATE(x_{rand}, N)
while *time_is_available* **do**
  x_{rand} = random point from F
  if $bias_(x_{rand}) \geq temp\_bias$ **then**
    return BEST_NODE(x_{rand}, N)[2]
  end if
end while

[1]make an extension node by evaluating every possible u according to (8).
[2]return the nearest node of N to x_{rand}.

ランダム探索を繰り返し行うことにより得られた複数の経路候補から，もっとも短時間で目的地へ到達できるものを選択し，その最初の動作をロボット制御器へ送ることによりロボットを動作させる．

3.5 経路の再利用

実移動ロボットでは，物体検出，地図更新，経路計画などの処理を速いサイクル（現状では，1 サイクル

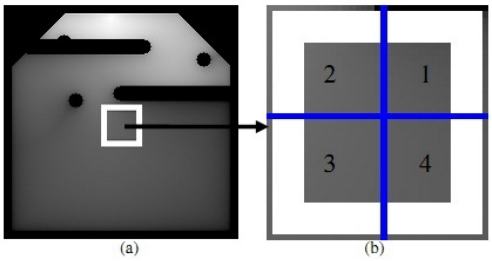
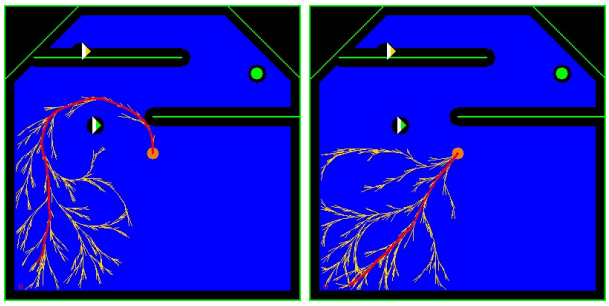


Fig. 3: (a) the arrival time field, (b) a part of (a) centered at the robot position, divided into four region



(a) (b)

Fig. 4: Effect of directing the initial heading of the robot, (a) without and (b) with the initial heading

500 $[ms]$）で行っているので，1 サイクル前の状態と現在の状況は大きく変化しない．そこで，1 サイクル前に選択された経路を現在位置から衝突するかどうかを順に調べてゆき，衝突の生じない部分だけを取り出し，探索木の初期木として利用する．

3.6 ロボットの初期姿勢の修正

KD 拘束があるとロボットは全方向へ移動することができないため，移動開始前にロボットが向いている方向と大きく異なる方向に目的地がある場合，大回りした経路が生成されてしまう．そこで，移動開始前にその場で旋回をして方向を合わせることにする．しかしながら，目的地方向へ完全に合わせることがよいかどうかは環境に依存するため，ここでは到着時間場を利用して，ロボットの方向をよいと思われる方向に大まかに合わせることにする．

まず，ロボットを中心とする狭い範囲を設定し，それを 4 つの領域に分け (Fig. 3 参照)，各分割領域に対し到達時間場の平均値を計算し，それがもっとも大きい領域を向くようにロボットを回転させる．ロボットの背面方向に目的地がある場合に，初期回転の有無による生成される経路の比較を行ったものが Fig. 4 である．初期回転がない場合には遠回りする経路が生成されているが (Fig. 4a)，適切な初期回転によって効率的な経路が生成されている (Fig. 4b)．

4 実験結果

4.1 既存手法との比較

提案手法（HeAT RT 法）と他の RRT に基づく手法との比較を行う．ここでは，基本的な RRT 法 [1] と hRRT 法 [3] を比較対象とする．RRT 法はランダム点を自由空間全体から選択する．一方，hRRT 法は目的地との距離を利用して，より目的地へ向かう方向に探索木を伸展させる．これらの手法は元々動的障害物や KD 拘束は考慮していないが，公平な比較のためランダム木伸展の部分だけを変更して比較を行った．ノード

Table 1: Comparison of Path Planning Algorithms

	Success Rate of 20 times Run (%)	Average of Execution Time of Successful Runs (ms)
Basic RRT	65	75
hRRT	70	95
HeAT RT	95	110

数の最大数は 1000，最大計算時間は 200 $[ms]$ とした．

Table 1 は目的地へ安全に到達した割合と平均実行時間の比較である．HeAT RT 法では到達時間場の計算が必要であるにもかかわらず，計算時間をそれほど増やすことなく，極めてロバストな経路生成が行えていることがわかる．また，Fig. 5 に探索木の様子を比較する．基本 RRT 法は幅広く探索しているが効率的ではなく，hRRT 法は経路コストの予測に障害物形状を陽に考慮していないため効率的な経路が生成できていない．これに対し，HeAT RT 法では到達時間場と経路評価のヒューリスティックを基に望ましい枝の伸展が行われていることが分かる．

4.2 実験結果

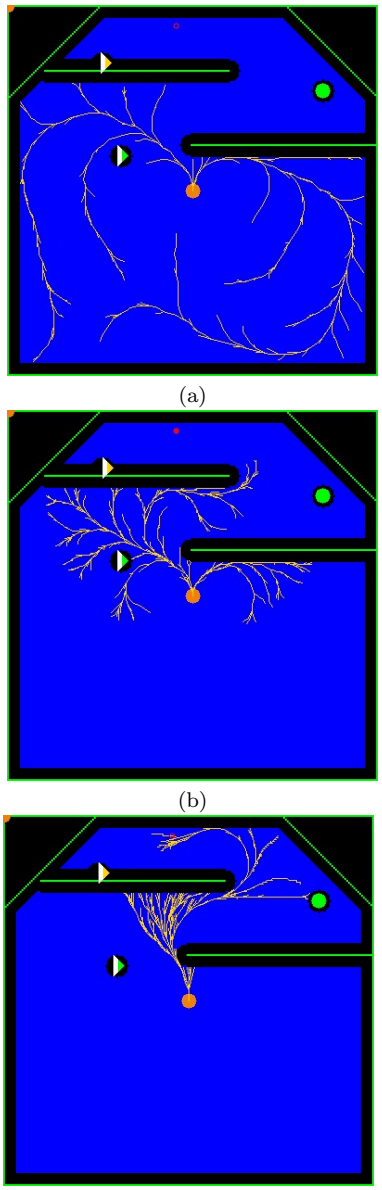
HeAT RT 法を，ロボット用ソフトウェア開発環境 RT ミドルウェア [9] 上で動作する経路計画 RT コンポーネント（RTC）として実装し，シミュレーション実験および実ロボットを用いた実験を行った．経路計画の実行はノート PC（Core2Duo, 2.1GHz, 2BG memory）で行った．

4.2.1 シミュレーション実験結果

多数人物シミュレータ [10] を用いてシミュレーション実験を行った．このシミュレータも RT コンポーネントとして実装されているため，接続が容易である．Fig. 6 にシミュレーションの様子を示す．ここでは，食堂での人の動きを想定し，複数の人物が券売機で整列して食券を買う，料理の載ったトレイを受け取る，空いている椅子に座って食事を取る，食器を下膳口へ持っていく，といった一連の行動がシミュレーションされている．ロボットは一人の人物を対象として追従行動を行っている．Table 2 は，追従対象が食堂に入ってから席に着くまでの間，ロボットが追従するというタスクを 10 回行った結果をまとめたものである．すべての場合で衝突なく追従することができ，HeAT RT 法の計算時間は 500 $[ms]$ 以下であった．これよりオンライン処理が可能であることが分かる．

4.2.2 実ロボット実験結果

富士通ヒューマノイド ENON にステレオカメラ（PointGrey 社 Bumblebee2），レーザ距離センサ（北陽電機 UTM-30LX）を搭載し，実験を行った．開発した経路計画 RTC は経由点リストを与えてそれらを順にたどるモードと，検出された複数人物のうち指示された人物を追従するモードがある．Fig. 7 は経由点リストを与えて行動した場合，Fig. 8 は人物追従を行った場合である．人物追従にはステレオ視に基づく人物発見・追跡手法 [11] を用いた．人物発見・追跡，自由空間地図更新，経路計画をすべて 500 $[ms]$ 以内に行い，最大速度 0.4 $[m/s]$ で安全に移動した．



(a) (b) (c)

Fig. 5: Comparison of tree expansion, (a) Basic RRT, (b) hRRT, and (c) HeAT Random Tree

Table 2: Statistic of Simulation Result

Statistic	Value
Arrival Time Field calculation	40 ms
Best-first Path calculation	10 ms
Random Tree calculation	250 ms
Number of nodes	3000 nodes
Maximum speed	400 mm/second
Number of simulation	10 times
Successful runs	100%

5 おわりに

到達時間場をバイアスとしてランダム経路探索を制御することにより，自由空間形状を考慮して効率的な経路を高速に生成するアルゴリズム（HeAT RT 法）を提案した．さらに，従来手法との比較や実験によってその有効性を確かめた．今後は，より効率的な経路の生成のために，動的障害物を考慮した 3 次元時空間で到達時間場を計算しバイアスとして利用することを考えている．

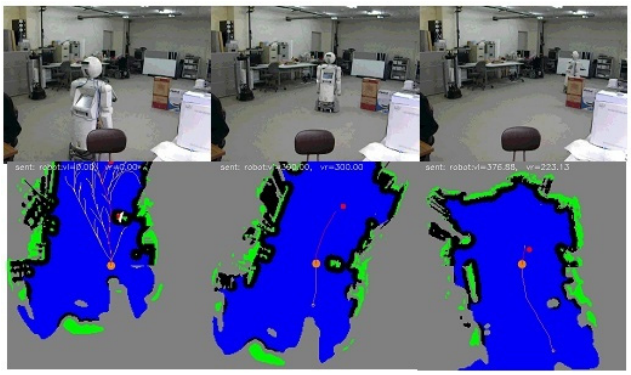


Fig. 7: Experiment of following waypoints, (from left to right) sequences of the real world (top) and local map scene (bottom)

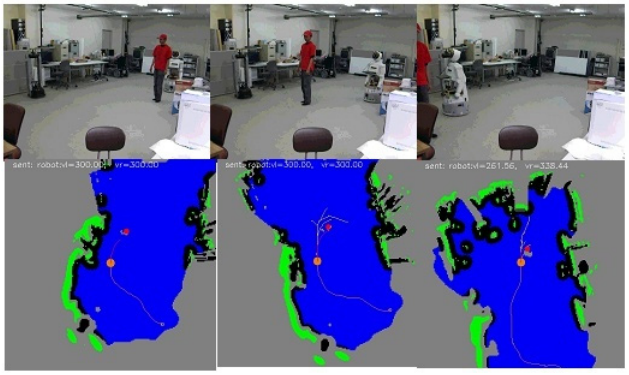
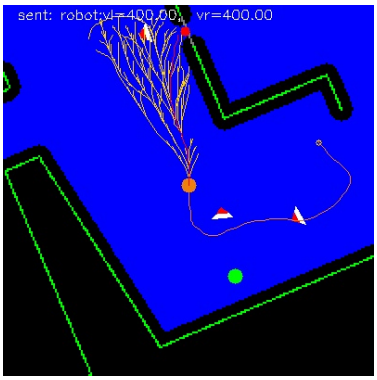
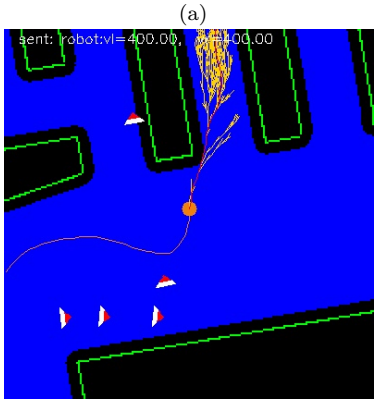


Fig. 8: Experiment of people tracking, (from left to right) sequences of the real world (top) and local map scene (bottom)



(a)



(b)

Fig. 6: Screenshot of simulation using Environment and People Movement Simulator. The orange circle at the center indicates the robot.

謝辞

本研究は NEDO 次世代ロボット知能化プロジェクトによるものである.

参考文献

1) S. M. LaValle and J. Kuffner. "Rapidly-exploring random trees: Progress and prospects". In Proc.of Fourth Intl. Workshop on Algorithmic Foundations on Robotics, 2000.

2) J. Bruce and M. Veloso. "Real-Time Randomized Path Planning for Robot Navigation", in Proc. IEEE/RSJ Conf. on Robotics and Systems, 2002.

3) C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, 2003.

4) S. Rodriguez, X. Tang, J.M. Lien, and N.M. Amato. "An obstacle-based rapidly-exploring random tree". In Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 895-900, 2006.

5) S. M. LaValleand J. Kuffner. "Randomized kinodynamic planning". In Proc.IEEE Int. Conf. Robotics and Automation, pages 473-479, 1999.

6) S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics, The MIT Press, 2005.

7) E. Plaku, L. E. Kavraki, and M. Y. Vardi. "A Motion Planner for a Hybrid Robotic System with Kinodynamic Constraints". In Proc. of the 2007 IEEE Int. Conf. on Robotics and Automation, pp. 692-697.

8) M. S. Hassouna, A. E. Abdel-Hakim and A. A. Farag., "PDE-based robust robotic navigation," Image and Vision Computing, vol. 27, pp. 10-18, 2009.

9) 安藤慶昭. 初心者のための RT ミドルウェア入門, 日本ロボット学会誌, Vol. 28, No. 5, pp. 550–555, 2010.

10) 石川裕基, 重村敦史, 佐竹純二, 三浦　純. 移動ロボット経路計画アルゴリズムの開発と多数人物動きシミュレータによる検証, 日本機械学会東海支部第 59 回講演会, 2010.

11) 佐竹純二, 三浦 純. ステレオビジョンを用いた移動ロボットの人物追従制御, 日本ロボット学会誌, Vol. 28, No. 9, pp. 1091–1099, 2010.