DEWS2006 4B-oi3

移動幾何オブジェクト 間の相互可視区間検索手法

草苅　良至† 　杉本　雄太†† 　能登谷淳一† 　笠井　雅夫†

† 秋田県立大学 システム科学技術学部　〒 015–0055 秋田県由利本荘市土谷字海老ノ口 84–4
†† 秋田県立大学 システム科学技術研究科　〒 015–0055 秋田県由利本荘市土谷字海老ノ口 84–4
E-mail: †{kusakari,m07b007,notoya,kasai}@akita-pu.ac.jp

あらまし　可視面決定等に代表される可視情報の計算は, 従来主にコンピュータグラフィクスや計算幾何学の分野で研究されてきている重要な問題である [2], [4], [5], [8], [9]. 一方, 近年, 連続的に運動する幾何オブジェクト (移動幾何オブジェクト) を計算機上で効率良く 扱う ための手法に関心が集っている [1], [10], [12]. 本稿では, 2 つの移動幾何オブジェクト が互いに等速直線運動する場合に, 移動幾何オブジェクト 間の “相互可視面” を効率良く 検索するための手法を提案する . ここで, 相互可視面とは一方の移動幾何オブジェクト から他方を “見た” ときに, 互いに可視であるような部分領域の組のことである . 本手法では, 移動幾何オブジェクト がそれぞれ n_0, n_1 個の頂点を持つ凸多角形である場合に, $O(N)$ の記憶量を用いて, $O(N \log N)$ 時間で相互可視区間検索木 (Mutual Visible-Intervals search tree,MVI 木) を構築する . ここで, N は 2 つの移動幾何オブジェクト の総頂点数であり , $N = n_0 + n_1$ である . MVI 木を用いること により , ある質問時刻 t における 2 つの凸多角形間の 1 次元相互可視面, 即ち相互可視区間を $O(\log N)$ 時間で検索することができる .

キーワード　時空間索引, 可視面決定, 相互可視区間, 移動幾何オブジェクト , MVI 木

A Method for Searching Mutual Visible-Intervals on Moving Objects

Yoshiyuki KUSAKARI†, Yuta SUGIMOTO††, Junichi NOTOYA†, and Masao KASAI†

† Faculty of Systems Science and Technology, Akita Prefectural University
Ebinokuchi 84–4, Tsuchiya, Yurihonjo, Akita 015–0055 Japan
†† Graduate School of Systems Science and Technology, Akita Prefectural University
Ebinokuchi 84–4, Tsuchiya, Yurihonjo, Akita 015–0055 Japan
E-mail: †{kusakari,m07b007,notoya,kasai}@akita-pu.ac.jp

Abstract　Computing visible information, such as visible-surface determination, is a significant problem and has been mainly studied in the fields of computational geometry and computer graphics [2], [4], [5], [8], [9]. Furthermore, recently, one might be attracted to the problems for dealing with the continuously moving objects in a geometrical space [1], [10], [12]. In this paper, we propose an indexing method, called a *Mutual Visible-Intervals search tree* (MVI-tree), by which one can efficiently find an “mutual visible-surfaces” on two moving objects from a query time. The “mutual visible-surfaces” is the pair of sub-regions which are “visible” each other. In this paper, we give an algorithm for constructing the MVI-tree from the set $\mathcal{M}$ of two convex polygons M_0 with n_0 vertices and M_1 with n_1 vertices, in the case where each convex polygon moves by uniform motion. Our algorithm construct the MVI-tree of $\mathcal{M}$ in time $O(N \log N)$ using space $O(N)$, where N is the total number of vertices and $N = n_0 + n_1$. One can find “mutual visible-intervals”, *i.e.* “one-dimensional mutual visible-surfaces”, of $\mathcal{M}$ in time $O(\log N)$ using the MVI-tree.

Key words　Spatio-Temporal Index, Visible-Surface Determination, Mutual Visible-Intervals, Moving Object, MVI-tree

1. はじめに

幾何オブジェクト (幾何学的対象) と 視点が与えられたとき, 視点から “見える” 幾何オブジェクト の一部を, その幾何オブジェクト の**可視面**という [7]. 可視面を求める問題は, 従来主にコンピュータグラフィクスや計算幾何学の分野等で注目されてきた問題であり , レイトレーシングへの応用を想定した研究が多く 行なわれてきている [2], [4], [5], [8], [9]. また, この問題は, 視点から “見えない” 部分 (隠面) を求める問題と 捉えることもでき , 特にコンピュータグラフィックスの分野での隠面除去にお

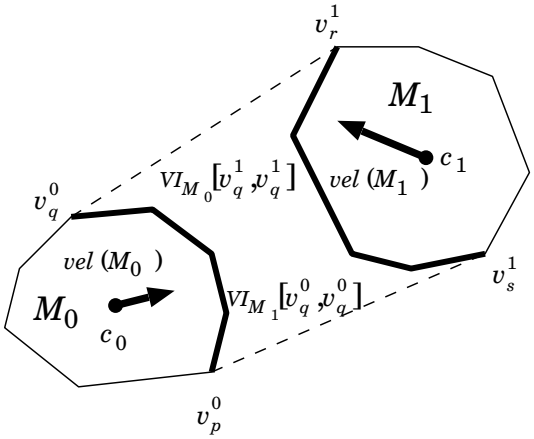

図 1　移動幾何オブジェクト 間の相互可視区間.
Fig. 1　The mutual visible-intervals of two moving objects.

いて重要な役割を持つ. しかし, コンピュータグラフックス分野以外でもこのような可視情報を扱う機会が増加しており, 多方面に渡る可視情報管理手法の開発が望まれている. 例えば, 可視情報管理手法の一つとして, 能登谷らは多数の幾何オブジェクトが与えられたとき, ある質問点から “見える” 幾何オブジェクトを効率良く検索する手法を開発している [8].

一方, 近年, 時間の経過と共に位置や形等の幾何情報が変化するような幾何学的対象に対する情報管理手法にも関心が集まっている [1], [10], [12]. 本稿では, このように時刻と共に位置や形等の幾何情報が変化するような幾何オブジェクトを**移動幾何オブジェクト** あるいは**移動体**と呼ぶ. これまでに, 移動体を効率的に保存・検索する手法の研究が多く行なわれている. 例えば, Patelらは, 移動体の軌跡が既知である場合に, 質問時刻における移動体の位置を効率よく検索する手法を提案している [10]. しかし, 移動体の情報管理に対して従来用いられている手法の多くは, 応用分野の知識を利用しながら, フレーム毎にシーンを更新する手法である [7]. ここで, フレームとはモデリングされた世界を時間軸を離散化して得られるある一時点であり, シーンとは各フレームにおけるモデルの幾何的な配置のことである. しかし, このようなシーン毎に幾何情報を再計算する手法は, 移動体の軌跡が既知である場合には計算量の面で不利である.

本研究では, 移動体の軌跡が既知である場合に可視情報を効率良く扱うためのデータ構造およびアルゴリズムの開発を目指す. 本稿では, $\mathbb{R}^2$ 上で 2 つの移動体 (凸多角形) が等速直線運動する場合を扱う. 特に, 移動体の変形や回転等の運動は考慮しない. このような運動を行なう 2 つの移動体に対して, 移動体間でお互いを “見た” ときに, 互いに可視であるような “一次元相互可視面” の効率的な検索手法を与える. 即ち, 本稿では “相互可視区間” を効率良く検索するための索引構造として, **相互可視区間検索木 (Mutual Visible-Intervals search tree)** を提案する. 図 1 に移動体の相互可視区間の概念図を示す. これ以降では相互可視区間検索木を単に MVI 木とも書く. 移動体が n_0, n_1 個の頂点を持つ凸多角形であるとき, $O(N)$ の記憶量を用いて $O(N \log N)$ 時間で MVI 木を構築するアルゴリズムを与える. ここで, N は移動体の総頂点数であり, $N = n_0 + n_1$ である. MVI 木を用いることにより, ある質問時刻 t における相互可視区間を $O(\log N)$ 時間で検索することができる. なお, MVI 木構築アルゴリズムでは, 多くの直線に対して, 移動体の接点を求める必要がある. そこで, 中間的なデータ構造として**接点検索木 (Tangent Point search tree)** を用いる. 接点検索木を用いれば, ある直線に接する凸多角形の頂点を高速に検索することができるので, 効率良く MVI 木を構築することができる.

本稿の以下は, 次のような構成である. 節 2. では用語や問題の定義を与える. 節 3. では相互可視区間の性質を示す. 節 4. では MVI 木を構築法を与える. 節 5. ではアルゴリズムの漸近計算量について述べる. 節 6. では実装結果を示す. 最後に, 節 7. で本稿をまとめる.

2. 準　　備

本節では用語や問題の定義を与える. まず, 静的な幾何学的対象に関する定義を与え, その後で動的な幾何学的対象に拡張する.

2.1 モデル化

本稿では, 移動体を $\mathbb{R}^2$ 直交空間で等速直線運動する凸多角形としてモデル化する. 移動体の集合を $\mathcal{M} = \{M_0, M_1\}$ と表す. 各移動体 $M_i, i = 0, 1$, は n_i 個の頂点を持つ凸多角形とし, $M_i = (v_0^i, v_1^i, \cdots, v_{n_i-1}^i)$ と表す. ここで, 各 $v_j^i, 0 \leqq j < n_i$, は M_i の頂点であり, 反時計廻りに順序付けられているとする. また, M_i の各辺を $e_j^i = (v_j^i, v_{j+1}^i), 0 \leqq j < n_i$, と表す. なお, 添字 i は 2 を法として表し $i+1 \pmod 2$ を単に $i+1$ と書き, 添字 j は n_i を法として表し $\pmod{n_i}$ は省略する. 移動体 M_i の各辺 e_j^i は, x 軸と y 軸のいずれとも平行でないと仮定する. 各移動体 M_i は各々等速直線運動するので, M_i はある固定された**速度ベクトル** $vel(M_i)$ にしたがって移動する. このような, 等速直線運動を単に**移動**と呼ぶ. 各移動体 M_i には, **代表点**と呼ばれる 1 点が定められているとする. M_i の代表点を c_i と書く. また, 代表点 c_i は移動体 M_i の真の内部にあると仮定する.

点 $p \in \mathbb{R}^2$ に対して, xy 絶対座標系での x 座標, y 座標をそれぞれ $x(p), y(p)$ と書く. また, 2 次元ベクトル v に対して, その x 成分, y 成分をそれぞれ $x(v), y(v)$ と書く.

2.2 静的な相互可視区間発見問題

凸多角形 M に対して, M の境界を $B(M)$ と書き, M の真の内部を $I(M)$ と書く. 即ち, $I(M) = M - B(M)$ である. 点 $p \in \mathbb{R}^2 - M$ から凸多角形の点 $q \in M$ が**可視**であるとは, 線分 $\overline{pq}$ が M の真の内部と共通部分を持たないことである. 即ち, $\overline{pq} \cap I(M) = \emptyset$ であるとき, 点 p から点 $q \in M$ が**可視**であるという. 凸多角形の外部の点 $p \in \mathbb{R}^2 - M$ から凸多角形の点 $q \in M$ が可視であるためには, $q \in B(M)$ でなければならない. 点 p から可視である境界 $B(M)$ の部分領域を (p からの)M の**可視領域**といい, $V_p(M)$ と書く. 点 $p_0 \in M_0$ と点 $p_1 \in M_1$ に対して, 線分 $\overline{p_0p_1}$ が 2 つの凸多角形の真の内部と共有部分を持たないとき, 点 p_0 と点 p_1 は**互いに可視**であるという. 即ち, $\overline{p_0p_1} \cap (I(M_0) \cup I(M_1)) = \emptyset$ であるとき, 点 p_0 と点 p_1 は互いに可視である. M_1 のいずれかの点から可視で

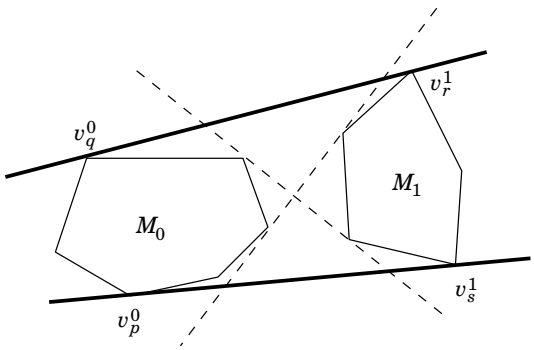

図 2　凸多角形間の共通接線.

Fig. 2　Four common tangents of two convex polygons.

あるような M_0 の部分領域を (M_1 からの)M_0 の**可視領域**といい, $V_{M_1}(M_0)$ と表す. $V_{M_1}(M_0)$ は, $B(M_0)$ 上で 2 つの頂点間を結ぶ道 $P = (v_p^0, v_{p+1}^0, \cdots, v_q^0)$ となる. このとき, 道 P の始点 v_p^0 と終点 v_q^0 を指定することによって, $V_{M_1}(M_0)$ を一意に定めることができる. このように, $V_{M_1}(M_0)$ を特定する M_0 の頂点対を, (M_1 からの)M_0 の**可視区間**といい $VI_{M_1}[v_p^0, v_q^0]$ と表す. 同様に, (M_0 からの)M_1 の可視領域 $V_{M_0}(M_1)$, (M_0 からの)M_1 の可視区間 $VI_{M_0}[v_r^1, v_s^1]$ も定める. M_1 からの M_0 の可視区間 $VI_{M_1}[v_p^0, v_q^0]$ と M_0 からの M_1 の可視区間 $VI_{M_0}[v_r^1, v_s^1]$ の対 $(VI_{M_1}[v_p^0, v_q^0], VI_{M_0}[v_r^1, v_s^1])$ を $\mathcal{M}$ の**相互可視区間**と呼ぶ. また, $\mathcal{M}$ の相互可視区間 $(VI_{M_1}[v_p^0, v_q^0], VI_{M_0}[v_r^1, v_s^1])$ を $MVI(p, q, r, s)$ とも書く. ここで, $p, q \in \{0, \cdots, n_0 - 1\}$, $r, s \in \{0, \cdots, n_1 - 1\}$ である. 静的な**相互可視区間発見問題**とは, $\mathbb{R}^2$ 上に 2 つの凸多角形 $\mathcal{M} = \{M_0, M_1\}$ が与えられたとき, 相互可視区間 $MVI(p, q, r, s)$ を求める問題である.

相互可視区間発見問題は 2 つの凸多角形間の共通接線を求める問題に帰着できる. 凸多角形 M の**接線** l とは, M と l が共通部分を持ち, しかも M が l の片側だけに存在するような直線である. M と接線 l の共通部分は, M の頂点かあるいは M の辺である. M と接線 l との共通部分が M の頂点 v_j であるとき, 頂点 v_j を接線 l に対する M の**接点**と呼ぶ. M と接線 l との共通部分が M の辺 e_j であるとき, 辺 e_j を接線 l に対する M の**接辺**と呼ぶ. 2 つの凸多角形 M_0, M_1 の**共通接線**は, M_0 の接線でありしかも M_1 の接線である直線である. 図 2 に示すように, 2 つの凸多角形 M_0, M_1 の共通接線は, 一般に 4 本引ける. 共通接線で定義される 2 つの半平面の内で, 同じ半平面だけに M_0,M_1 が含まれるときその共通接線を**同共通接線**と呼び, 異なる半平面に M_0,M_1 がそれぞれ含まれるときその共通接線を**異共通接線**と呼ぶ. 図 2 において, 同共通接線は実線で, 異共通接線は破線で示している. 2 本の同共通接線により相互可視区間を特定することができる. 2 本の同共通接線は計算幾何学での標準的な手法により $O(N)$ 時間で求めることができるので [9], [11], 相互可視区間は $O(N)$ 時間で求めることができる.

2.3　動的な相互可視区間検索問題

ここでは, 動的な幾何学的対象としての定義を与える.

時刻の集合を $T = [0, \infty) \subset \mathbb{R}$ とし, 時刻 $t \in T$ の変化に伴なって 2 つの凸多角形が 2 次元平面 $\mathbb{R}^2$ 上を移動する場合を考える. 各凸多角形 M_i の移動は空間 $\mathbb{R}^2 \times T$ 内の部分領域として捉えることができる. 空間 $\mathbb{R}^2 \times T$ を**移動空間**と呼ぶ. 移動する幾何オブジェクト o が占める $\mathbb{R}^2 \times T$ の部分領域を o の**軌跡**と呼ぶ. ここで, 幾何オブジェクト o には, 点, 線分, 直線, 多角形等を含む. ある時刻 $t_c \in T$ の**シーン**あるいは**スナップショット**とは, 移動空間と時刻 $t = t_c$ なる平面との交わりであり, $SS : T \to 2^{\mathbb{R}^2}$ という写像 SS として捉えることができる. 移動空間中の各幾何オブジェクト o に対して, スナップショット $SS(t)$ での幾何学的配置を $o(t)$ と表す. 例えば, 時刻 $t \in T$ における移動体を $M_i(t) = (v_0^i(t), \ldots, v_{n_i-1}^i(t))$ と表す. また, すべての時刻 $t \in T$ で $M_0(t) \cap M_1(t) = \emptyset$ と仮定する. $t = 0$ におけるスナップショ ショット $SS(0)$ を**初期配置**と呼ぶ. 初期配置 $SS(0)$, 移動体の速度ベクトル $vel(M_0), vel(M_1)$, およびある時刻 $t \in T$ が与えられたとき, $\mathcal{M}(t) = \{M_0(t), M_1(t)\}$ の相互可視区間 $(VI_{M_1(t)}[v_p^0(t), v_q^0(t)], VI_{M_0(t)}[v_r^1(t), v_s^1(t)])$ を求める問題を, **(移動幾何ブジェクト 間の) 相互可視区間検索問題**と呼ぶ. $\mathcal{M}(t)$ の相互可視区間を $MVI(t)(p, q, r, s)$ とも表す.

これまで, このような相互可視区間を求める手法の多くは, 時刻 t における各スナップショット $SS(t)$ を求めて, その後で $SS(t)$ 中で静的な手法を用いることで相互可視区間を求めていた. 即ち, 時刻 t における各移動体の配置 $M_0(t)$, $M_1(t)$ を求め, これらの凸多角形間で 2 本の同共通接線を求めることで $MVI(t)(p, q, r, s)$ を求めていた. 明かに, 初期配置 $M_i(0)$ と速度ベクトル $vel(M_i)$ より $M_i(t)$ は $O(n_i)$ 時間で求めることができる. また, 前述したように $M_0(t), M_1(t)$ 間での同共通接線は $O(N)$ 時間で求めることができる [9], [11]. したがって, ある時刻 $t \in T$ における相互可視区間 $MVI(t)(p, q, r, s)$ は $O(N)$ 時間で求めることができる. しかし, 相互可視区間を多くの時刻に対して保持更新しなければならないような場合には, このような計算を繰り返し行なう必要がある. 例えば, コンピュータグラフィックスの分野では総フレーム数 F の大きさを持つ離散的な時刻の集合 $T_D = (t_1, t_2, \cdots, t_F)$ を扱う必要がある. このような, 離散時刻の集合 T_D に対して, 毎時刻同共通接線を求めるような手法では, すべての相互可視区間を求めるために $O(FN)$ 時間必要となる. これに対して, 本稿では後述するように $O((F + N) \log N)$ 時間ですべての相互可視区間を求める手法を与える. F が十分大きい場合には提案手法が有効である.

3.　相互可視区間の性質

本節では, MVI 木の構築に必要となる相互可視区間の性質について述べる.

2 つの移動体 M_i, M_{i+1} に対して, たとえ 2 つの速度ベクトル共に $\mathbf{0}$ ベクトルでなかったとしても, 片方の移動体 M_i の速度ベクトルを $\mathbf{0}$ ベクトルとし, 残りの移動体 M_{i+1} の速度ベクトルを元の速度ベクトル同士の差で表すことができる. このように, 速度ベクトルを置き換えたとしても, 相互可視区間は双方が移動する場合と同じになる. 即ち, $\mathcal{M} = \{M_0, M_1\}$ に対して, $vel(M_0') = vel(M_0) - vel(M_0) = \mathbf{0}$, $vel(M_1') = vel(M_1) - vel(M_0)$ となるような移動体集合 $\mathcal{M}' = \{M_0', M_1'\}$ を考えれば, $\mathcal{M}(t)$ の $MVI(t)(p, q, r, s)$ と $\mathcal{M}'(t)$

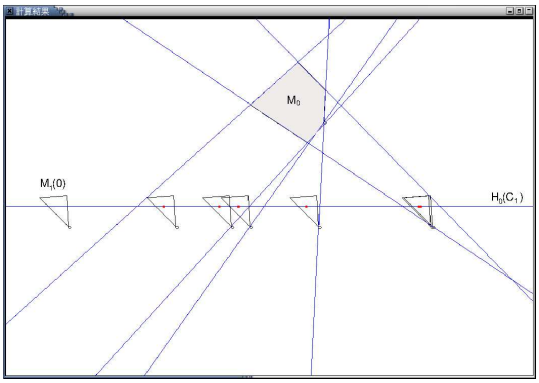

図 3　M_0 の変移空間.

Fig. 3　The transition space of M_o.

の $MVI(t)(p,q,r,s)$ はすべての時刻 $t \in T$ で同一となる. この操作は, 移動空間 $\mathbb{R}^2 \times T$ を $-vel(M_0)$ に基づいて剪断することに対応する. このようにして得られる 3 次元空間を M_0 の**静止空間**と呼ぶ. M_1 の**静止空間**も同様に定める. M_i の静止空間では $vel(M_i) = \mathbf{0}$ であるので, M_i を特に**静止体**と呼ぶこともある. M_i の静止空間において, 一般性を失うことなく, 移動体 M_{i+1} の速度ベクトルの y 成分を 0 とし, x 成分を非負とすることができる. この操作は, M_i の静止空間を $vel(M_{i+1})$ に基づいて回転することに対応する. 以降では M_i の静止空間では, 常に $y(vel(M_{i+1})) = 0, x(vel(M_{i+1})) \geqq 0$ と仮定する. また, 各 M_i に対して, M_i の静止空間を $t = 0$ なる平面に正射影して得られる 2 次元空間を M_i の**変移空間**と呼び H_i と表す. 図 3 に, 6 角形の静止体 M_0 の変移空間 H_0 を示す.

移動空間において, M_i の代表点 c_i の軌跡を含む直線を (M_i の) **移動線**と呼び C_i と書く. M_i の変移空間 H_i における移動体 M_{i+1} の移動線を $H_i(C_{i+1})$ と書く. このとき, 移動線 $H_i(C_{i+1})$ は x 軸と平行な直線である. 一般性を失うことなく, $y(c_i) > y(c_{i+1}(t))$ とする. 即ち, 静止体は移動線 $H_i(C_{i+1})$ より上にある. また, 変移空間 H_i において, 静止体 M_i の各辺 e_j^i, を延長して得られる直線を**延長線**と呼び $l(e_j^i)$ または l_j^i と書く. 静止体 M_i の延長線の集合を $L^i = \{l_j^i | 0 \leqq j < n_i\}$ と書く. これらの各延長線 $l_j^i \in L^i$ は x 軸と平行ではないと仮定する. たとえ, x 軸と平行である延長線 $l_j^i \in L^i$ があったとしても, 本提案手法を拡張することは容易である.

x 軸と平行ではない直線 l に対して, l で定まる 2 つの半平面の内で, $x(p) = -\infty$ なる点 $p \in \mathbb{R}^2$ を含む半平面を**左半平面**と呼び, $x(p) = +\infty$ なる点 $p \in \mathbb{R}^2$ を含む半平面を**右半平面**と呼ぶ. 直線 l とある点 p に対して, p が l の左半平面に含まれるとき p は l の**左**にあるといい, p が l の右半平面に含まれるとき p は l の**右**にあるという. 直線 l と凸多角形 M に対して, 凸多角形 M のすべての点が l の左にあるとき凸多角形 M は l の**左**にあるといい, 凸多角形 M のすべての点が l の右にあるとき凸多角形 M は l の**右**にあるという. 2 つの凸多角形 M_i, M_{i+1} とその同共通接線 l に対して, M_i, M_{i+1} が共に l の左にあるとき, 即ち l が M_i, M_{i+1} の右にあるとき, l を M_i,M_{i+1} の**右同共通接線**と呼ぶ. 同様に, M_i, M_{i+1} に対する**左同共通接線**を定める.

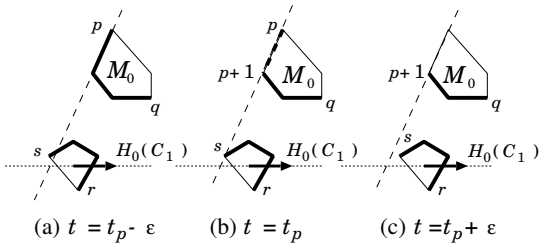

図 4　変更時刻 t_p 付近における M_0 の可視区間の減少.

Fig. 4　Decreasing the visible interval of M_0 around the change time t_p.

このとき次の補題が成り立つ.

［ 補題 1］　時刻 $t, t' \in T, (t < t')$ に対して

$$MVI(t)(p,q,r,s) \neq MVI(t')(p',q',r',s')$$

であるための必要十分条件は, 次の (p), (q), (r), (s) のいずれかが成り立つことである.

(p)　ある時刻 $t_p, t \leqq t_p < t'$ に対して, M_0 の変移空間において M_0 の延長線 $l(e_j^0) \in L^0$ が存在し, $l(e_j^0)$ が静止体 M_0 と移動体 $M_1(t_p)$ の左同共通接線である.

(q)　ある時刻 $t_q, t < t_q \leqq t'$ に対して, M_0 の変移空間において M_0 の延長線 $l(e_j^0) \in L^0$ が存在し, $l(e_j^0)$ が静止体 M_0 と移動体 $M_1(t_q)$ の右同共通接線である.

(r)　ある時刻 $t_r, t \leqq t_r < t'$ に対して, M_1 の変移空間において M_1 の延長線 $l(e_j^1) \in L^1$ が存在し, $l(e_j^1)$ が静止体 M_1 と移動体 $M_0(t_r)$ の左同共通接線である.

(s)　ある時刻 $t_s, t < t_s \leqq t'$ に対して, M_1 の変移空間において M_1 の延長線 $l(e_j^1) \in L^1$ が存在し, $l(e_j^1)$ が静止体 M_1 と移動体 $M_0(t_s)$ の右同共通接線である.

証明　相互可視区間 $MVI(t)(p,q,r,s)$ が変化するのは, 静止体 M_i と移動体 $M_{i+1}(t)$ の同共通接線が, どちらかの辺の延長線となっているときだけであることは容易に確かめられる. よって, 以下では, (p), (q), (r), (s) が成立つときに, 相互可視区間 $MVI(t)(p,q,r,s)$ が変化することを示す.

(p): 時刻 t_p において, (p) の条件を満す延長線を $l(e_j^0)$ とし, 相互可視区間を $MVI(t_p)(p,q,r,s)$ とする. (図 4 参照.) このとき, $l(e_j^0)$ は辺 e_j^0 を接辺とする M_0 の接線である. したがって, $p = j$ であり相互可視区間は $MVI(t)(j,q,r,s)$ である. 任意の微少な正数 $\varepsilon > 0$ に対して, 静止体 M_0 と移動体 $M_1(t_p - \varepsilon)$ の左同共通接線は v_j^0 を接点とする M_0 の接線である. したがって, 時刻 $t_p - \varepsilon$ における相互可視区間は, $MVI(t_p - \varepsilon)(j,q,r,s)$ である. 一方, 時刻 $t_p + \varepsilon$ においては, e_j^0 のもう一つの頂点 v_{j+1}^0 を接点とする M_0 の接線が, 静止体 M_0 と移動体 $M_1(t_p + \varepsilon)$ の左同共通接線である. よって, 時刻 $t_p + \varepsilon$ における相互可視区間は, $MVI(t_p + \varepsilon)(j+1,q,r,s)$ である.

以上より, 時刻 t_p 前後では, M_0 の可視区間が $VI_{M_1}[v_p^0, v_q^0]$ から $VI_{M_1}[v_{p+1}^0, v_q^0]$ へ減少し, 相互可視区間 $MVI(t)(p,q,r,s)$ の p が変化することがわかる.

(q): 時刻 t_q において, (q) の条件を満す延長線を $l(e_j^0)$ とし, 相互可視区間を $MVI(t_q)(p,q,r,s)$ とする. (図 5 参照.) このとき, $l(e_j^0)$ は辺 e_j^0 を接辺とする M_0 の接線である. よって, $q = j+1$

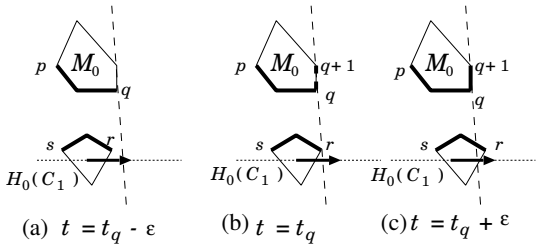

図 5　変更時刻 t_q 付近における M_0 の可視区間の増加.

Fig. 5　Increasing the visible interval of M_0 around the change time t_q.

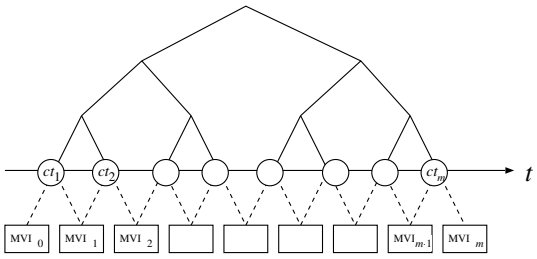

図 6　変更時刻集合 CT と MVI 木.

Fig. 6　The MVI-tree and the change time set CT.

であり 相互可視区間は $MVI(t)(p, j+1, r, s)$ である. 任意の微少な正数 $\varepsilon > 0$ に対して, 静止体 M_0 と 移動体 $M_1(t_q - \varepsilon)$ の右同共通接線は v_j^0 を接点とする M_0 の接線である. したがって, 時刻 $t_q - \varepsilon$ における相互可視区間は, $MVI(t_q - \varepsilon)(p, j, r, s)$ である. 一方, 時刻 $t_q + \varepsilon$ においては, e_j^0 のもう一つの頂点 v_{j+1}^0 を接点とする M_0 の接線が, 静止体 M_0 と 移動体 $M_1(t_q + \varepsilon)$ の右同共通接線である. よって, 時刻 $t_p + \varepsilon$ における相互可視区間は, $MVI(t_p + \varepsilon)(p, q+1, r, s)$ である.

以上より , 時刻 t_q 前後では M_0 の可視区間が $VI_{M_1}[v_p^0, v_q^0]$ から $VI_{M_1}[v_p^0, v_{q+1}^0]$ へ増加し, 相互可視区間 $MVI(t)(p, q, r, s)$ の q が変化することがわかる.

(r)**:** M_1 の変移空間において (p) の証明と 同様にして, 時刻 t_r 前後では M_1 の可視区間が減少し $MVI(t)(p, q, r, s)$ の r が変化することがわかる.

(s)**:** M_1 の変移空間において (q) の証明と 同様にして, 時刻 t_s 前後では M_1 の可視区間が増加し $MVI(t)(p, q, r, s)$ の s が変化することがわかる. □

補題 1 より 相互可視区間が変化する時刻は離散的である. 相互可視区間 $MVI(t)(p, q, r, s)$ が変化する時刻を**変更時刻**(Change Time) と 呼ぶ. すべての変更時刻の集合を CT と 書く. CT は離散的な T の真部分集合である. 補題 1 の (p), (q), (r), (s) が成立する時刻の集合をそれぞれ, CT_p, CT_q, CT_r, CT_s と書く. また, $CT^0 = CT_p \cup CT_q$, $CT^1 = CT_r \cup CT_s$ とする. このとき, CT^0 は M_0 の可視区間が変化する時刻の集合であり , CT^1 は M_1 の可視区間が変化する時刻の集合である. また, $CT = CT^0 \cup CT^1 = CT_p \cup CT_q \cup CT_r \cup CT_s$ である. 例えば, 時刻 $t_p \in CT_p \subseteqq CT^0 \subseteqq CT$ の前後において, 相互可視区間は $MVI(t_p - \varepsilon)(p, q, r, s)$ から $MVI(t + \varepsilon)(p+1, q, r, s)$ へと 変化する (図 4 参照). 一方, CT 以外の時刻 $t \notin CT$ に対しては, t の前後で相互可視区間は変化せず, $MVI(t \pm \varepsilon)(p, q, r, s) = MVI(t)(p, q, r, s)$ である.

CT は時刻の集合なので自然に全順序をつけることができ, $CT = (ct_1, ct_2, \cdots, ct_m)$ と表すことができる. 各 k, $1 \leqq k < m$ に対して, $ct_k \leqq t < ct_{k+1}$ では $MVI(t)(p, q, r, s)$ は同一である. $ct_k \leqq t < ct_{k+1}$ での相互可視区間を MVI_k と 書く. また, $t < ct_1$ での相互可視区間を MVI_0, $ct_m \leqq t$ での相互可視区間を MVI_m と 書く. 相互可視区間のすべての種類からなる集合を $\mathcal{MVI}$ とする. このとき, $\mathcal{MVI}$ には (変更) 時刻を基に全順序を付けることができ, $\mathcal{MVI} = (MVI_0, MVI_1, \cdots, MVI_m)$ と 表すことができる. **相互可視区間検索木 (MVI 木)** とは, CT に基づく 木構造索引を持ち, CT の各要素から $\mathcal{MVI}$ の各要素を参照できる索引構造である. ここで, MVI 木の木構造索引部分は, 時刻をキー値とした平衡木として実現できる. 実際, 平衡木として AVL 木や二色木等 [3] を用いれば, ある時刻における相互可視区間を $O(\log N)$ 時間で求めることができる. また, CT の各要素から $\mathcal{MVI}$ の各要素を参照する構造も 時刻を元に自然に構成することができる. 図 6 に MVI 木を示す.

4. MVI 木の構築

節 3. より , MVI 木を構築するためには, 変更時刻集合 CT および相互可視区間集合 $\mathcal{MVI}$ を求める必要がある. 本節では, CT および $\mathcal{MVI}$ の求め方を与え, MVI 木を構築するアルゴリズムを与える.

まず, MVI 木を構築するアルゴリ ズムの概略を次に示す.

[アルゴリ ズム 1]　**(Construct MVI-tree)**

(MVI1)　M_0 の変移空間に基づいて CT^0 を求める.

(MVI2)　M_1 の変移空間に基づいて CT^1 を求める.

(MVI3)　CT^0 および CT^1 を時刻を元にマージして, CT を求める.

(MVI4)　初期配置 $\mathcal{M}(0) = \{M_0(0), M_1(0)\}$ より , 初期相互可視区間 $MVI_0 = MVI(0)(p, q, r, s)$ を求める.

(MVI5)　MVI_0 および CT より $\mathcal{MVI}$ を求め, MVI 木を構築する.

」

アルゴリ ズム **Construct MVI-tree** のより 詳細な実現法を以下で与える.

4.1　接点検索木

M_i の変移空間において, 静止体 M_i の延長線 $l_j^i \in L^i$ に接するときの移動体 $M_{i+1}(t)$ の接点 v_k^{i+1} を求めることで CT^i を求めることができる. 単純には, l_j^i を通過する $M_{i+1}(t)$ の頂点を順次調べれば, l_j^i に対する $M_{i+1}(t)$ の接点 v_k^{i+1} を求めることができる. しかし, このような単純な手法では, 1 本の延長線 l_j^i に対して $O(n_{i+1})$ 時間かかってしまい, CT^i を求めるために $O(n_0 n_1) = O(N^2)$ 時間かかってしまう. これに対して, 本稿ではより 高速に CT^i を求める 手法を与える. 主なアイディアは, 移動体 M_{i+1} に対して繰り 返し 接点を求める 必要があることに注目し, 接点を求めるための中間的なデータ 構造として**接点検索木 (Tangent Point search tree)** を構築すること である. これ以降, 接点検索木を TP 木とも 書く. ここでは, この TP 木について述べる.

2 点 $p_1, p_2 \in \mathbb{R}^2$ に対して, ベクトル $\overrightarrow{p_1p_2}$ の**角度**は, $+x$ 方向から反時計周りに計られ, $[0, 2\pi)$ のいずれかの値を取るものとする. ベクトル $\overrightarrow{p_1p_2}$ の角度を $\theta(\overrightarrow{p_1p_2})$ と表す. また, 角度どうしの加減算は常に $[0, 2\pi)$ の値を取るものとし, $(\mathrm{mod}\ 2\pi)$ は省略する. 直線 l に対する**法線ベクトル** $\mathbf{n}(l)$ は, 2 点 $p_1, p_2 \in l$ に対して, $\theta(\mathbf{n}(l)) = \theta(\overrightarrow{p_1p_2}) \pm \frac{\pi}{2}$ を満たすような大きさ 1 のベクトルである. 辺 $e_j^i = (v_j^i, v_{j+1}^i)$ に対する**法線ベクトル** $\mathbf{n}(e_j^i)$ は, $\theta(\mathbf{n}(e_j^i)) = \theta(\overrightarrow{v_j^i v_{j+1}^i}) - \frac{\pi}{2}$ を満すような大きさ 1 のベクトルである. 明らかに辺 e_j^i に対する法線ベクトルは $\overrightarrow{v_j^i v_{j+1}^i}$ と垂直で M_i の外部に向かうベクトルである. M_i に対して, 一般性を失うことなく, すべての辺 e_j^i の法線ベクトルの中で, e_0^i が最小の角度 $\theta(\mathbf{n}(e_0^i))$ を持つとしてよい.

このとき, 明らかに次の 2 つの補題が成立つ.

[補題 2] M_i の辺 e_j^i に対して, 次式が成立つ.

$$\theta(\mathbf{n}(e_0^i)) < \theta(\mathbf{n}(e_1^i)) < \cdots < \theta(\mathbf{n}(e_{n_i-1}^i)).$$

」

[補題 3] M_i の変移空間において, x 軸と平行ではない直線 l に移動体 $M_{i+1}(t)$ が頂点 v_k^{i+1} で接するための必要十分条件は次の (i) あるいは (ii) が成立つことである.

(i)

$$\theta(\mathbf{n}(e_{k-1}^{i+1})) \leqq \theta(\mathbf{n}(l)) \leqq \theta(\mathbf{n}(e_k^{i+1}))$$

(ii)

$$\theta(\mathbf{n}(e_{k-1}^{i+1})) \leqq \theta(\mathbf{n}(l)) + \pi \leqq \theta(\mathbf{n}(e_k^{i+1}))$$

」

補題 2 より, M_i の各辺 e_j^i には, その法線ベクトルの角度 $\theta(\mathbf{n}(e_j^i))$ により, (環状に) 順序を付けることができる. M_i の**接点検索木** $TP(M_i)$ とは, M_i の各辺 e_j^i を要素に持ち, その法線ベクトルの角度をキー値とするような平衡木である. ただし, $e_{n_i-1}^i$ の "次" は辺 e_0^i とする. したがって, 空要素の接点検索木に, 各辺 $e_j^i, 0 \leqq j < n_i$, を角度 $\theta(e_j^i)$ を計算しながら, 順に挿入することによって接点検索木 $TP(M_i)$ を構築することができる. また, 補題 3 より, $TP(M_{i+1})$ を用いることにより, M_i の変移空間において直線 l と接する移動体 M_{i+1} の接点 v_k^{i+1} を効率良く求めることができる.

4.2 MVI 木構築の詳細

本節では, **Construct MVI-tree** の詳細について述べる.

まず, M_i の変移空間における幾何問題を解くことで CT^i を求めることでができることを示す. これにより, (MVI1), (MVI2) の各ステップを実現できる. 直線 l をベクトル $\overrightarrow{p_1p_2}$ 方向に平行移動して得られる直線を $l(\overrightarrow{p_1p_2})$ と書く. CT^i を求めるアルゴリズム **Find** CT^i が以下のように構成できる.

[アルゴリズム 2] (**Find** CT^i)

(CT1) 移動体 M_{i+1} の接点検索木 $TP(M_{i+1})$ を構築する.

(CT2) 静止体 M_i の各延長線 $l_j^i \in L^i$ に対して, 以下を行なう.

(CT3) $TP(M_{i+1})$ に対し $\theta(\mathbf{n}(l_j^i))$ および $\theta(\mathbf{n}(l_j^i)) + \pi$ で検索し, l_j^i に接するときの 2 接点 $v_k^{i+1}, v_{k'}^{i+1}$ をそれぞれ求める.

(CT4) 直線 $l_j^i(\overrightarrow{v_k^{i+1}c_{i+1}})$, $l_j^i(\overrightarrow{v_{k'}^{i+1}c_{i+1}})$ と移動線 $H_i(C_{i+1})$ との交点 $cp_k, cp_{k'}$ をそれぞれ求める. (図 7 参照.)

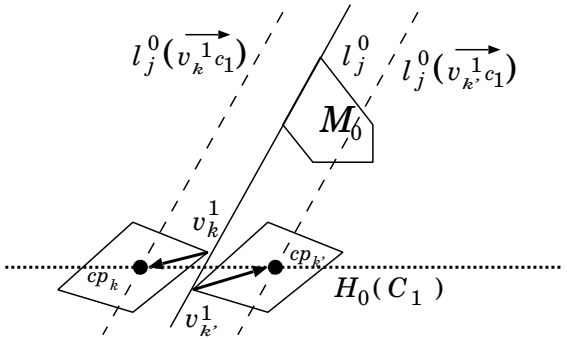

図 7 延長線と変更時刻の関係.

Fig. 7 The relationship between an extended line and a change time.

(CT5) 交点 $cp_k, cp_{k'}$ の内で, l_j^i に対して静止体の代表点 c_i と同じ側にあるものを求め, cp^* とする. 交点 cp^* に移動体 M_{i+1} の代表点 c_{i+1} が到達する時刻 ct を求め, CT^i に挿入する.

」

(CT3) において, 各延長線 l_j^i に対して, 2 つの接点 $v_k^{i+1}, v_{k'}^{i+1}$ が求められる. このことは, 静止体 M_i と移動体 $M_{i+1}(t)$ が同共通接線と異共通接線を持つことに対応する. 即ち, 移動体 M_{i+1} において, 2 つの接点 $v_k^{i+1}, v_{k'}^{i+1}$ は, 一方が同共通接線の接点であり, 他方が異共通接線の接点である. よって, 延長線 l_j^i と M_i, M_{i+1} の位置関係より, l_j^i が同共通接線と異共通接線のどちらであるかを判定する. この判定は, (CT5) で行なう. 代表点 c_i, c_{i+1} はそれぞれ M_i, M_{i+1} の内部にあるので, 共通接線 l が M_i, M_{i+1} の同共通接線であるときには, c_i と c_{i+1} は l の同じ側にある. したがって, 代表点 c_i, c_{i+1} と延長線 l_j^i の位置関係より, l_j^i が同共通接線と異共通接線のどちらであるかを判定できる. 図 7 に延長線と変更時刻の関係を示す.

次に, (MVI3) の実現について述べる. M_0 の変移空間に対して, アルゴリズム **Find** CT^0 を実行することにより, $CT^0 = (ct_1^0, \cdots, ct_{m_0}^0)$ を求めることができる. また, M_1 の変移空間に対して, **Find** CT^1 を行なうことで, $CT^1 = (ct_1^1, \cdots, ct_{m_1}^1)$ を求めることができる. 典型的なマージアルゴリズム技法 [3] により, CT^0 および CT^1 を時刻を元にマージし $CT = (ct_1, \cdots, ct_m)$ を求めることができる. ここで, $m_0 = |CT^0|$, $m_1 = |CT^1|$, $m = |CT| (= m_0 + m_1)$ である.

次に, (MVI4) の実現について述べる. 前述したように, 初期配置 $\mathcal{M}(0) = \{M_0(0), M_1(0)\}$ に対して $M_0(0)$ と $M_0(1)$ の同共通接線を求め [9], [11], 初期相互可視区間 $MVI(0)(p, q, r, s)$ を求めることができる.

最後に, $\mathcal{MVI}$ の求め方を示し, (MVI5) の実現法を与える. $CT = (ct_1, \cdots, ct_m)$ の各要素 $ct_k \in CT$ は 4 つの部分集合 CT_p, CT_q, CT_r, CT_s のいずれか 1 つだけに属する. $ct_k \in CT$ がどの部分集合に属するかは, CT を求める時に判定できる. 即ち, (CT5) で CT^i に挿入される ct_k に対しては, 代表点 c_i, c_{i+1} と延長線 l_j^i の位置関係より, どの部分集合に属するかを判定できる. また, ct_k の種類と補題 1 の証明より, MVI_{k-1} から MVI_k を求められる. 例えば, $ct_k \in CT_p$ であるとき, $MVI_{k-1} = MVI(ct_k - \varepsilon)(p, q, r, s)$ から $MVI_k =$

$MVI(ct_k+\varepsilon)(p+1,q,r,s)$ を求められる. $CT=(ct_1,\cdots,ct_m)$ の各要素 $ct_k \in CT$ を ct_1 から順に辿ることで, $MVI_0 = MVI(0)(p,q,r,s)$ から $MVI_m = MVI(\infty)(p,q,r,s)$ まで順に $MVI_k = MVI(t)(p,q,r,s)$ を求められる.

5. 漸近計算量

本節では, 提案アルゴリズムの漸近計算量解析を行なう.

まず, アルゴリズム **Find** CT^i の漸近計算量について述べる. 一般に m 個の要素を持つ平衡木は, 空の要素の平衡木に m 個の要素を順に挿入することで, $O(m\log m)$ 時間で構築することができる [3]. このとき, 平衡木の高さは $O(\log m)$ である. 移動体 M_{i+1} の接点検索木 $TP(M_{i+1})$ は, n_{i+1} 個の要素を持つ平衡木であるので, $O(n_{i+1}\log n_{i+1}) = O(N\log N)$ 時間で構築できる. よって, (CT1) は $O(N\log N)$ 時間で実行できる. また, (CT2)-(CT5) では, 静止体 M_i の各延長線 $l_j^i \in L^i$ に対して, 接点検索木 $TP(M_{i+1})$ への検索が 2 回行なわれるだけである. $TP(M_{i+1})$ の高さは $O(\log n_{i+1})$ であるので, 各検索は $O(\log n_{i+1})$ 時間で行なえる. $|L^i| = n_i$ なので, (CT2)-(CT5) は $O(n_i \log n_{i+1})$ 時間で行なえる. また, これらのアルゴリズムに必要な領域量が $O(N)$ であることも容易に確かめられる.

以上より, 次の補題が成立つ.

［補題 4］ **Find** CT^i の時間計算量は $O(N\log N)$ であり, 領域計算量は $O(N)$ である. 」

次に, アルゴリズム **Construct MVI-tree** の漸近計算量について述べる. 変更時刻の集合 CT に関して次の補題が成り立つ.

［補題 5］ CT の総数は $O(N)$ である.

証明 M_i の変移空間において, 延長線集合 L^i の各要素 l_j^i が静止体 M_i と移動体 $M_{i+1}(t)$ の同共通接線となることは, 丁度 1 回しかない. したがって, $|CT^i| = |L^i| = n_i$ である. よって, $|CT| = |CT^0| + |CT^1| = n_0 + n_1 = N$ である. □

補題 4, 5 より, (MVI1), (MVI2) はそれぞれ $O(N\log N)$ 時間で行なえる. 補題 5 より, (MVI3) のマージは $O(|CT|) = O(N)$ 時間で行なえる [3]. (MVI4) において, 初期相互可視区間 $MVI(0)(p,q,r,s)$ を求めることは $O(N)$ 時間で行なえる [9], [11]. また, 節 4. の議論により, $\mathcal{MVI}$ を求めることも $O(N)$ 時間で行なえる. さらに, MVI 木の木構造索引部分は平衡木であるので, その構築は補題 5 より $O(N\log N)$ 時間で行なえる. CT の各要素と $\mathcal{MVI}$ の各要素を結合することは $O(N)$ 時間で容易に行なえる. さらに, **Construct MVI-tree** に必要な領域量が $O(N)$ であることも容易に確かめられる.

以上より, 次の補題が成立つ.

［補題 6］ **Construct MVI-tree** の時間計算量は $O(N\log N)$ であり, 領域計算量は $O(N)$ である. 」

補題 5 より, MVI 木の木構造索引部分の高さは $O(\log N)$ である. したがって, MVI 木を用いた相互可視区間 MVI_k の検索は $O(\log N)$ 時間で行なうことができる. また, F 個の離散的な時刻の集合 $T_D = (t_1, t_2, \cdots, t_F)$ に対して, すべての相互可視区間の列 $\mathcal{MVI}\,(T_D) = (MVI(t_1), MVI(t_2), \cdots, MVI(T_F))$ を求める問題を, T_D に対する **相互可視区間列発見問題** と呼ぶ. MVI 木を用いることにより $\mathcal{MVI}\,(T_D)$ は, $O(F\log N)$ 時間で求められる.

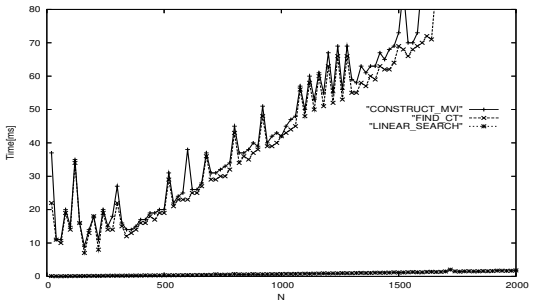

図 8　MVI 木構築の実験結果.

Fig. 8　Experimental results for constructing the MVI-tree.

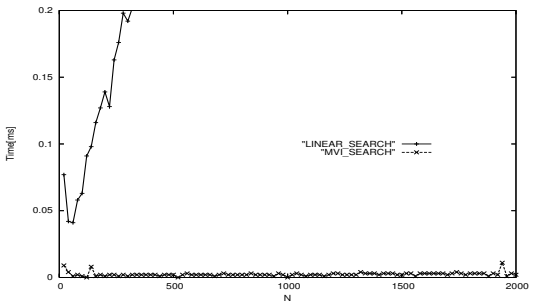

図 9　相互可視区間検索の実験結果.

Fig. 9　Experimental results for searching an mutual visible-intervals.

以上より, 次の定理が成立つ.

［定理 1］ 相互可視区間検索問題は $O(N\log N)$ 時間の前処理を行なうことで, $O(N)$ の記憶量を用いて $O(\log N)$ 時間で解くことができる. T_D の相互可視区間列発見問題は $O(F+N)$ の記憶量を用いて, $O((F+N)\log N)$ 時間で解くことができる. 」

6. 実装評価

提案手法を実装し計算時間を計測する実験を行なった. 本節ではその実験結果を示す.

実験では, TP 木や MVI 木で利用する平衡木として二色木を用いた. また, 実験で用いた計算機は, CPU が Pentium4 3GHz, OS が Linux 2.4 である. この環境の下で Java(JDK1.4) を用いて実装を行なった.

各移動体 M_i は, 半径 r_i の円に含まれ n_i 個の頂点を持つランダムな凸多角形として生成する. ここでは, M_0 と M_1 がほぼ同じ大きさを持つ場合の結果だけを示す. 即ち, $r_0 = r_1 = 1$ として, M_0, M_1 を生成した結果を示す. なお, $r_0 \neq r_1$ の場合も同様の結果が得られている. また, 予備的な実験の結果, M_0 の代表点 c_0 と M_1 の移動線 $H_0(C_1)$ の距離 $dist$ は, 計算時間にあまり大きな影響を与えなかった. よって, $dist = 20$ の場合についての結果だけを示す. また, 移動体の頂点数が等しく, $n_0 = n_1 (= N/2)$ である場合だけを示す. $n_0 \neq n_1$ の場合も同様の結果が得られている. このとき, 総頂点数 N を $N = 20$ から $N = 2000$ まで増加させ実験を行なった. (なお, 実験では, Java 言語の `double` 型の表現能力の限界により, $n_i = 1000$ までしか信頼できる凸多角形を得ることができなかった.) MVI 木構築に関する実験結果を図 8 に, 相互可視区間検索に関する実験結果を図 9 に示す.

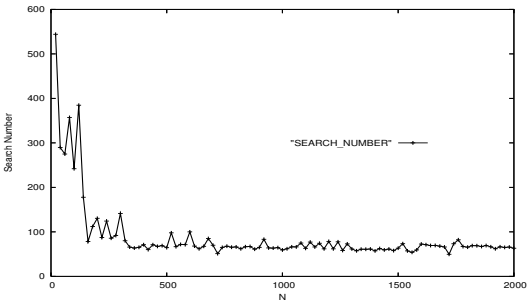

図 10　MVI 木構築が優位となる検索回数.

Fig. 10　Search numbers from which the constructing MVI tree method dominates over the linear search method.

図 8, 9 において, 各系列の意味は以下の通り である.

- LINEAR_SEARCH は, 同共通接線を求める標準的な幾何アルゴリズムの実行時間である.
- FIND_CT は MVI 木構築の中で, 変更時刻 CT を見つけるために用いられた実行時間である.
- CONSTRUCT_MVI は MVI 木構築全体での実行時間である.
- MVI_SEARCH は, MVI 木を用いて相互可視区間 MVI_k を検索する際の実行時間である.

図 8 より, MVI 木構築に用いられた計算時間のほとんどが, 変更時刻の集合 CT を求める部分であることがわかる. また, 図 9 より, MVI 木を用いた相互可視区間の検索は, 相互可視区間発見の幾何アルゴリズムを用いるよりも, 大幅に高速であることが読み取れる.

ここで, 相互可視区間列発見問題に対して, 本提案手法が優位となるための検索回数を求める. 総頂点数 N に対して, MVI 木構築に用いられる計算時間を $T_M(N)$ とし, MVI 木を用いた 1 回の検索時間を $t_M(N)$ とし, 同共通接線より MVI_k を求める計算時間を $t_L(N)$ とする. このとき, 本手法が優位になるためには, 検索回数 $F(N)$ に関して次式が成立つ必要がある

$$T_M(N) + F(N) \times t_M(N) \leqq F(N) \times t_L(N)$$

$$\therefore F(N) \geqq \frac{T_M(N)}{t_L(N) - t_M(N)}$$

実験結果に基づいて, 上式より計算した検索回数 $F(N)$ を図 10 に示す. 図 10 より, 総頂点数が 200 を越え, 検索回数が 100 回を越えるような場合には, 本提案手法が有効である.

7. むすび

本稿では, 移動幾何オブジェクトの軌跡が既知である場合に, 相互可視区間を効率良く検索する手法の提案を行なった. 即ち, 凸多角形の移動体 M_0, M_1 がそれぞれ等速直線運動している場合に, 相互可視区間を検索できる索引構造 MVI 木を $O(N)$ の記憶量を用いて $O(N \log N)$ 時間で構築するアルゴリズムを与えた. ここで, N は総頂点数であり, $N = n_0 + n_1$ である. この MVI 木を用いることにより, ある質問時刻 t における相互可視区間を $O(\log N)$ 時間で検索することができる. また, MVI 木の格納に必要な記憶量は $O(N)$ である.

本提案手法を実装した実験結果より, 総頂点数 N が 2000 以下では, MVI 木構築に必要なほとんどの計算時間は変更時刻の集合 CT を求める部分であることを確認した. また, MVI 木を構築することによって, 1 回の検索に用いられる計算時間が大幅に小さくなることも確認した. 実験より, 総頂点数が 200 を越え, 検索回数が 100 回を越えるような場合に, MVI 木を構築する提案手法が繰り返し接線を求める方法よりも優位となった. 実装においては平衡木としては二色木を用いたが, AVL 木, B^+ 木等の他の平衡木を利用して MVI 木を実装し提案手法と各種平衡木との整合性を検証することは今後の課題である.

本稿では時刻集合 $T = [0, \infty)$ に対して相互可視区間を求める手法を与えた. 時刻集合 $T = (-\infty, \infty)$ に対して相互可視区間を求めるように, 本稿の手法を以下のように拡張することができる. 初期配置が与えられた時刻 $t = 0$ で時刻集合 $T = (-\infty, \infty)$ を $T^- = (-\infty, 0]$ と $T^+ = [0, +\infty)$ の 2 つに分割する. T^- に対する MVI 木および T^+ に対する MVI 木を求め, それらの木をマージすることにより, $T = T^- \cup T^+$ に対する MVI 木を求める. ここで, T^- に対する MVI 木は, 各移動体 M_i の速度ベクトルを $-vel(M_i)$ とすれば本稿と同様に求めることができる.

提案手法を, 3 次元の凸多面体同士の相互可視面検索に拡張することや, 移動体の運動が等速直線運動に限らないより一般的な運動である場合の相互可視面検索手法を開発することが今後の課題である. また, 3 体以上の移動体集合に対する可視情報管理手法を開発することも今後の課題である.

文　献

[1] P. K. Agarwal, L. J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. Har-peled, J. Hershberger, C. Jensen, L. Kavraki, P. Koehl, M. Lin, D. Manocha, D.Metaxas, B. Mirtich, and D. Mount, "Alogorithmic Issues in Modeling Motion, " ACM computing Surveys, **34**, 4, pp.550–572(2002).

[2] J. Bittner, "Hierachical Techniques for Visibility Determination, " Postgraduate Study Report DC-PSR-99-05, Czech Technical University, (1999).

[3] T. H. Cormen, C.E. Leiserson, and R.L. Rivest, Intoroduction to Algorithms, MIT press, Cambridge, MA(1990).

[4] S.Coorg and S.Teller, "Real-time occlusion culling for models with large occluders, " SI3D:Proc. of Symp. on Interactive 3D graphics, New York, NY, USA, ACM Press, pp.83–ff(1997).

[5] M. deBerg, M. van Kreveld, M. Overmars, and O.Schwarzkopf, Computational Geometry Algorithms and Applications, Springer-Verlag(1997).

[6] D. Gordon and S.Chen, "Front-to-back display of BSP trees, " IEEE Computer Graphics and Applications, **11**, 5, pp.79–85(1991).

[7] J. D. Foley, A. van Dan, S. K. Feiner, and J.F. Hughes, Computer graphics:principles and practice 2nd ed. in C, Addison-Wesley (1991).

[8] 能登谷淳一, 杉本雄太, 草苅良至, 笠井雅夫, "空間データベースシステムのために可視探索手法, " 日本データベース学会 Letters, **4**, 2, pp.9–12(2005).

[9] J. O'Rourke, Computational Geometry in C, Cambridge, (1998).

[10] J. M. Patel, Y. Chen, and V. P. Chakka, "STRIPES: An Efficient Index for Predicated Trajectories, " Proc. of SIGMOD Conference 2004, Paris, France pp. 637-646(2004).

[11] 譚 学厚, 平田 富夫, 計算幾何学入門, 森北出版, (2001).

[12] Y. Tao, D. Papadias, and J. Sun, "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries, " VLDB, pp.790–801 (2003).