Methods for Searching Mutual Visible Intervals on Moving Objects

Yoshiyuki Kusakari†, Yuta Sugimoto‡, Junichi Notoya†, and Masao Kasai†
† Department of Electronics and Information Systems,
Faculty of Systems Science and Technology, Akita Prefectural University
‡ Department of Electronics and Information Systems,
Graduate School of Systems Science and Technology, Akita Prefectural University
E-mail:{ kusakari,m07b007,notoya,kasai}@akita-pu.ac.jp

Abstract Computing visible information, such as a visible surface determination, is a significant problem and has been mainly studied in the fields of computational geometry and/or computer graphics[2, 4, 6, 10, 11, 13]. Furthermore, recently, one might be attracted to problems for dealing with continuously moving objects in a geometrical space [1, 9, 12, 14]. In this paper, we propose two indexing methods, called an *Mutual Visible Intervals search tree* (MVI-tree) and an *Mutual Visible Intervals search list* (MVI-list). Using each index, one can efficiently find "mutual visible-surfaces" of two moving objects from a query time. "Mutual visible surfaces" are subregions which are "visible" each other. We give algorithms for constructing an MVI-tree and an MVI-list from a set $\mathcal{M}$ of two convex polygons M_0 with n_0 vertices and M_1 with n_1 vertices, in the case where every convex polygon moves by uniform motion. An MVI-tree of $\mathcal{M}$ can be constructed in time $O(N \log N)$ using space $O(N)$ and an MVI-list of $\mathcal{M}$ can be constructed in time $O(N)$ using space $O(N)$, where N is the total number of vertices and $N = n_0 + n_1$. "Mutual visible intervals", *i.e.* "one-dimensional mutual visible surfaces", of $\mathcal{M}$ at a query time can be found in time $O(\log N)$ using an MVI-tree or an MVI-list.
keyword Spatio-Temporal Index, Visible Surface Determination, Mutual Visible Intervals, Moving Object, MVI-tree, MVI-list

1 Introduction

Given a set of geometrical objects, such as line segments,polytopes,$\cdots$, and a view point, *visible surfaces* are parts of the given objects which are "visible" from the given view point [7]. The problems of determining the visible surfaces have been mainly studied in the fields of computational geometry and/or computer graphics [2, 4, 6, 10, 11, 13]. One can observe that these problems are equivalent to determine the "hidden surfaces" of objects, which are not visible from the given view point. Especially, these problems are significant for deleting "hidden surfaces" in the field of computer graphics. However, it has been increasing in the fields other than computer graphics to deal with such problems of determining "visible surfaces" or "hidden surfaces." Thus, it is desired to develop the methods for managing such visible information. For example, Notoya *et. al.* propose a method for searching all "visible objects" from many geometrical objects [10]. On the other hand, recently, much interest is attracted to methods for dealing with "moving geometrical objects", which change those geometrical information, such as shapes, positions, and so on, depending on the time. We call such objects *moving objects*. There are many study for storing or retrieving such moving objects [1, 9, 12, 14]. For such problems, a representative method is computing a "scene" for every "frame", where a scene is a geometrical configuration including coordinates and a frame is a discrete time stamp in the modeling world[7]. Recently, Tao *et. al.* give an efficient method for storing and finding positions of objects from a query time if the trajectories of moving objects are known[14]. Patel *et. al.* also give another efficient method for solving a similar problem[12].

In this paper, we will present algorithms for constructing two data structures, if the trajectories of moving objects are given. We assume that two moving objects are convex polygons in the plane $\mathbb{R}^2$ and each objects moves by uniform motion, that is, moves along a straight line with a fixed velocity. We also assume that each object does not transform its shape and does not rotate. For such situation, we

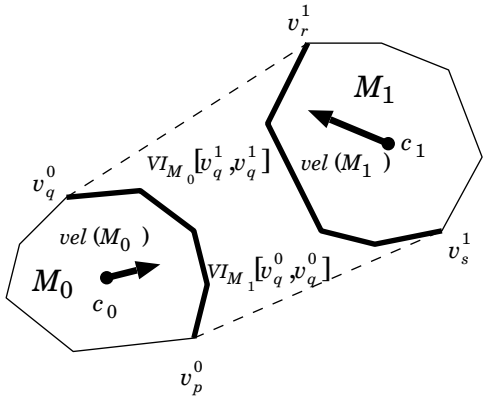

Figure 1: The mutual visible-intervals of two moving objects.

give two methods for searching a pair of one dimensional mutual visible surfaces, called *mutual visible intervals*, whose points are visible each other and which are two subregions of boundary of convex polygons. Figure 1 illustrates mutual visible intervals. We present two indices (data structures), one is called an *mutual visible intervals search tree* or an *MVI-tree* and the other is called an *mutual visible intervals search list* or an *MVI-list*. One can efficiently find mutual visible intervals using an MVI-tree or an MVI-list.

The remainder of this paper is organized as follows. In Section 2, we give some preliminary definitions. In Section 3 we show some properties of mutual visible intervals. In Section 4, we give an indexing method using an MVI-tree. In Section 5, we present an indexing method using an MVI-list. Finally, we conclude in Section 6.

2 Preliminaries

In this section, we define terms and notations, and formally describe our problem. We first give a static model, and then extend it to a kinetic model.

2.1 Static Model

Let $\mathcal{M} = \{M_0, M_1\}$ be two convex polygons representing two moving objects. For each $i \in \{0, 1\}$, moving object M_i has n_i vertices and is represented by the sequence of vertex list $(v_0^i, v_1^i, \cdots, v_{n_i-1}^i)$ counterclockwise, where v_j^i, $0 \leq j < n_i$, is a vertex of M_i. Each edge of M_i is denoted by $e_j^i = (v_j^i, v_{j+1}^i)$, $0 \leq j < n_i$. We may simply denote by i for $i \pmod 2$, and by j for $j \pmod{n_i}$. We denote the total number of vertices of $\mathcal{M}$ by N, that is $N = n_0 + n_1$. We assume that every edge e_j^i of M_i is not horizontal, that is, is not parallel to the x-axis. Each M_i has a *representative point* $c_i \in M_i$ and a fixed *velocity* $vel(M_i)$. We denote the x-, y-coordinate of a point $p \in \mathbb{R}^2$ by $x(p)$, $y(p)$,respectively, and also denote the x-,y-component of a 2-dimensional vector v by $x(v)$,$y(v)$, respectively.

For a convex polygon M, the boundary of M is denoted by $B(M)$, and the proper interior of M is denoted by $I(M)$, that is $I(M) = M - B(M)$. A point $q \in M$ is *visible* from an exterior point $p \in \mathbb{R}^2 - M$ if the line segment $\overline{pq}$ does not intersect $I(B)$, that is $\overline{pq} \cap I(M) = \emptyset$. One can easily observe that every visible point $q \in M$ is on the boundary $B(M)$ of M. The *visible region from a point* p is a maximal subregion of $B(M)$ which is visible from a point p, and denoted by $V_p(M)$. Two points $p_0 \in M_0$ and $p_1 \in M_1$ are *mutual visible* if the line segment $\overline{p_0 p_1}$ does not intersect both interiors of convex

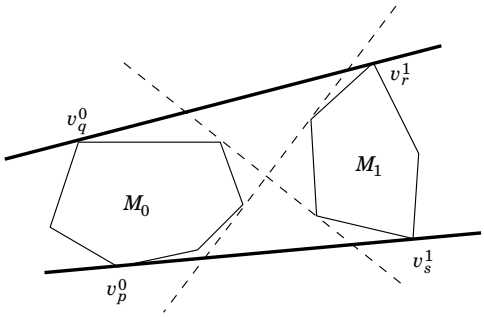

Figure 2: Four common tangents of two convex polygons.

polygons, that is $\overline{p_0p_1} \cap (I(M_0) \cup I(M_1)) = \emptyset$. A *visible region* or *visible interval of* M_0 *from* M_1 is a maximal subregion of M_0 whose point is visible from some point $p_1 \in M_1$, and denoted by $V_{M_1}(M_0)$. One can easily observe that a visible interval $V_{M_1}(M_0)$ is a continuous path $P = (v_p^0, v_{p+1}^0, \cdots, v_q^0)$ of $B(M_0)$. Thus, we may also denote $V_{M_1}(M_0)$ by $VI_{M_1}[v_p^0, v_q^0]$. Similarly, we denote a visible interval of M_1 from M_0 by $V_{M_0}(M_1)$ or $VI_{M_0}[v_r^1, v_s^1]$. A pair $(VI_{M_1}[v_p^0, v_q^0], VI_{M_0}[v_r^1, v_s^1])$ is called by *mutual visible intervals* of $\mathcal{M}$=$\{M_0, M_1\}$ and denoted by $MVI(\ \mathcal{M}\ ) = (p, q, r, s)$ for short, where $p, q \in \{0, \cdots, n_0 - 1\}$ and $r, s \in \{0, \cdots, n_1 - 1\}$. The *mutual visible intervals determination* is a problem for finding $MVI(\ \mathcal{M}\ ) = (p, q, r, s)$ from a given non-intersecting convex polygons $\mathcal{M} = \{M_0, M_1\}$.

This problem can be solved by finding "common tangent lines." A line l is called a *tangent line* of M if M lies on the one side of l and l passes through a vertex of M, called a *tangent vertex*, or an edge of M, called a *tangent edge*. A line l is called a *common tangent line* of $\mathcal{M}$=$\{M_0, M_1\}$ if l is a tangent line of both M_0 and M_1. A common tangent line l of $\mathcal{M}$=$\{M_0, M_1\}$ is called *iso- common tangent* if both M_0 and M_1 are contained in the same half plane defined by l, otherwise l is called *hetero common tangent*. Generally, there exist four common tangent lines, two of them are iso common tangent lines, and other two are hetero common tangent lines. In Figure 2, iso-common tangent lines drawn by solid lines, hetero-common tangent lines drawn by dashed lines. Using an ordinary method of computational geometry, one can find these common tangent lines in time $O(N)$, thus one can find mutual visible intervals $MVI(\ \mathcal{M}\ )$ in linear time [6, 11, 13].

2.2 Kinetic Model

Let $T = (-\infty, \infty) \subset \mathbb{R}$ be a set of time. A moving object o can be regarded as a subregion of $\mathbb{R}^2 \times T$, and hence we call the space $\mathbb{R}^2 \times T$ the *kinetic space* $\mathcal{K}$. A *trajectory* of o is a subregion of $\mathcal{K}$ taken by o, where moving objects o may be points, line segments, lines,or polygons. For a time $t_f \in T$, a *scene* of t_f is a intersection of the kinetic space $\mathcal{K}$ and a plane of $t = t_f$, and is represented by a map $SC : T \to 2^{\mathbb{R}^2}$. For every objects $o \subseteq \mathcal{K}$, the configuration of o in scene $SC(t)$ is denoted by $o(t)$. For example, we denote a convex polygon M at a time $t \in T$ by $M_i(t) = (v_0^i(t), \ldots, v_{n_i-1}^i(t))$. We assume $M_0(t) \cap M_1(t) = \emptyset$ for every time $t \in T$. An *initial configuration* is a scene $SC(0)$ at time $t = 0$. Given an initial configuration $SC(0)$ containing $\mathcal{M}\ (0) = \{M_0(0), M_1(0)\}$, two velocities $vel(M_0)$,$vel(M_1)$, and a time $t \in T$, a *determining kinetic MVI problem* of $\mathcal{M}\ (t) = \{M_0(t), M_1(t)\}$ is to determine kinetic mutual visible intervals $(VI_{M_1(t)}[v_p^0(t), v_q^0(t)], VI_{M_0(t)}[v_r^1(t), v_s^1(t)])$. We also denote kinetic mutual visible intervals by $MVI(\ \mathcal{M}\ (t)) = (p, q, r, s)$ or $MVI(t) = (p, q, r, s)$ for short.

One can solve this kinetic MVI determination problem using a straightforward algorithm as follows. It first computes the scene $SC(t)$ of $t \in T$, then finds two iso-common tangent lines of $M_0(t)$ and $M_1(t)$, finally finds kinetic mutual visible intervals $MVI(t)$. Each $M_i(t)$ can clearly be computed from

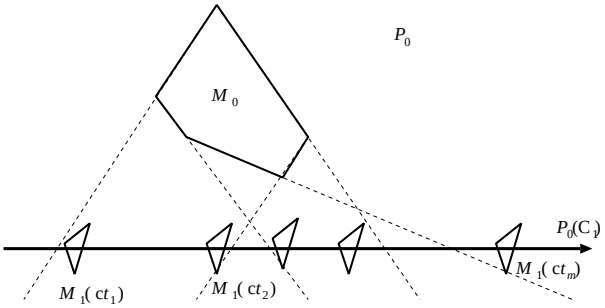

Figure 3: The projected space P_0 of moving object M_o.

$M_i(0)$ and $vel(M_i)$ in time $O(n_i)$, and hence a scene $SC(t)$ can be computed in time $O(N)$. Two iso-common tangent lines can also be found in time $O(N)$, as mentioned before[6, 11, 13]. Thus,this algorithm would take time $O(N)$. However, this time complexity is not optimal for the case where many mutual visible intervals should be found for many time stamps. Let $T_F = (t_1, t_2, \cdots, t_F) \subset T$ be a set of discrete time stamps. Given a initial configuration $SC(0)$, two velocities $vel(M_0)$,$vel(M_1)$, and a set $T_F = (t_1, t_2, \cdots, t_F)$ of discrete times, a *searching kinetic MVI problem* of $\mathcal{M}\ (t)$ is to find the sequence $(MVI(t_1), MVI(t_2), \cdots, MVI(t_F))$ of mutual visible intervals. This searching problem can be also solved by using the above ordinary algorithm repeatedly, however such algorithm would take time $O(FN)$. On the other hand, we present faster methods to solve a searching problem. We propose two data structures, called an MVI-tree and an MVI-list. We give an algorithm for constructing an MVI-tree which runs in time $O(N \log N)$. Each $MVI(t_f)$ can be found in time $O(\log N)$ for each query time $t_f \in T_F$ using an MVI-tree. Thus, a searching MVI problem can be solved in time $O((N + F) \log N)$ using an MVI-tree. We also give an algorithm for constructing an MVI-list which runs in time $O(N)$. Each $MVI(t_f)$ can be found in time $O(\log N)$ for each query time $t_f \in T_F$ using an MVI-list. Thus, the searching MVI problem can be solved in time $O(N + F \log N)$ using an MVI-list.

3 Properties of Mutual Visible Intervals

In this section, we show some properties of mutual visible intervals for constructing data structures.

We may assume that one of moving objects $M_i(t) \in \mathcal{M}\ (t)$ has a zero vector $\mathbf{0}$ as a velocity $vel(M_i)$. If both objects $M_i(t)$ do not have zero vectors, we transform $\mathcal{M}\ (t) = \{M_0(t), M_1(t)\}$ to $\mathcal{M}\ '(t) = \{M_0'(t), M_1'(t)\}$ which are defined by $M_0'(0) = M_0(0)$, $M_1'(0) = M_1(0)$, $vel(M_0') = vel(M_0) - vel(M_0) = \mathbf{0}$, and $vel(M_1') = vel(M_1) - vel(M_0)$. It can be easily observed that each mutual visible intervals $MVI(\mathcal{M}\ (t))$ has the same vertex set of $MVI(\mathcal{M}\ '(t))$ in all time $t \in T$. This transformation above is corresponding to shear the kinetic space $\mathcal{K}$ according $-vel(M_0)$. We call such a space a *static space of* $M_0(t)$, and denote it by S_0. A *static space* S_1 *of* $M_1(t)$ is similarly defined. In each static space S_i of $M_i(t)$, the objects $M_i(t)$ is *static*, and hence is denoted by M_i. Without loss of generality, we assume that $y(vel(M_i)) = 0$, $x(vel(M_i)) = 0$, $y(vel(M_{i+1})) = 0$, and $x(vel(M_{i+1})) \geq 0$. A *projected space* P_i *of* $M_i(t)$ is two dimensional plane which is obtained by projecting the 3-dimensional static space S_i of $M_i(t)$ on the plane of $t = 0$. Figure 3 illustrates a projected space P_0 of pentagon M_0. We denote by C_i the trajectory of the representative point c_i of M_i in the kinetic space $\mathcal{K}$, by $P_i(C_{i+1})$ the image of the trajectory C_{i+1} in the projected space P_i. We assume that $P_i(C_{i+1})$ is horizontal and $y(c_i) > y(c_{i+1}(t))$. In the projected space P_i, a line extended from each edge e_j^i of M_i is called *extension* and denoted by $l(e_j^i)$ or l_j^i. Let $L^i = \{l_j^i | 0 \leq j < n_i\}$. Then, every extension $l(e_j^i) \in L^i$ is not horizontal

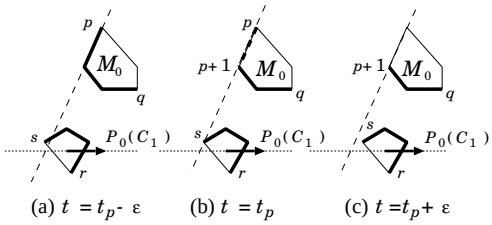

Figure 4: Decreasing the visible interval of M_0 around the changing time t_p.

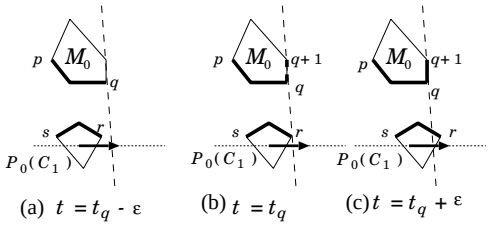

Figure 5: Increasing the visible interval of M_0 around the changing time t_q.

since e_j^i is not horizontal. For a non-horizontal line l, a *left (right) plane of* l is the half plane divided by l and has a point $p \in \mathbb{R}^2$ having y-coordinate $y(p) = 0$ and x-coordinate $x(p) = -\infty$ ($x(p) = \infty$). A polygon M is on *the left(right) of a line* l if M is contained on the left (right) plane of l. A non-horizontal line l is on *the left(right) of polygon* M if M is on the right (left) of l. Then, the following lemma holds.

Lemma 1 *For any two times* $t, t' \in T$, $(t < t')$,

$$MVI(t) \neq MVI(t')$$

if and only if the one of following conditions satisfies:

(p) *There exists a time* t_p,$t \leq t_p < t'$, *when an extension* $l(e_j^0) \in L^0$ *becomes an iso-common tangent on the left of both* M_0 *and* $M_1(t_p)$ *in the projected space* P_0*;*

(q) *There exists a time* t_q,$t < t_q \leq t'$, *when an extension* $l(e_j^0) \in L^0$ *becomes an iso-common tangent on the right of both* M_0 *and* $M_1(t_q)$ *in the projected space* P_0*;*

(r) *There exists a time* t_r,$t \leq t_r < t'$, *when an extension* $l(e_j^1) \in L^1$ *becomes an iso-common tangent on the left of both* $M_0(t_r)$ *and* M_1 *in the projected space* P_1*;*

(s) *There exists a time* t_s,$t < t_s \leq t'$. *when an extension* $l(e_j^1) \in L^1$ *becomes an iso-common tangent on the left of* $M_0(t_s)$ *and* M_1 *in the projected space* P_1.

Proof One can easily observe that mutual visible intervals $MVI(t)$ changes only if an extension becomes an iso-common tangent of M_i and $M_{i+1}(t)$. Thus, we only show the sufficiency in below.
(p)**:** Let $l(e_j^0)$ be an extension of e_j^0 satisfying (p) at time t_p, and $MVI(t_p) = (p, q, r, s)$ be the corresponding mutual visible intervals. See Figure 4. Then, $p = j$ and $MVI(t_p) = (j, q, r, s)$ since $e_j^0 = (v_j^0, v_{j+1}^0)$. For $t \in T$, let $l_p(t)$ be the left iso-common tangent of M_0 and $M_1(t)$. Then, for a small real number $\varepsilon > 0$, $l_p(t_p - \varepsilon)$ touches v_j^0 at time $t_p - \varepsilon$, and hence $MVI(t_p - \varepsilon) = (j, q, r, s)$. On the

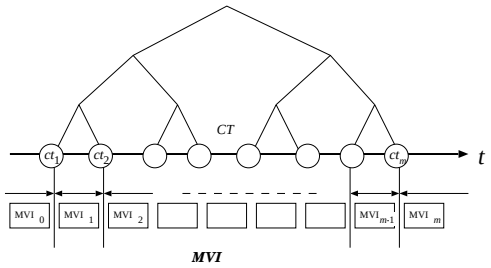

Figure 6: An MVI-tree stores CT and $\mathcal{MVI}$.

other hand, the $l_p(t_p + \varepsilon)$ touches another vertex v^0_{j+1} of e^0_j at time $t_p + \varepsilon$, and hence $MVI(t_p + \varepsilon) = (j + 1, q, r, s)$. Thus, the visible interval $VI_{M_1}[v^0_p, v^0_q]$ decrease and turns into $VI_{M_1}[v^0_{p+1}, v^0_q]$.
(q)**:** Let $l(e^0_j)$ be an extension of e^0_j satisfying (q) at time t_q, and $MVI(t_q) = (p, q, r, s)$ be the corresponding mutual visible intervals. See Figure 5. Then, $q = j + 1$ and $MVI(t_q) = (p, j + 1, r, s)$ since $e^0_j = (v^0_j, v^0_{j+1})$. For $t \in T$, let $l_q(t)$ be the right iso-common tangent of M_0 and $M_1(t)$. Then, for a small real number $\varepsilon > 0$, $l_q(t_q - \varepsilon)$ touches v^0_j at time $t_q - \varepsilon$, and hence $MVI(t_q - \varepsilon) = (p, j, r, s)$. On the other hand, the $l_q(t_q + \varepsilon)$ touches another vertex v^0_{j+1} of e^0_j at time $t_p + \varepsilon$, and hence $MVI(t_p + \varepsilon) = (p, j + 1, r, s)$. Thus, the visible interval $VI_{M_1}[v^0_p, v^0_q]$ increases and turns into $VI_{M_1}[v^0_p, v^0_{q+1}]$.
(r)**:** The proof of this case is similar as the case (p). The r changes and the interval VI_{M_0} decrease if this case is occur.
(s)**:** The proof of this case is similar as the case (s). The s changes and interval VI_{M_0} increase if this case is occur. □

This lemma 1 implies that the sort of mutual visible intervals are discrete and finite. The *changing time* is the time when mutual visible intervals change. We denote by $CT \subset T$ the set of all of changing times. Let CT_p, CT_q,CT_r,CT_s be the sets of changing times satisfying the condition of lemma $1(p)$, (q), (r), (s),respectively, and let $CT^0 = CT_p \cup CT_q$, $CT^1 = CT_r \cup CT_s$. Note that $CT = CT^0 \cup CT^1 = CT_p \cup CT_q \cup CT_r \cup CT_s$. We assign the total order to CT according to the natural order of time. The ordered set CT is denoted by $(ct_1, ct_2, \cdots, ct_m)$, where m is the number of changing times. For every k,$1 \leq k < m$, mutual visible intervals $MVI(t)$ is the same during $ct_k \leq t < ct_{k+1}$. Let MVI_k be the mutual visible intervals $MVI(t)$ during $ct_k \leq t < ct_{k+1}$, let MVI_0 be during $t < ct_1$, and let MVI_m during $ct_m \leq t$. We denote the ordered set of mutual visible intervals by $\mathcal{MVI} = (MVI_0, MVI_1, \cdots, MVI_m)$.

4 MVI-tree

An *mutual visible intervals search tree* (MVI-tree) is a tree like data structures, which stores each mutual visible intervals $MVI_k \in \mathcal{MVI}$ in leaves, stores each changing time $ct_k \in CT$ in internal nodes, and enable to search $MVI(t)$ for a query time $t \in T$. The structure of internal nodes is called by *an index part* of an MVI-tree, and can be constructed as a balanced tree such as a red-black tree[3]. Thus, every $MVI(t)$ can be found in time $O(\log N)$ using an MVI-tree. Figure 6 illustrates an MVI-tree.

Now,we give an algorithm **Construct MVI-tree** for constructing an MVI-tree as follows.

Algorithm 1 ***Construct MVI-tree***

(T1) *Find $CT^0 = CT_p \cup CT_q$ in the projected space P_0 of M_0;*

(T2) *Find $CT^1 = CT_r \cup CT_s$ in the projected space P_1 of M_1;*

(T3) *Find CT by merging CT^0 and CT^1;*

(T4) *Calculate the initial mutual visible intervals $MVI_{k_0} = MVI(0)$ from the initial configuration $\mathcal{M}(0) = \{M_0(0), M_1(0)\}$;*

(T5) *Calculate mutual visible intervals $MVI_k \in \mathcal{MVI}$ from MVI_{k-1} or MVI_{k+1} for every k,$1 \leq k < m$;*

(T6) *Construct MVI-tree from CT and $\mathcal{MVI}$.*

In below, we explain the detail of **Construct MVI-tree**.

4.1 Tangent Point search tree

Each CT^i can be found by detecting tangent points v_k^{i+1} of $M_{i+1}(t)$ passed by the extension $l_j^i \in L^i$. Using this property, a straightforward algorithm is obtained as follows. For each extension $l_j^i \in L^i$ of M_i, it finds tangent point v_k^{i+1} on $M_{i+1}(t)$ by checking every vertex of $M_{i+1}(t)$, and would take time $O(n_{i+1})$. Thus, it would take time $O(n_0 n_1) = O(N^2)$ for finding CT^i.

Our first idea is to construct an intermediate date structure called a *tangent point search tree* (TP-tree) since many tangent points should be found. We denote a tangent point search tree of $M_{i+1}(t)$ by $TP(M_{i+1}(t))$. For two points $p_1, p_2 \in \mathbb{R}^2$, the *angle* $\theta(\overrightarrow{p_1p_2})$ of a vector $\overrightarrow{p_1p_2}$ is measured counterclockwise at a point p_1 from the $+x$-direction and ranges in $[0, 2\pi)$. We may omit $\pmod{2\pi}$ since every angle is ranges in $[0, 2\pi)$. A *normal vector* $\mathbf{n}(e_j^{i+1})$ of $e_j^{i+1} = (v_j^{i+1}, v_{j+1}^{i+1})$ is a unit vector satisfying $\theta(\mathbf{n}(e_j^{i+1})) = \theta(\overrightarrow{v_j^{i+1}v_{j+1}^{i+1}}) - \frac{\pi}{2}$. Similarly, a *normal vector* $\mathbf{n}(l)$ of a line l is defined by two points $p_1, p_2 \in l$. For each e_j^{i+1}, a normal vector $\mathbf{n}(e_j^{i+1})$ is perpendicular to an ordered edge $\overrightarrow{v_j^{i+1}v_{j+1}^{i+1}}$ and points outside of $M_{i+1}(t)$. We assume, with out loss of generality, that the angle $\theta(\mathbf{n}(e_0^{i+1}))$ of edge e_0^{i+1} is the minimum during all normal vectors of edges e_j^{i+1}. Then, the following lemmas obviously holds.

Lemma 2 *For each edge $e_j^{i+1}$of $M_{i+1}(t)$, the following inequality satisfies:*

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

Lemma 3 *In projected space P_i, a non-horizontal line l passing through a vertex v_k^{i+1} of $M_{i+1}(t)$ is a tangent line of $M_{i+1}(t)$ if and only if either (i) or (ii) holds:*

(i)

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

(ii)

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

.

By Lemma2, we can assign the total order to the set of edges e_j^{i+1} of $M_{i+1}(t)$ using the angle $\theta(\mathbf{n}(e_j^{i+1}))$. Thus, the tangent point search tree $TP(M_{i+1}(t))$ stores all edges e_j^{i+1} of $M_{i+1}(t)$ according to the total order of the angles. We insert every edge e_j^{i+1}, $0 \leq j < n_{i+1}$, to $TP(M_{i+1}(t))$ by calculating the angle $\theta(e_j^{i+1})$ as key. We construct $TP(M_{i+1}(t))$ as a balanced tree. Using $TP(M_{i+1}(t))$, we can find a tangent point v_k^{i+1} touched by extension l_j^i of M_i in time $O(\log n_{i+1})$.

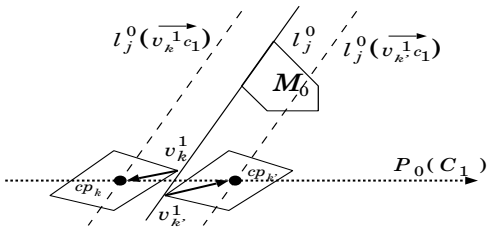

Figure 7: The relationship between an extension and a changing time.

4.2 The detail of Construct MVI-tree

We first show the detail of steps (T1)-(T3). We denote by $l(\overrightarrow{p_1p_2})$ the line translated from l according to $\overrightarrow{p_1p_2}$. Then, the following algorithm **Find** CT^i finds CT^i.

Algorithm 2 ***Find*** CT^i

(CT1) *Construct a tangent point search tree* $TP(M_{i+1}(t))$ *of* $M_{i+1}(t)$*;*

(CT2) *For each edge* e_j^i *of* M_i *and its extension* $l_j^i \in L^i$*, execute the following (CT3)-(CT5);*

(CT3) *Search on* $TP(M_{i+1}(t))$ *as keys* $\theta(\mathbf{n}(l_j^{i+1}))$ *and* $\theta(\mathbf{n}(l_j^{i+1})) + \pi$*, and find tangent points* v_k^{i+1} *and* $v_{k'}^{i+1}$*, each of which is passed by a tangent line parallel to* l_j^i*;*

(CT4) *Calculate the crossing points* $cp_k = l_j^i(\overrightarrow{v_k^{i+1}c_{i+1}}) \cap P_i(C_{i+1})$ *and* $cp_{k'} = l_j^i(\overrightarrow{v_{k'}^{i+1}c_{i+1}}) \cap P_i(C_{i+1})$*. (See Figure 7.) Determine which crossing points above (*cp_k *or* $cp_{k'}$*) lies on the same side of* l_j^i *as the representative point* c_i*, and let* cp^* *be such crossing point;*

(CT5) *Calculate the changing time* ct *when the representative point* $c_{i+1}(t)$ *reaches the cross point* cp^**, then insert* ct *to* CT^i*.*

For M_i and $M_{i+1}(t)$, there exist both iso-common tangent and hetero-common tangent. Therefore, for each extension l_j^i, two tangent points are found v_k^{i+1}, $v_{k'}^{i+1}$ by (CT3), one of which is the tangent point of iso-common tangent and the other is one of hetero-common tangent. However, from the configuration of the scene, we can determine which tangent point is one of the iso-common tangent. These determination is done in (CT4). Figure 7 illustrates the relationship between an extension and a changing time. The set of changing time $CT^0 = CT_p \cup CT_q = (ct_1^0, \cdots, ct_{m_0}^0)$ can be found by executing **Find** CT^0 in P_0. Similarly, $CT^1 = CT_r \cup CT_s = (ct_1^1, \cdots, ct_{m_1}^1)$ can be found by executing **Find** CT^1 in P_1. Each of these sets has the total order, and hence the total ordered set $CT = (ct_1, \cdots, ct_m) = CT_0 \cup CT_1$ is obtained by merging CT^0 and CT^1, where $m_0 = |CT^0|$, $m_1 = |CT^1|$, $m = |CT|(= m_0 + m_1)$. This merging step can be done in time $O(|CT|) = O(N)$ [3].

In step (T4), iso-common tangents of $M_0(0)$, $M_1(0)$ can be found by typical geometric algorithm [6, 11, 13], and hence the initial mutual visible intervals $MVI(0)$ can be also found in time $O(N)$.

Finally, we give the detail of steps (T5),(T6). Each changing time $ct_k \in CT = (ct_1, \cdots, ct_m)$ is contained in the one of four subsets CT_p, CT_q, CT_r, CT_s. These assignment can be found when **Find** CT^i is executed. Thus, by the proof of Lemma 1, each mutual visible intervals MVI_k can be calculating from MVI_{k-1} or MVI_{k+1}. For example, if $ct_k \in CT_p$ then $MVI_k = MVI(ct_k + \varepsilon) = (p+1, q, r, s)$ and $MVI_{k-1} = MVI(ct_k - \varepsilon) = (p, q, r, s)$. Note that $MVI_{k_0} = MVI(0)$, and hence the following inequality is satisfied: $ct_{k_0} \leq 0 \leq ct_{k_0+1}$. For any positive changing times $ct_k > ct_{k_0}$,

we can calculate every mutual visible intervals $MVI(ct_k)$ from MVI_{k-1}, and hence we can obtain mutual visible intervals for positive changing times from $MVI_{k_0} = MVI(0)$ to $MVI_m = MVI(\infty)$. Similarly, we can obtain mutual visible intervals for negative changing times from $MVI_{k_0} = MVI(0)$ to $MVI_0 = MVI(-\infty)$. Thus, we can obtain $\mathcal{MVI} = (MVI_0, \cdots, MVI_{k_0}, \cdots, MVI_m)$, where $MVI(-\infty) = MVI_0$ $MVI_{k_0} = MVI(0)$, and $MVI_m = MVI(\infty)$). The MVI-tree can be construct by inserting every $ct \in CT$ as internal nodes and every $MVI_k \in \mathcal{MVI}$ as leaf nodes.

4.3 Complexities

In this section, we analyze the complexity of the algorithm **Construct MVI-tree**.

We first analyze the complexity of the algorithm **Find** CT^i. A balanced tree with m nodes can be constructed in time $O(m \log m)$ by inserting every nodes to the empty tree, and it has height at most $O(\log m)$ [3]. Thus, a tangent point search tree $TP(M_{i+1}(t))$ can be constructed in time $O(n_{i+1} \log n_{i+1})$ in (CT1). In (CT2)-(CT5), every tangent points can be found in time $O(\log n_{i+1})$ since $TP(M_{i+1}(t))$ has height at most $O(\log n_{i+1})$. Therefore, all tangent points of CT^i can be found in time $O(n_i \log n_{i+1})$ since $|L^i| = n_i$. Therefore, the following lemma holds.

Lemma 4 *The algorithm **Find** CT^i runs in time $O(N \log N)$ using space $O(N)$.*

We next analyze the complexity of **Construct MVI-tree**. The following lemma holds for the set CT of changing time.

Lemma 5 *The set CT has at most $O(N)$ elements.*

Proof In projected space P_i, each extension $l^i_j \in L^i$ becomes an iso-common tangent of M_i and $M_{i+1}(t)$ exactly once, and hence $|CT^i| = |L^i| = n_i$. Thus, $|CT| \leq |CT^0| + |CT^1| = n_0 + n_1 = N$. □

By lemma 4 and lemma5, each set of changing time CT^0 and CT^1 is found in time $O(N \log N)$ in (T1),(T2). The merging operation take linear time[3], and hence CT can be found in time $O(|CT|) = O(N)$ in (T3). The initial mutual visible intervals $MVI(0)$ is calculated in time $O(N)$ in (T4) using typical geometrical algorithm[6, 11, 13]. The set $\mathcal{MVI}$ of mutual visible intervals can be found from CT and $MVI(0)$ in time $O(N)$ in (T5). Thus, an MVI-tree can be constructed in time in time $O(N \log N)$ and has height at most $O(\log N)$ since it is a balanced tree [3]. Therefore, the following theorem and corollary holds.

Theorem 1 *The algorithm **Construct MVI-tree** constructs an MVI-tree in time $O(N \log N)$ using space $O(N)$. For any query time $t \in T$, mutual visible intervals $MVI(t)$ can be found in time $O(\log N)$ using the MVI-tree.*

Corollary 1 *Using an MVI-tree, a searching problem of moving objects $\mathcal{M}$ (t) and the time set T_F can be solved in time $O((F + N) \log N)$ using $O(N)$ space, where $N = n_0 + n_1$ and $F = |T_F|$.*

5 MVI-list

An *mutual visible search list* is a sequential data structures which stores pairs of changing time $ct_k \in CT$ and mutual visible intervals $MVI_k \in \mathcal{MVI}$. Each elements (ct_k, MVI_k) of an MVI-list is stored according to the total order of the changing time. Figure 8 illustrates an MVI-list. $MVI(t)$ for a query time t can be found by using "binary search algorithm" on the MVI-list, since an MVI-list stores each element by the increasing order of time [3].

Now, we give an algorithm **Construct MVI-list** for constructing MVI-list as follows.

0	1	- - -	k	- - -	m
(-∞ ,MVI_0)	(ct_1,MVI)	- - -	(ct_1 ,MVI_k)	- - -	(∞ ,MVI_m)

Figure 8: An MVI-list.

Algorithm 3 ***Construct MVI-list***

(L1) *For every $u \in \{p, q, r, s\}$, construct u-list;*

(L2) *Construct MVI-list by merging p- q- r- s-lists.*

The main idea of this algorithm is to find every subsets CT_u, $u \in \{p, q, r, s\}$, of changing time CT separately. In below, we show only the algorithm for constructing p-list since the other three lists can be similarly found.

5.1 Constructing p-list

We first show additional lemmas for constructing p-list.

Lemma 6 *Let v_a^0 be the vertex of M_0 with maximum y-coordinate, and v_b^0 be the vertex of M_0 with minimum x-coordinate. Then, $MVI(-\infty) = (a, q, r, s)$, $MVI(\infty) = (b, q, r, s)$, and every vertex v_p^0 is contained in the path from v_a^0 to v_b^0 on $B(M_0)$ during $t \in T = (-\infty, \infty)$.*

Proof One can easily observe that the angle of the left iso-common tangent line ranges in $(\pi, 2\pi)$ since M_0 lies above $P_0(C_1)$. Furthermore, obviously, the vertex v_a^0 is passed by the tangent line l_a^0 with the angle $\theta(l_a^0) = \pi$ when $t = -\infty$, and hence v_a^0 is one of the terminal of mutual visible intervals $MVI(-\infty)$. Similarly, the vertex v_b^0 is passed by the tangent line l_b^0 with the angle $\theta(l_b^0) = 2\pi$ when $t = \infty$. Moreover, every v_p^0 clearly appears between v_a^0 and v_b^0. Thus, this lemma holds. □

For each v_p^0 and $e_p^0 = (v_p^0, v_{p+1}^0)$, $a \leq p \leq b$, let $ct(v_p^0)$ be the changing time when the extension $l(e_p^0) \in L^0$ becomes the left iso-common tangent of M_0 and $M_1(t)$. Then, the following lemma satisfies.

Lemma 7 *For each vertex v_p^0 in the path $(v_a^0, \cdots, v_p^0, \cdots, v_b^0)$, each changing times $ct(v_p^0)$ increase according to increasing index p from a to b, that is, the set $CT_p = (\ ct(v_a^0), \cdots, ct(v_p^0), \cdots, ct(v_b^0))$ has the natural order of time.*

Proof For each p, $a \leq p \leq b$, the angle $\theta(e_p^0)$ increase according to increasing index p since M_0 is a convex polygon. Therefore, each extension $l(e_p^0)$ intersects the horizontal line $P_0(C_1)$ by the increasing order of the x-coordinate. Moreover, one can easily observe that every changing time appears by the increasing order. See Figure 9. □

By Lemmas 6 and 7, we show the algorithm **Construct p-list** as follows.

Algorithm 4 ***Construct p-list***

(P1) *Find the vertex v_a^0 of M_0 with the maximum y-coordinate, and find the vertex v_b^0 of M_0 with the minimum y-coordinate;*

(P2) *For each p, $a \leq p \leq b$, repeat the following (P3)-(P5);*

(P3) *Find the tangent vertex v_s^1 of $M_1(t)$ touched by the extension $l(e_p^0)$ from left, then calculate the corresponding changing time $ct(v_p^0)$ from a left iso-common tangent line $v_p^0 v_s^1$;*

(P4) *Add the pair $(ct(v_p^0), v_p^0)$ to the last of p-list.*

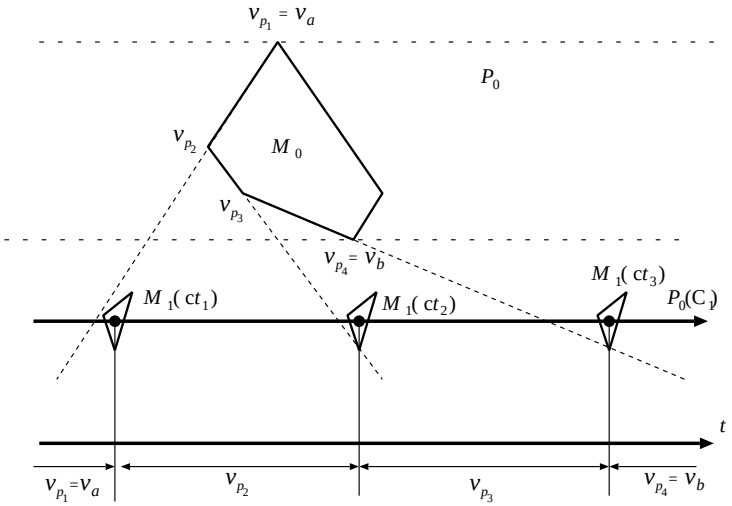

Figure 9: Constructing p-list.

5.2 Complexities

In this subsection, we analyze the complexity of the algorithm **Construct MVI-list**.

We first analyze the complexity of the algorithm **Construct p-list**. One can easily find the vertex with maximum or minimum y-coordinate in time $O(N)$, and hence (P1) can be executed in time $O(N)$. Every v_p^0 can be found by the previous edge $e_{p-1}^0 = (v_{p-1}^0, v_p^0)$ of M_0. Furthermore, one can easily observe that the tangent vertex v_s^1 of the left iso-common tangent line $v_p^0 v_s^1$ appears counterclockwise in the vertex list of $M_1(t)$. Thus, by Lemma 7, the changing time $ct(v_p^0)$ can be found in time $O(1)$ in (P3). Thus, (P2)-(P4) can be executed in linear time in total. Thus, the algorithm **Construct p-list** can be executed in time $O(N)$.

For the algorithm **Construct MVI-list**, (L1) can be done by executing **Construct u-list** for every $u \in \{p, q, r, s\}$, and hence would take time $O(N)$. In (L2), the four-way merging technique should be used, and this can be done in time $O(N)$ [3].

Thus, the following theorem and corollary holds.

Theorem 2 *The algorithm **Construct MVI-list** constructs an MVI-list in time $O(N)$ using space $O(N)$. For any query time $t \in T$, the mutual visible intervals $MVI(t)$ can be found in time $O(\log N)$ using the MVI-list.*

Corollary 2 *Using the MVI-list, a searching problem of moving objects $\mathcal{M}(t)$ and the time set T_F can be solved in time $O(N + F \log N)$ using $O(N)$ space, where $N = n_0 + n_1$ and $F = |T_F|$.*

6 Conclusion

In this paper, we give several efficient methods for searching the mutual visible intervals for the case where the trajectories of moving objects are known. We give two data structures, called an MVI-tree and an MVI-list, if two convex polygons M_0, M_1 moves by uniform motion. The algorithm **Construct MVI-tree** constructs an MVI-tree in time $O(N \log N)$ using space $O(N)$. Each mutual visible intervals of an query time t can be found in time $O(\log N)$ by using an MVI-tree, that is, by tracing a root-leaf path on the MVI-tree. The algorithm **Construct MVI-list** constructs an MVI-tree in linear time using

linear space. Each mutual visible intervals of an query time t can be found in time $O(\log N)$ using an MVI-list, that is by using the binary search algorithm on the MVI-list.

The following future works remain:

(1) develop methods for searching mutual visible surfaces of two convex polytopes in the space of three or more degree dimension,

(2) develop methods for searching mutual visible surfaces of moving objects which may moves by non-uniform motion, and

(3) develop methods pairs of mutual visible surfaces among three or more moving objects,. . . .

References

[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] R. Diestel,Graph Theory,Springer-Verlag,1997.

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

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

[9] Y.kusakari, Y.Sugimoto, J.Notoya, and M.Kasai, "A Method for Searching Mutual Visible-Intervals on Moving Objects," Proc. of DEWS 2006, 4B-oi3,(2006)(in Japanese).

[10] J.Notoya, Y.Sugimoto, Y.Kusakari, and M.Kasai, "Visibility Search for Spatial Database System," DBSJ Letters, **4**, 2, pp.9–12(2005).(in Japanese).

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

[12] 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).

[13] F.P. Preparata, M.I. Shamos, Computational Geometry:An Introduction, Springer-Verlag, 1985.

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