

**Modeling and Querying Current Movement:
The Uncertainty of the Trajectory of a Moving Object**

The Uncertainty of the Trajectory of a Moving Object

Consider a motion plan for a moving point object together with an uncertainty threshold.

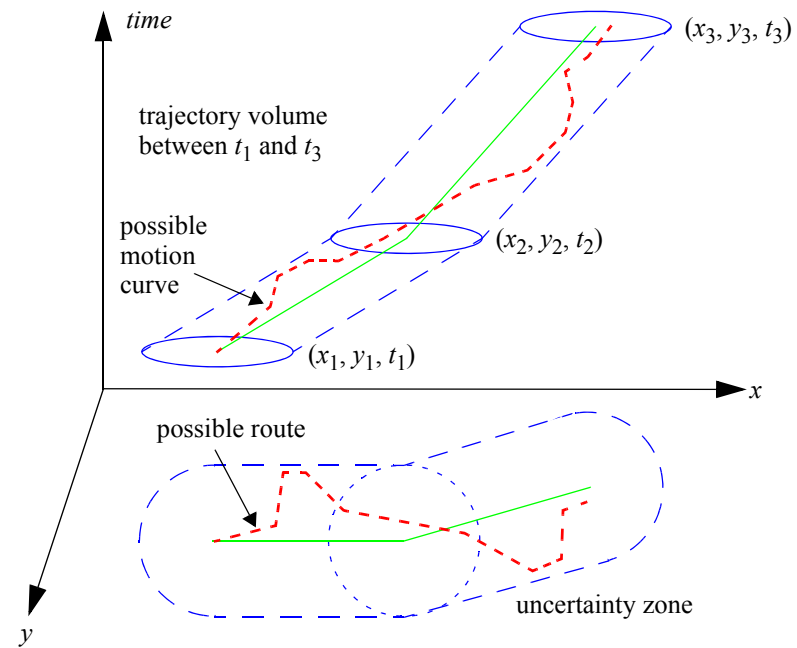
Motion plan called a **trajectory** here.

Will allow us to ask queries:

- Retrieve the current location of the trucks that will *possibly* be inside a region R , *sometime* between 1:00 pm and 1:10 pm.
- Retrieve the number of planes which will *definitely* be inside the region R , *sometime* between 4:30 pm and 4:45 pm.
- Retrieve the police cars which will *possibly* be inside the region R , *always* between 9:20 am and 9:50 am

A Model of a Trajectory

Trajectory together with uncertainty threshold \rightarrow a cylindrical volume in 3D (2D + time) space



Querying Moving Objects with Uncertainty

Interesting operations for querying moving objects described by uncertain trajectories?

Point Queries

$\underline{trajectory} \times \underline{instant}$	$\rightarrow \underline{point}$	whereAt
$\underline{trajectory} \times \underline{point}$	$\rightarrow \underline{set(instant)}$	whenAt

Semantics:

- **whereAt**(tr, i) returns expected position at instant i
- **whenAt**(tr, p): return the instants when the expected position is p

Spatio-Temporal Range Queries

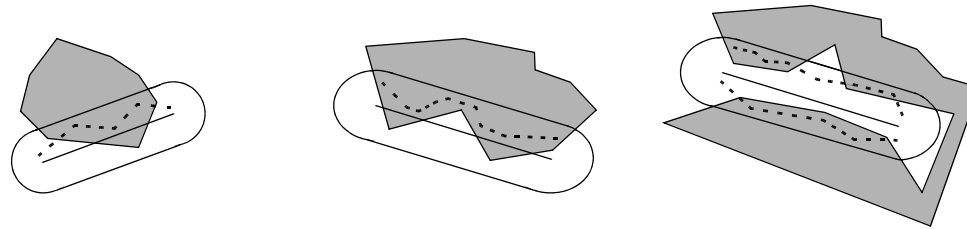
$\text{trajectory} \times \text{region} \times \text{instant} \times \text{instant} \rightarrow \text{bool} \quad \langle \text{inside} \rangle$

$\langle \text{inside} \rangle(ut, r, t_1, t_2)$ satisfied if moving object described by ut is inside region (polygon) r within the time interval $[t_1, t_2]$. Variants according to

- **definitely** or **possibly** (uncertainty)
- **sometimes** or **always** (time)
- **uncertainty-time** or **time-uncertainty** (order)

8 combinations leading to operators

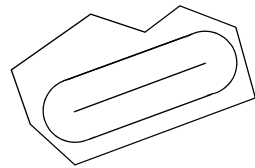
- **PossiblySometimeInside**
- **SometimePossiblyInside**
- **PossiblyAlwaysInside**
- **AlwaysPossiblyInside**
- **DefinitelySometimeInside**
- **SometimeDefinitelyInside**
- **DefinitelyAlwaysInside**
- **AlwaysDefinitelyInside**



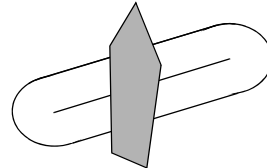
(a)

(b)

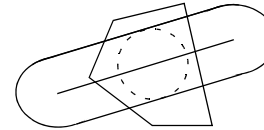
(c)



(d)



(e)

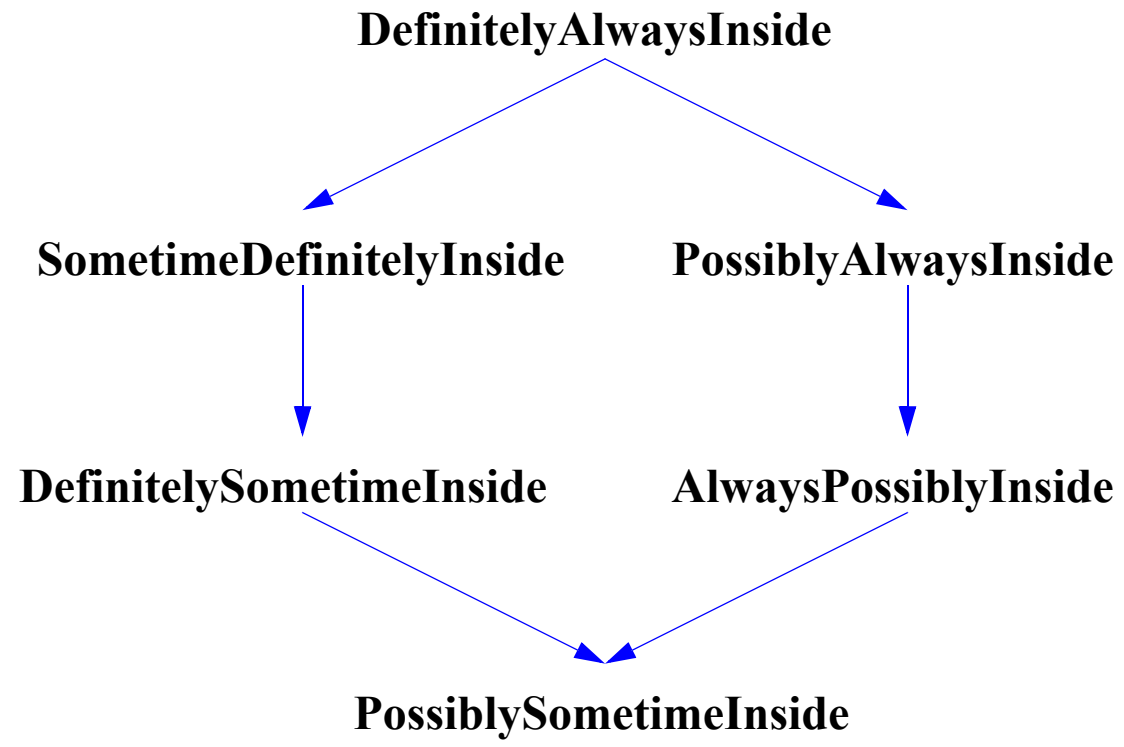


(f)

PossiblySometimeInside (a), PossiblyAlwaysInside (b), AlwaysPossiblyInside (c),
 DefinitelyAlwaysInside (d), DefinitelySometimeInside (e), SometimeDefinitelyInside (f)

Relationships Between Predicates

- **PossiblySometimeInside**(ut, r, t_1, t_2) \Leftrightarrow **SometimePossiblyInside**(ut, r, t_1, t_2)
 $\exists x \exists y P(x, y) \Leftrightarrow \exists y \exists x P(x, y)$
- **DefinitelyAlwaysInside**(ut, r, t_1, t_2) \Leftrightarrow **AlwaysDefinitelyInside**(ut, r, t_1, t_2)
 $\forall x \forall y P(x, y) \Leftrightarrow \forall y \forall x P(x, y)$
- **PossiblyAlwaysInside**(ut, r, t_1, t_2) \Rightarrow **AlwaysPossiblyInside**(ut, r, t_1, t_2)
 $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$
- **PossiblyAlwaysInside**(ut, r, t_1, t_2) $\not\Leftarrow$ **AlwaysPossiblyInside**(ut, r, t_1, t_2)
 \rightarrow cases (b), (c)
- **SometimeDefinitelyInside**(ut, r, t_1, t_2) \Rightarrow **DefinitelySometimeInside**(ut, r, t_1, t_2)
 $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$
- **SometimeDefinitelyInside**(ut, r, t_1, t_2) $\not\Leftarrow$ **DefinitelySometimeInside**(ut, r, t_1, t_2)
 \rightarrow cases (e), (f)



Algorithms for Spatio-Temporal Operations and Predicates

Point Query Operators

whereAt

$\underline{trajectory} \times \underline{instant} \rightarrow \underline{point}$ **whereAt**

Binary search + interpolation, $O(\log n)$ for a trajectory with n line segments.

whenAt

$\underline{trajectory} \times \underline{point} \rightarrow \underline{set(instant)}$ **whenAt**

Consider each segment of the trajectory, $O(n)$.

Predicates

Restriction to convex polygonal regions

Given polygon r and time interval $[t_1, t_2]$. Consider **query prism**

$$p(r) = \{(x, y, t) \mid (x, y) \in r \wedge t_1 \leq t \leq t_2\}$$

Filter + refinement technique

Filter Step

Approximate each segment trajectory volume between times t_i and t_{i+1} by a **minimum bounding volume** (MBV, box in 3D with axis-parallel faces).

Retrieve in the filter step the line segments of trajectories whose MBVs intersect the query prism $p(r)$. **Candidates** for the refinement step.

For faster retrieval, store MBVs in an index.

Refinement Step

Ideas:

- Reduce 3D case to 2D case
- Reduce region-region interactions to line-region interactions
(route segment-transformed query region instead of route segment uncertainty zone-query region)

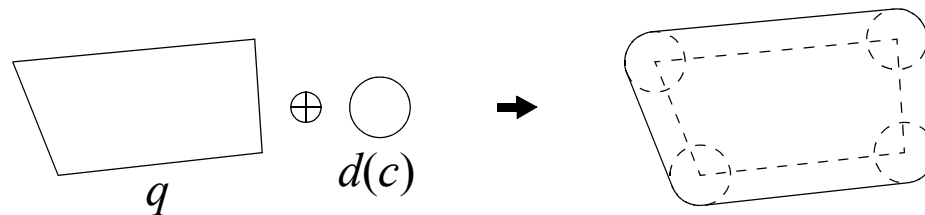
Refinement Step (cont.)

Some notations:

- $v(ut)$ the trajectory volume of a given uncertain trajectory $ut = (tr, th)$ between t_1 and t_2
- $v(ut, r) = v(ut) \cap p(r)$, intersection with query prism
- $\pi(tr)$ – projection of tr on the xy -plane (the *route* of tr)

Concept of **Minkowski sum**

- polygon q , disk $d(c)$ with radius c
- Minkowski sum $q \oplus d(c)$: the area swept by the disk is added to the polygon



- Can be computed in $O(n)$ time for a polygon with n edges.

Concept of **Minkowski difference**

- “Complementary” to Minkowski sum:
the area swept by the disk is removed from the polygon. Construction time: $O(n)$

PossiblySometimeInside

True iff $v(ut, r) \neq \emptyset$.

```
algorithm PossiblySometimeInside ( $ut, r, t_1, t_2$ )  
  construct  $r \oplus d(th)$ ;  
  if  $\pi(tr) \cap (r \oplus d(th)) = \emptyset$  then return false else return true fi  
end.
```

Complexity is $O(kn)$ where k is the number of line segments of tr between t_1 and t_2 , and n is the number of edges of r .

PossiblyAlwaysInside

True if $v(ut, r)$ contains a possible motion curve between t_1 and t_2 .

```
algorithm PossiblyAlwaysInside ( $ut, r, t_1, t_2$ )  
  construct  $r \oplus d(th)$ ;  
  if  $\pi(tr)$  lies completely inside  $r \oplus d(th)$  then return true else return false fi  
end.
```

Complexity is $O(kn)$.

DefinitelyAlwaysInside

True iff $v(ut, r) = v(ut)$ (i.e., if $v(ut, r)$ is inside $p(r)$)

```
algorithm DefinitelyAlwaysInside ( $ut, r, t_1, t_2$ )  
  for each straight line segment of  $tr$  do  
    if the uncertainty zone of the segment is not inside  $r$  then return false fi  
  od  
  return true  
end.
```

Check if the route segment has a distance from some edge of r less than th . This is done by checking whether $\pi(tr)$ is not inside the Minkowski difference of r and $d(th)$.

Complexity $O(kn)$.

SometimeDefinitelyInside

True iff $v(ut, r)$ contains an entire horizontal disk.

```
algorithm SometimeDefinitelyInside ( $ut, r, t_1, t_2$ )  
  for each straight line segment  $s$  of  $tr$  with  $\pi(s) \cap r \neq \emptyset$  do  
    if  $r$  contains a circle with radius  $th$  centered at some point on  $s$  then  
      return true  
    fi  
  od  
  return false  
end.
```

Done by checking whether $\pi(s)$ intersects the Minkowski difference of r and $d(th)$.

Complexity $O(kn)$.