# Progress in Kinetic Data Structures

Algorithms for Tracking Aggregate Properties
in a System of Moving Objects

**Leonidas J. Guibas**
Stanford University

# Algorithmics of Motion

The modeling of motion combines the continuous with the discrete.

- Motion is ubiquitous in the physical world.
- Typically motion exhibits continuity or coherence; instantaneously, object follow a continuous physical law.
- At irregular intervals, discrete events happen that alter the evolution laws of the moving system.

We are interested in tracking attributes of a system in motion.

# Knowledge of Motion

- The value of a continuously changing variable is hard to know exactly.
- Relationships between variables are more stable, and therefore easier to track.
- The value of the attribute of interest can be easy to compute, if we maintain a useful set of properties of the environment.
- We want to maintain a set of assertions that are relatively stable, yet useful for the computation at hand — an assertion cache.

# Kinetic Data Structures

A Kinetic Data Structure (KDS) dynamically maintains a set of assertions, called certificates, that altogether prove the correctness of an easy computation for the attribute of interest.

- Events of interest to the KDS are the failures of these certificates.
- At each certificate failure, the proof being maintained has to be repaired, and the attribute computation possibly updated.

Thus a KDS consists of an attribute certification, incrementally updated through time.

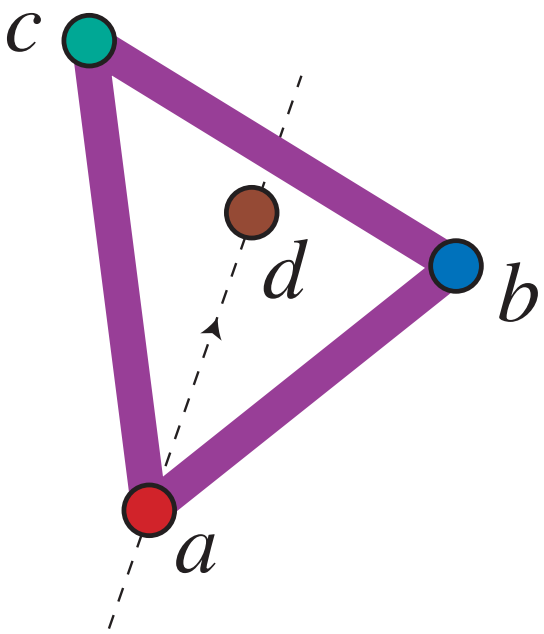
# Motion Models

The certificate failure times must be either detected or predicted. Prediction is possible if the objects follow known motion laws.

- At any moment, the motion law of an object may change. The failure times of all certificates involving that object must be then updated.
- More typically, certificate failures will trigger motion plan updates.

Thus a KDS performs an event-driven simulation and is a completely on-line structure.

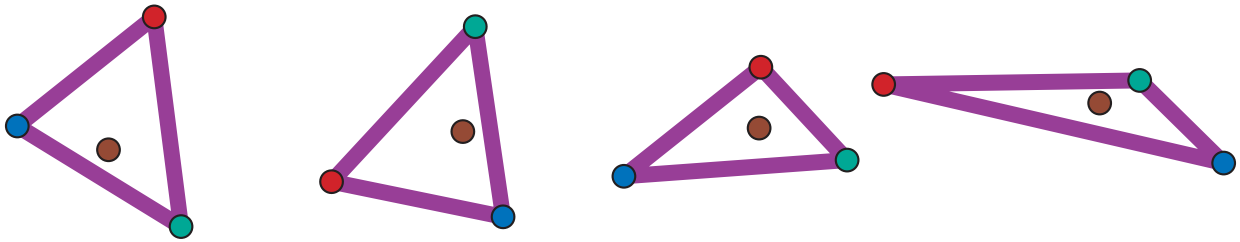
# Convex Hull of Four Points




Proof of correctness:

- $a$ is to the left of $bc$
- $d$ is to the left of $bc$
- $b$ is to the right of $ad$
- $c$ is to the left of $ad$

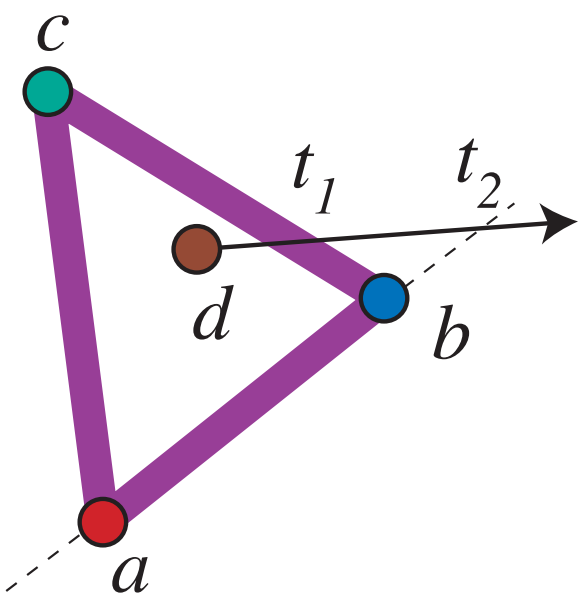
Four sidedness certificates.



(LJG)

# Failure Times and the Event Queue

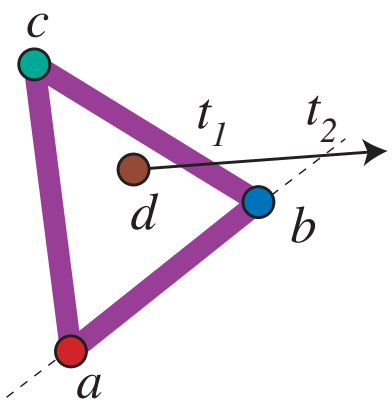
- Failure time of each certificate



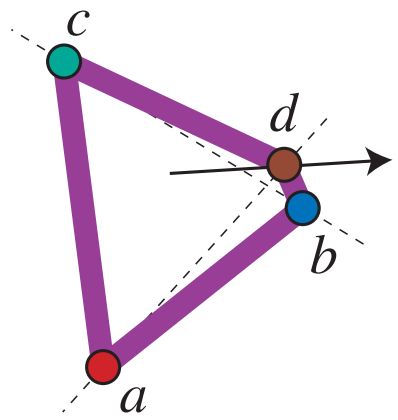

| Certificate | Failure time |
|---|---|
| $a$ left of $bc$ | never |
| $d$ left of $bc$ | $t_1$ |
| $b$ right of $ad$ | $t_2$ |
| $c$ left of $ad$ | never |

- Put the certificates in an event queue

# Processing an Event




| Old proof | New proof |
|---|---|
| $a$ left of $bc$ | $a$ left of $bc$ |
| $d$ left of $bc$ | $d$ right of $bc$ |
| $b$ right of $ad$ | $b$ right of $ad$ |
| $c$ left of $ad$ | $c$ left of $ad$ |



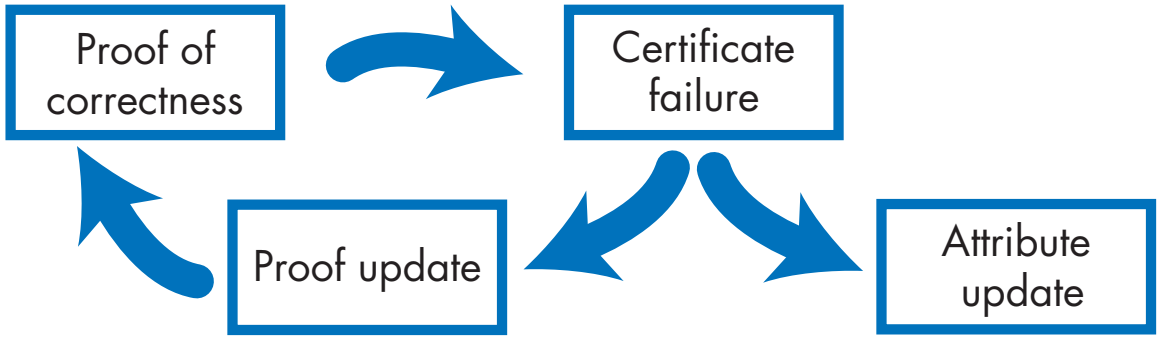

*Proof update*

# The Eternal KDS Loop

Two structures:

- Proof of correctness (based on certificates),
- Priority queue (sorted by failure time).

Event loop



# How to Obtain a KDS Proof

Here is a blind recipe. It rarely works:

- Stop the motion.
- Run a static algorithm to compute the attribute of interest.
- Collect all the tests the algorithm performed and certify their outcomes.
- Continue the motion.

But what to do when a certificate fails? How is the proof to be repaired and the attribute updated?

## How Much to Know? The Faustian Dilemma

The more we know about the world, the easier it will be to repair the proof. But the more assertions about the world we maintain, the more certificate failures we will have to process.

A good KDS is

- responsive $\Rightarrow$ allows quick proof repair
- efficient $\Rightarrow$ minimizes the number of events to be processed
- compact $\Rightarrow$ minimizes of certificates maintained
- local $\Rightarrow$ permits inexpensive motion plan updates

Designing a good KDS is still an art; trade-offs, just as for classical DSs.

# A Smarter Sampling of Time

A fixed sampling of time will either oversample or undersample a system whose evolution is governed by irregularly spaced discrete events.

The proof maintained by the KDS provides intelligent advice to the simulation on when to sample the system.

A good KDS only samples the system when the attribute of interest, or its computation process, needs to change.

# KDS Properties Summary

A KDS maintains through time a set of assertions about the world facilitating or trivializing the computation of the attribute of interest. KDSs gain efficiency by exploiting coherence in the motion. They

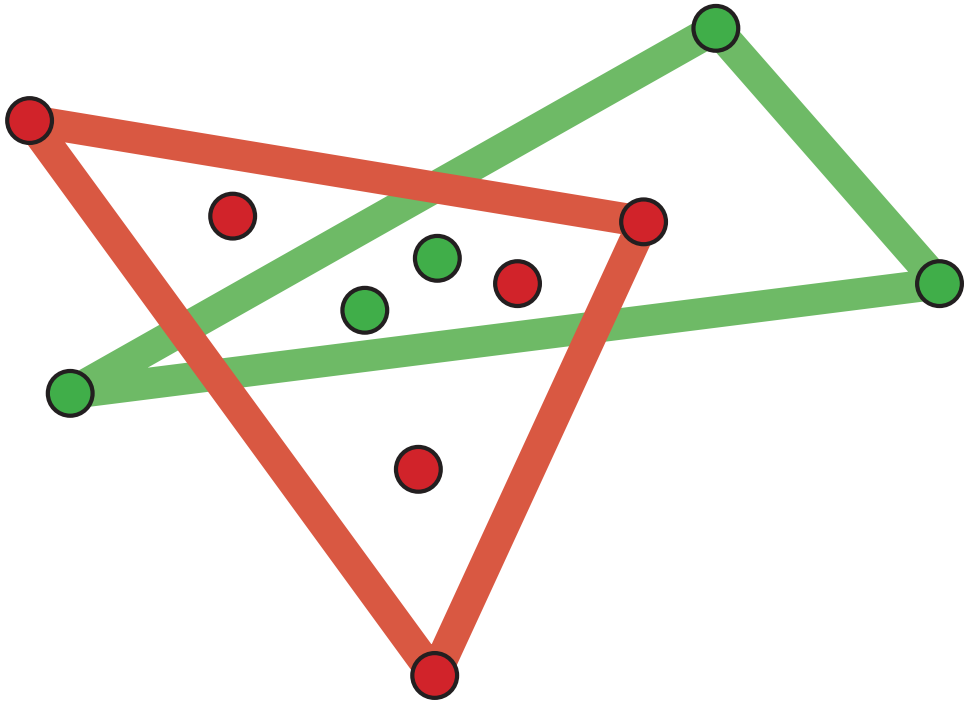
- sample the system only as needed to maintain the attribute of interest,
- allow motion plan updates at any time,
- provide the first formal framework for assessing the performance and complexity of on-line motion algorithms, and
- are well suited to continuous systems whose evolution is punctuated by discrete events.

# Early KDS Development

- **Extent Problems**: convex hull, diameter, width for moving points
- **Proximity Problems**: closest pair, Voronoi/Delaunay diagrams
- **Connectivity and Communication Problems**: MST for geometric and general graphs, kinetic clustering, kinetic routing graphs
- **Visibility**: BSPs and visibility orders
- **Collision Detection**: for rigid and flexible objects

# Kinetic Convex Hull Algorithm

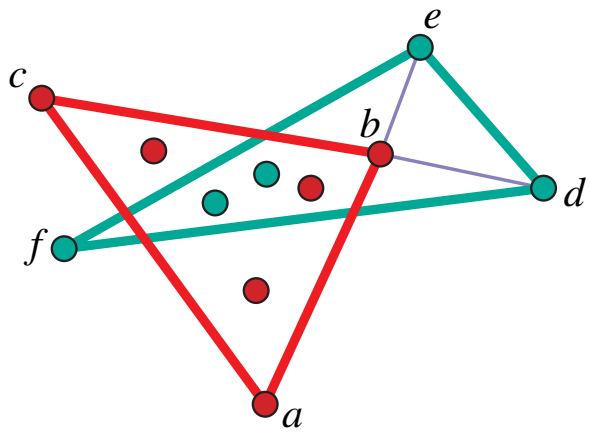
Which are the relevant CCW certificates?



# Certificates

Certificates for merging step:

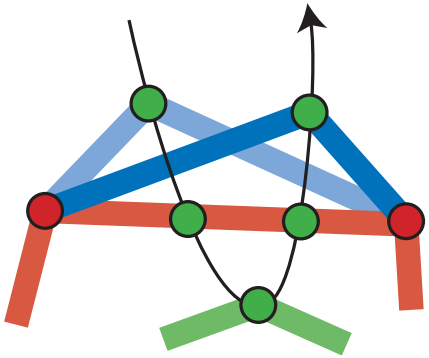
- Slope comparisons between edges:
  - $(ab, de)$ is counter-clockwise
  - $(de, bc)$ is counter-clockwise
  - ...
- Sideness tests:
  - $b$ left of $de$,
  - ...



# Events

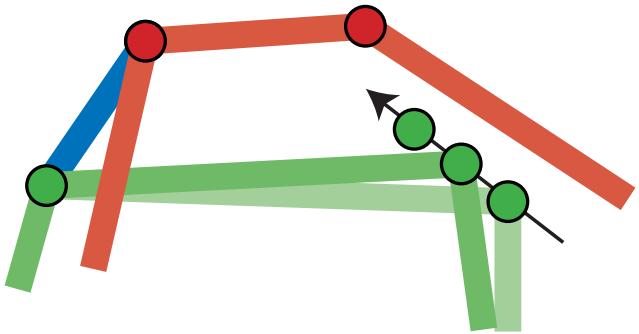
- Events happen when certificates fail:

Sideness test



Triangle event(s)

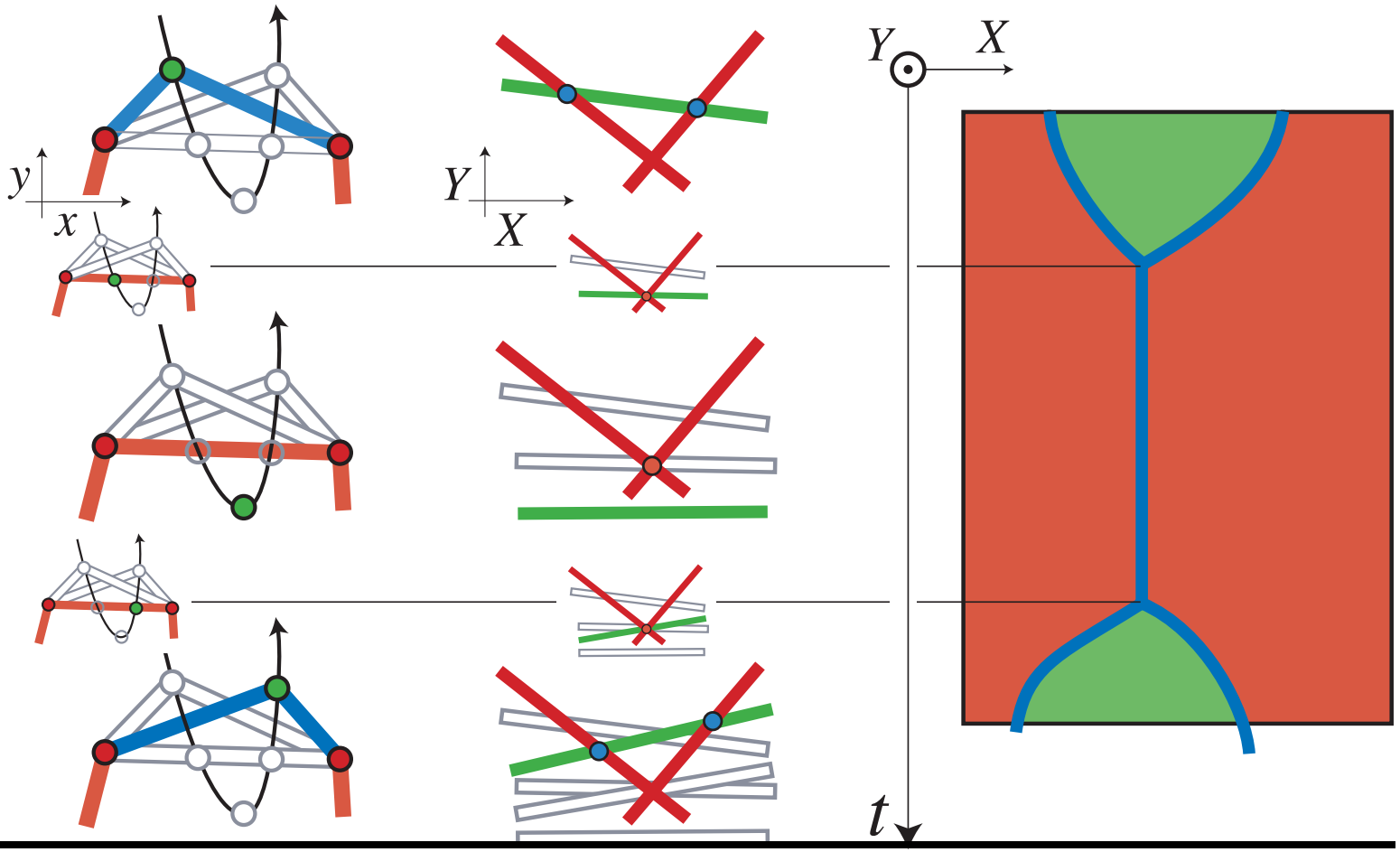
Counter-clockwise test



Slope event

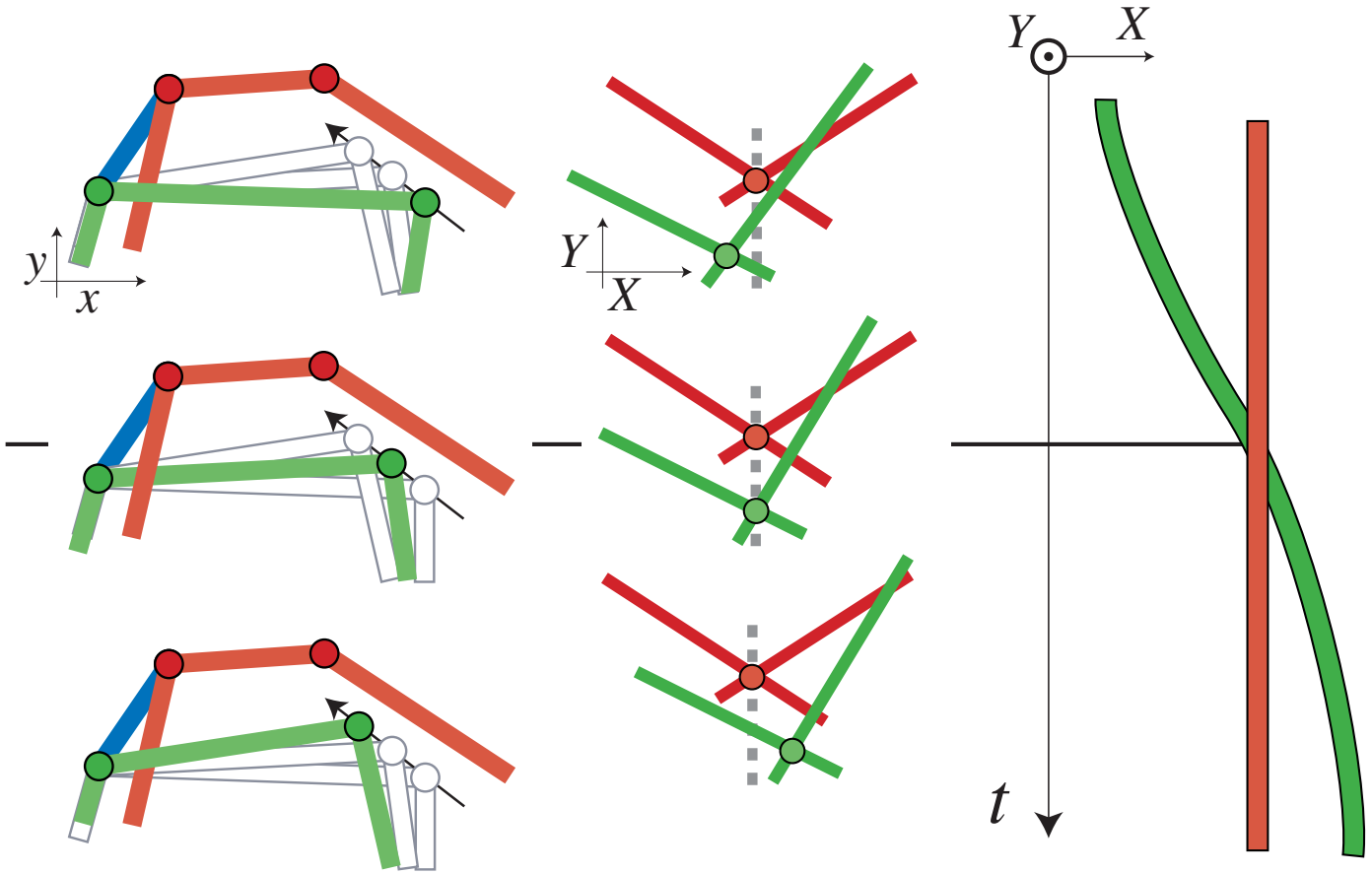
- It is easy to update the proof (responsive)
- Events can happen at all levels of the recursion tree

# Triangle Events: Duality and Space/Time View




(LJG)

# Slope Events




(LJG)

# Proof of Efficiency

Number of events $\approx$ Upper Envelope/Overlay Complexity

- **Theorem.** [Halperin, Sharir '94] The upper envelope of $n$ algebraic surfaces of fixed degree has complexity $O(n^{2+\epsilon})$.
- **Theorem.** [Agarwal, Schwarzkopf, Sharir '94] The overlay of two upper envelopes of $n$ algebraic surfaces of fixed degree has at most $O(n^{2+\epsilon})$ bichromatic intersections.

# Convex Hull Summary

- D&C Kinetic Data Structure:

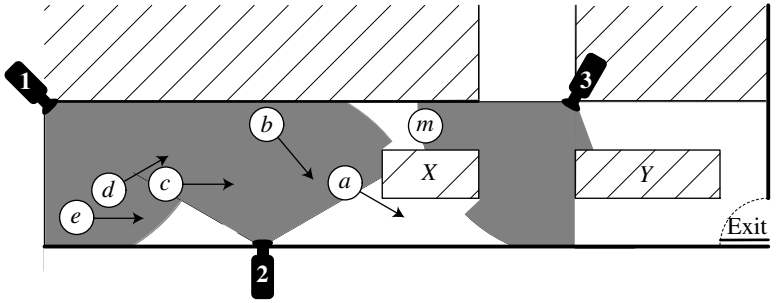
| | |
|---|---|
| Compact | $n \log n$ |
| Responsive | $\log n$ |
| Local | $\log n$ |
| Efficient | $n^{2+\epsilon}$ |

The KDS maintains the convex hull in a completely on-line fashion.

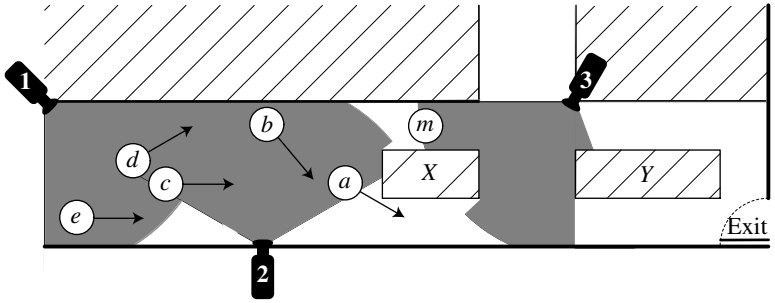
# KDS Videos

# KDSs for Real, not Virtual Settings

Certificate failures must be detected using sensors. KDS now provides a spatial reasoning mechanism.



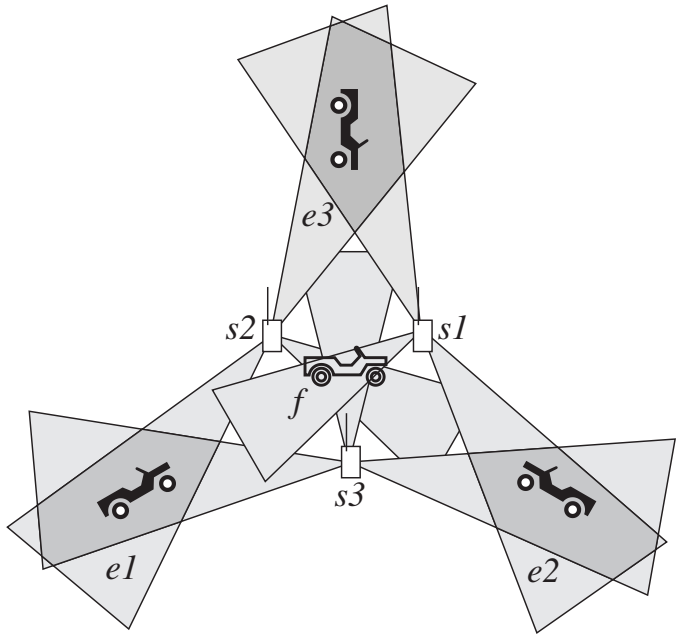

| |
|---|
| $a \succ b$ (camera 2) |
| $b \succ c$ (camera 2) |
| $c \succ d$ (camera 1) |
| $d \succ e$ (camera 1) |




| |
|---|
| $a \succ b$ (camera 2) |
| $b \succ c$ (camera 2) |
| $b \succ d$ (camera 1) |
| $d \succ e$ (camera 1) |

# Direct Sensing of Relations

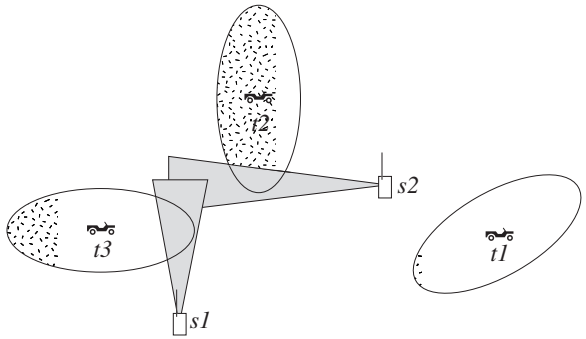
Sensors may be able to ascertain spatial realtions without object localization.

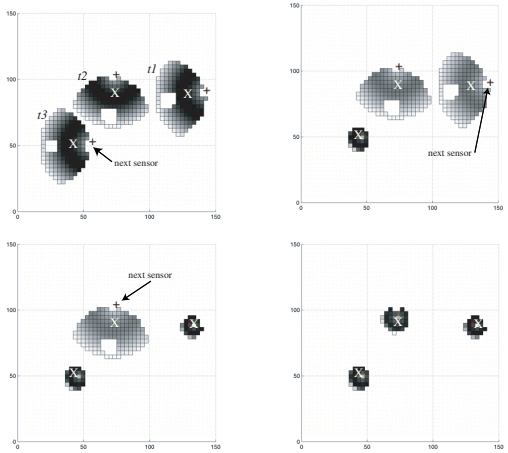



| $\mathrm{CCW}(e_1, e_2, f)$ |
|---|
| $\mathrm{CCW}(e_2, e_3, f)$ |
| $\mathrm{CCW}(e_3, e_1, f)$ |
| $f$ surrounded by $e_1, e_2, e_3$ |

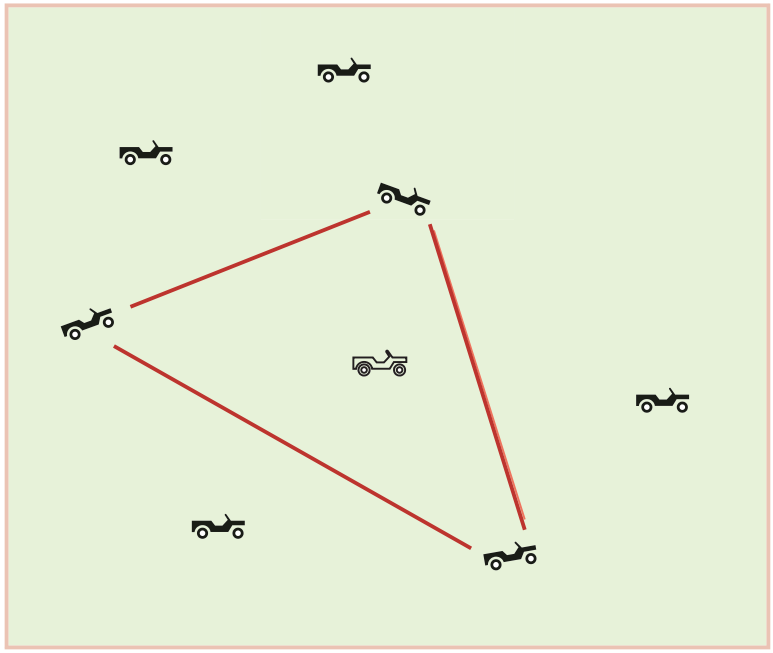
# Relation Uncertainty and Sensor Tasking Strategies

Certainty in a relation is coupled to target localization.



# Global Reasoning Without Full World Models



# Plans for Future Work

- Investigate new KDSs for specific geometric/combinatorial problems
- Deal with multiple certificate failures at once
- Parallel implemenations for distribured simulations
- Incorporate the cost of sensing in the KDS model
- Develop KDSs with probabilistic motion knowledge and probabilistic reasoning about attribute values

# Acknowledgments

Personnel: J. Basch, J. Comba, J. Gao, O. Hall-Holt, M. Karavelas, C. Silverstein, E. Veach, L. Zhang, A. Zhu

Senior Collaborators: P. Agarwal, D. Eppstein, J. Erickson, J. Hershberger, M. Herzinger, J. Stolfi

# Kinetic Data Structures are Fun