Interactive Collision Detection

Author: Philip M. Hubbard
Speaker: Martin Duncko

Introduction

- Simulation of physical world
- Collision-handling algorithm
  - Detection algorithm
  - Response algorithm
- Approximation
  - Space-time bound
  - Sphere-tree

Naive algorithm

for $t \leftarrow 0$ to $\hat{t}$ in steps of Δt
    for each agent $A_i \in \{A_1, \ldots, A_N\}$
        move A_i to its position at time t
        for each agent $A_j \in \{A_{i+1}, \ldots, A_N\}$
            move A_j to its position at time t
            if (surfaces of A_i, A_j penetrate)
                then a collision occurs at time t

- Problems:
    - Fixed-timestep weakness
    - All-pairs weakness
    - Pair-processing weakness

Space-time bounds

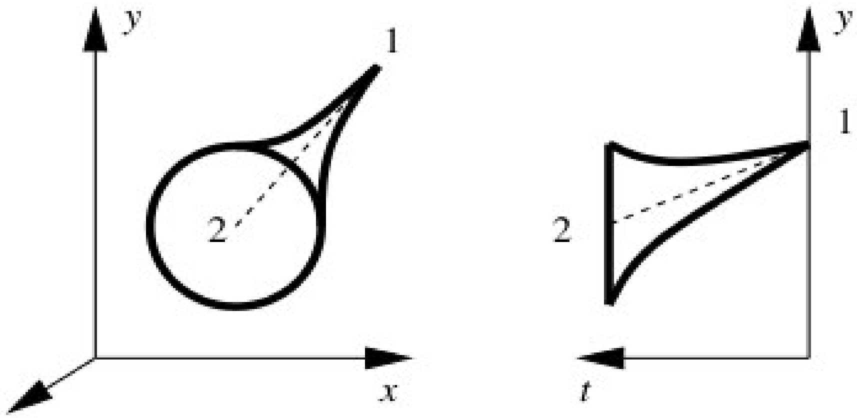
- Point A:
  - Position in space in time : x(t)
  - Velocity x'(t)
  - Acceleration x''(t)
- We know:
  - x(0), x'(0),
  - scalar M:

$$|\mathbf{x}(t) - [\mathbf{x}(0) + \dot{\mathbf{x}}(0)t]| \le \frac{M}{2}t^2, \quad 0 \le t \le \hat{t}.$$

Space-time bounds 2

- 3D bound (sphere) at time *t*
- 4D bounds through time *t*

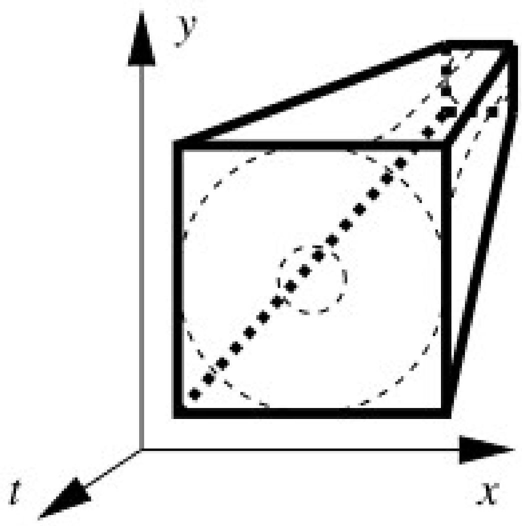
- Parabolic horn – bounding structure



Space-time bounds 3

- Parabolic factor – costly intersections
- Hyper-trapezoid – simple 4D polyhedron
  - Encloses parabolic horn
- Cross section :
  - at $t = 0$ and $t = \hat{t}$ - isothetic cubes
  - linear interpolation between endpoint cubes

Space-time bounds 4



- Hyper-trapezoid for motion in 2D

Space-time bounds 5

- Hyper-trapezoid has 6 4D faces
- One for each 3D face of cross-sectional cubes
- Each cross-section of 4D face is isothetic 3D square

Space-time bound intersections

- 2 agents collide at time t
- 2 space-time bounds collide at time t'
  - $t' < t$

- Detection alg. compute t' as smallest t, where collision can appear

Intersection

- Intersection between two hyper-trapezoid faces is condition for intersection of two space-time bounds

- So search for intersection of face intersections

Intersection 2

- Each 3D cross section of a 4D hyper-trapezoid face is normal to one of standard axis
- Set of all 4D faces partition to:

$$F_\alpha = \{f \,|\, \text{face } f \text{ is normal to axis } \alpha\},\ \alpha \in \{x, y, z\}.$$

- If hyper-trapezoids intersect for first time, there must be an intersection in the same ax

Intersection 3

- Itersection in one set:
  - Project each face $f \in F_\alpha$ to α-t plane
  - 2D line segment
  - Faces intersect if $t = t'$ cross section intersect
- Two cross sections are isothetic squares with the same ax - two dimensional problem
- Bentley-Ottman algorithm

Detecting collisions

$t_{\text{build}} \leftarrow t_{\text{end}}$ /* $t_{\text{end}} = 0$ before first call */
while ($t \geq t_{\text{end}}$)
    rebuild space-time bounds as of t_{build}
    $\check{t}_i \leftarrow$ earliest inter. between bounds, B_1, B_2
    $t_{\text{end}} \leftarrow \check{t}_i$
    if ($t_{\text{end}} - t_{\text{build}} < \Delta t$)
        $could\_overlap \leftarrow$ TRUE
        if (B_1 and B_2 really inter. before expiring)
            if (pair-processing algorithm finds that B_1's
                        and B_2's agents penetrate at t_{build})
                return collision at t_{build}
        $t_{\text{end}} \leftarrow t_{\text{build}} + \Delta t$
    else
        $could\_overlap \leftarrow$ FALSE
    $t_{\text{build}} \leftarrow t_{\text{end}}$
return no collision as of t

Detecting collisions 2

- $t = 0$
  - algorithm builts space-time bounds for all agents
- Bounds expire, when M are unknown
- Compute t'
- No working when $t < \hat{t}$
- Δt – minimum temporal resolution

Sphere trees

- Easy check for penetration
- Agents are approximates as spheres
- Sphere tree
  - Deeper level – more spheres
  - Children of one sphere at level *i* are al spheres at level *i+1* that it bounds
- Agents sphere-tree is built just once
- Same rigid body transformation as agents

Building sphere trees

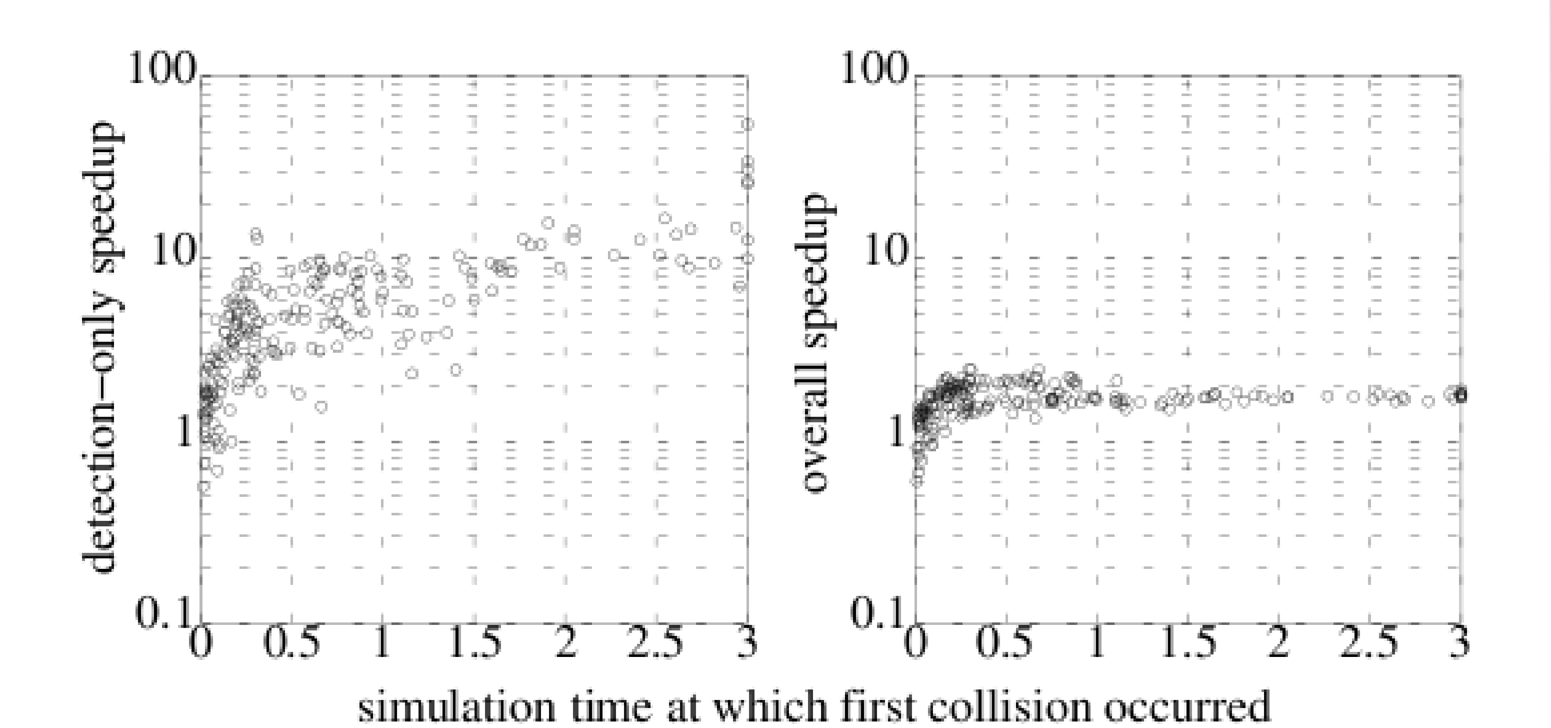
- Built octree for agent
- Circumscribing spheres of octants at level j are spheres set at level j
- Resolution doubles with each level

- Only polyhedral agents

Performance – broad phase

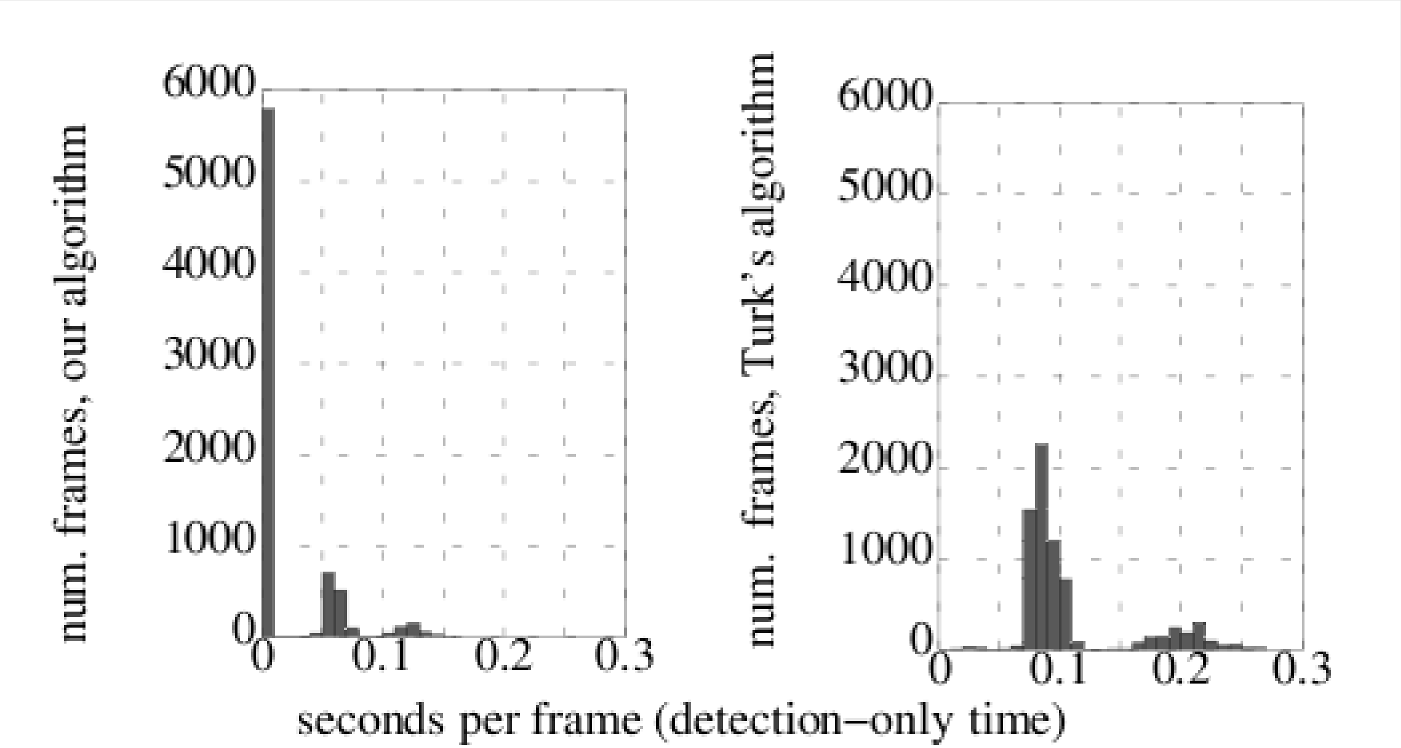
- Compared to Turk's algorithm
- Test program generates random configurations of isothetic cubes and forces
- Unix clock routine measures time to find first collision

Performance-broad phase2



- Turk's algorithm divide this algorithm
  - Only 1+

Per frame performance



- Slowest time was slower than Turk's
- But not often

Performance - narrow phase

- Spaceship simulator
  - User control – forward and rotate
- Dron ships
  - Random moves
- Sphere tree vs BSP trees
  - BSP – exact results

Performance

- Broad + narrow phase 5 to 7 times faster than Turk's algorithm with BSP

Thank you for your attention