Collision and proximity avoidance for robust behaviour of real-time robot applications

Alexandru Dumitrache Theodor Borangiu Anamaria Dogar

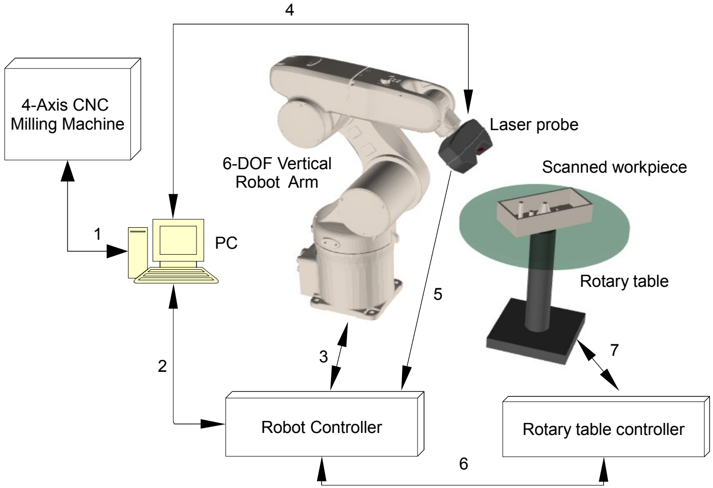
Centre for Research & Training in Industrial Control
Robotics and Materials Engineering
University Politehnica of Bucharest

RAAD 2010

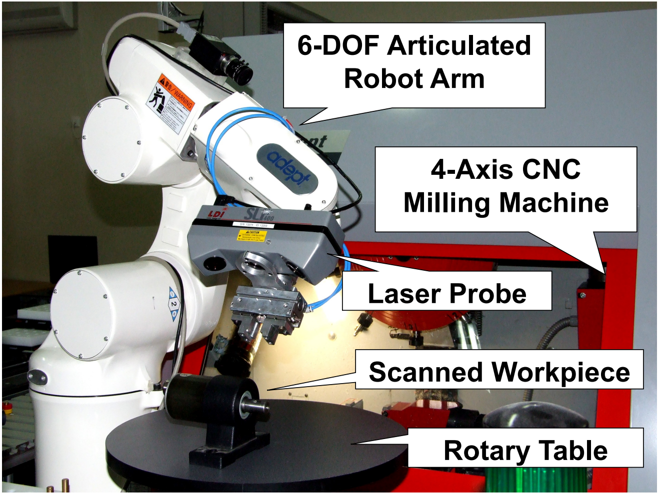
Outline

1. Overview
   - Application – 3D Laser Scanning System
   - Collision detection support in robot controllers
2. Related Work
   - Surveys
   - Collision detection libraries
3. Collision detection and avoidance
   - Implementation with Open Dynamics Engine
   - Integration with motion planning algorithms
4. Experiments
   - Collision query benchmark
   - Predictive collision detection for manual operation
   - Collision detection for existing robot applications
   - Educational robot simulator

3D Laser Scanning System Overview



3D Laser Scanning System Overview



Collision detection support in robot controllers

Adept Technology - SmartController

- 4 user-defined static objects
- Only the tool center point is tested against the obstacles
- Robot program cannot alter the obstacles

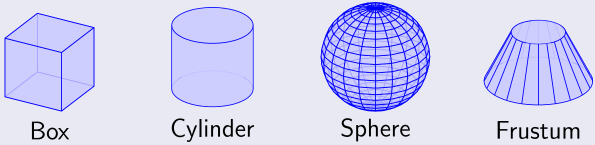


ABB Robotics - IRC5

- RobotWare option for Collision Detection
- Stops the robot if the torques exceed allowed values

Surveys

Ming C. Lin and Stefan Gottschalk
Collision Detection Between Geometric Models: A Survey
In Proc. of IMA Conference on Mathematics of Surfaces, pp. 37–56, 1998.

P. Jiménez and F. Thomas and C. Torras
3D Collision Detection: A Survey
Computers and Graphics, vol. 25, pp. 269–285, 2000.

Collision detection libraries

Engines for rigid body dynamics

- Proprietary licenses: NVidia PhysX, Intel HAVOK, Newton Game Dynamics, True Axis
- Public licenses: Open Dynamics Engine (LGPL/BSD), Bullet Physics (zlib), JigLib (zlib), Tokamak (BSD)

Standalone libraries for collision and proximity queries

- Traditional (discrete) collision detection
  - Convex polyhedra: GJK, I-COLLIDE, SWIFT
  - Polygon soups: RAPID, PQP, V-COLLIDE, SWIFT++, V-CLIP, OPCODE, GIMPACT
- Continuous collision detection (CCD)
  - FAST: for rigid polyhedra
  - CATCH: for articulated models

Implementation with Open Dynamics Engine

Open Dynamics Engine

- Open source physics engine (LGPL / BSD)
- Python wrappers: PyODE (low level), cgkit (3D rendering)
- It is possible to use only the collision tests (OPCODE/GIMPACT)

Primitive function: `collisionQuery`

Verifies a given robot configuration against collisions

```
def collisionQuery(Robot, Joints)
    ...
    collisions = []
    space.collide(collisions, nearCallback)
    ...
def nearCallback(collisions, geom1, geom2):
    if collisions: return
    if ode.collide(geom1, geom2):
        collisions.append((geom1,geom2))
```

Integration with motion planning algorithms

Heuristic motion planner: *Ray Shooting*

- Exploits redundancy of the 7-DOF mechanism
- Generates smooth motions for rotary table
- Avoids singularities, robot configurations close to joint limits, and also collisions

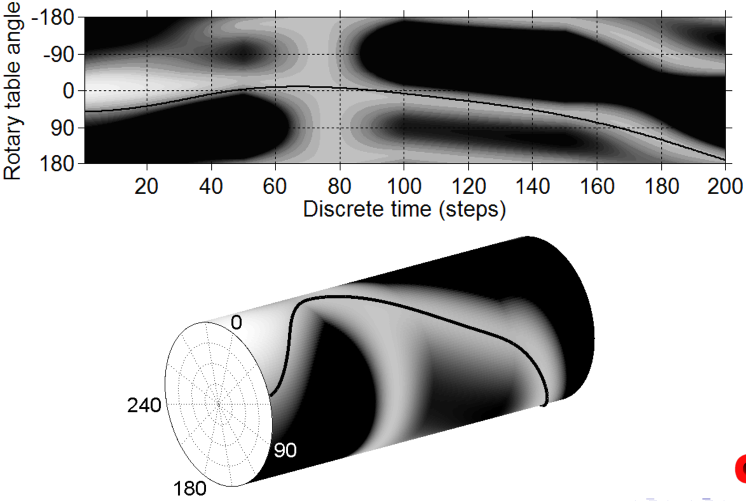
Constraints

- Real-valued functions in $[0 \ldots 1]$
  - $f_i = 1$: fully satisfied
  - $f_i = 0$: not satisfied
  - $0 < f_i < 1$: satisfied only partly
- Multiply all constraints to get the global value:

$$f = \prod_{i=1}^{m} f_i$$

Integration with motion planning algorithms

The constraints can be visualized as grayscale configuration maps

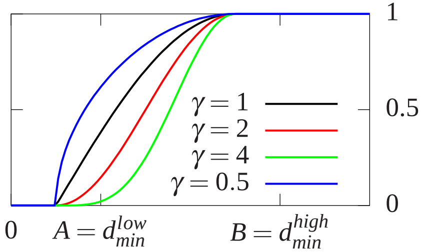


Integration with motion planning algorithms

Constraint function for collision detection

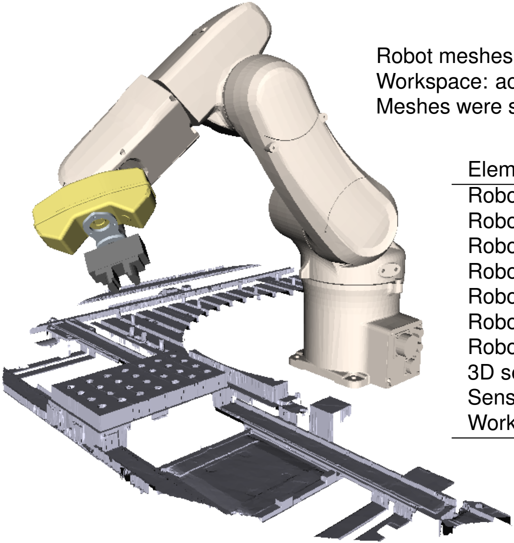
- Input: minimum distance between two bodies
- Tuning parameters:
  - $A = d_{min}^{low}$
  - $B = d_{min}^{high}$
  - γ_C – shape factor

$$f_C(d_{min}) = \begin{cases} 0, d_{min} < A \\ sin\left(\frac{d_{min}-A}{B-A} \cdot \frac{\pi}{2}\right)^{\gamma_C} \\ 1, d_{min} > B \end{cases}$$



- Minimum distance has to be higher than A
- If it is B or higher, that's perfect

Collision query benchmark



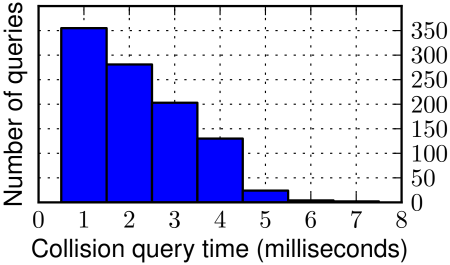
Robot meshes: from manufacturer's CAD files
Workspace: acquired by 3D scanning
Meshes were simplified in MeshLab

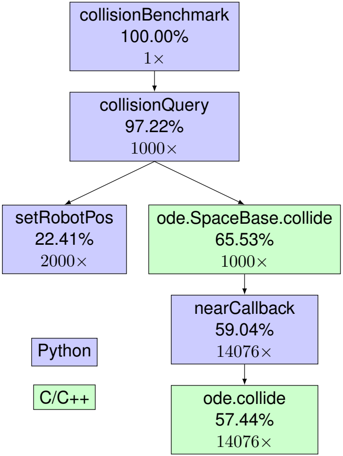
Element	Triangles	Vertices
Robot base	5000	2490
Robot link 1	5000	2538
Robot link 2	2500	1257
Robot link 3	2500	1262
Robot link 4	2500	1252
Robot link 5-6	5000	1275
Robot gripper	388	196
3D sensor	10000	4489
Sensor fixture	5000	2494
Workspace	50000	26344

Collision query benchmark

Monte Carlo simulation

- 1000 random configurations for the robot arm
- Each joint is uniformly distributed between 0 and $360°$
- Average query time: 2.2 ms
- Worst case: under 10 ms.





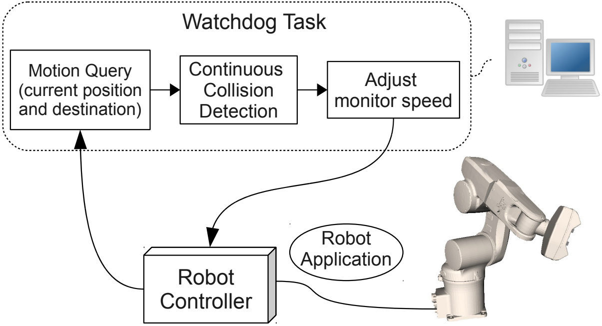
An implementation in C/C++ will be max. 1.5 times faster

Predictive collision detection for manual operation

Collision detection for existing robot applications

Motion supervisor task

- Robot motion capture via Ethernet
- Trajectory is known in advance $\Rightarrow$ no prediction necessary

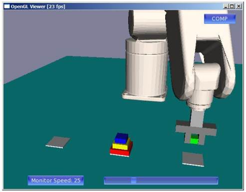


Network delay may slow the robot down, but it should not affect reliability

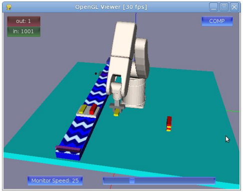
Cross-platform robot arm simulator

robot-sandbox *http://alexdu.github.com/robot-sandbox/*

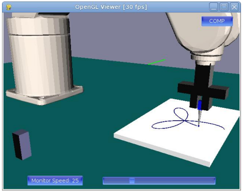
- Open source implementation using Python, cgkit and ODE
- Uses rigid body dynamics, including collision detection
- Suitable for classroom usage
- Robot language: a subset of Adept V^+



(a) Hanoi towers



(b) Conveyor belt



(c) Robot drawing

Conclusions

- Collision avoidance techniques for robot applications
- Open source implementation: Python, Open Dynamics Engine
- Applicable in physical and simulated environments
- Suitable for real-time applications
- Can be implemented on existing robot controller
- Requires 3D meshes of the robot and environment