Multi-agent snake-like motion with reactive obstacle avoidance

George Birbilis
Prof. Nikos Aspragathos

Mechanical Engineering & Aeronautics Department, University of Patras

Introduction – the problem

- calculate motion of a serial linkage (chain)
  - physical chains: serial manipulators
  - virtual chains: mobile robot snake-like swarm formations
- time varying environment, possibly containing unknown obstacles
  - stationary obstacles
  - moving obstacles (at speeds relative to joint motors’ speed)
- each chain part receives sensor feedback on proximity to obstacles
- using a *geometric constraint satisfaction* approach

Introduction – approaches

- Motion planning in a time varying environment: Many approaches (see Hwang and Ahuja review)
- Unknown moving obstacles favours local planning, global re-planning too expensive:
  - with many obstacles, or
  - when having to plan for many DOF - Degrees Of Freedom, highly redundant systems (see Chen and Hwang, Challou et al.)
- more degrees of freedom (DOFs) = higher system flexibility
- value an approach that:
  - easily scales up to highly redundant systems
  - supporting systems with less DOFs
- Top-down centralized approach: increasing complexity and cost as the number of DOFs rises
- Bottom-up, modular approach suggested instead, modelled as *multi-agent system*

Introduction – swarm approaches

Mobile robot swarms

Dorigo et al.:

- evolving self-organizing behaviours for “swarm-bot”
- eight robots connected by flexible links in snake formation
- able to:
  - negotiate a unique direction
  - produce coordinated movement along negotiated direction
  - collectively avoid walls
- Flexible links:
  - swarm tends to change shape during coordination phases and during collision with obstacles
- Members tend to maintain own direction of movement:
  - swarm able to go through narrow passages, deforming shape according to obstacles’ configuration

Introduction – manipulator approaches

<u>Robotic manipulators</u>

based on multi-agent platform for motion planning problem:

- motion planning on each subpart of manipulator structure (almost all)
- combine solutions of sub-problems into global problem solution (almost all)
- composition implemented in real-time, via interaction (cooperation or contention) of agents that control various manipulator parts (most approaches)
- Overgaard et al.:
  - multi-agent system with joint and link agents
  - control 25-DOF snakelike robot
  - environment with obstacles modelled using an *artificial potential field (Khatib)*
- Bohner and Lüppen:
  - 7-DOF robot
  - only robot joints as agents
  - *sensor-data integration* and *motion planning* per-agent
  - decomposing problem and reducing complexity to sum of subproblems

Introduction – our approach

<u>Our previous work</u>

- Multi-agent system
- Each software agent controls specific part of planar manipulator joint-link chain
- Agents interact with each other to adapt (in real-time) manipulator configuration to
  - external events
  - changing situations
- Geometric constraint satisfaction approach
- Reduces motion planning of manipulator to motion planning of single part of it (e.g. tooltip)
- rest of manipulator chain parts **react** and *adapt* or *object (veto)* to moving part's motion.

<u>In this paper</u>

- slightly revise agent interactions
- implement the proposed architecture:
  - 2D plane
  - 3D space
  - physical chain linkages (serial manipulators)
  - virtual chains (swarms of mobile robots)
- new findings on potential applications.

Multi-Agent systems

Liu et al. define:

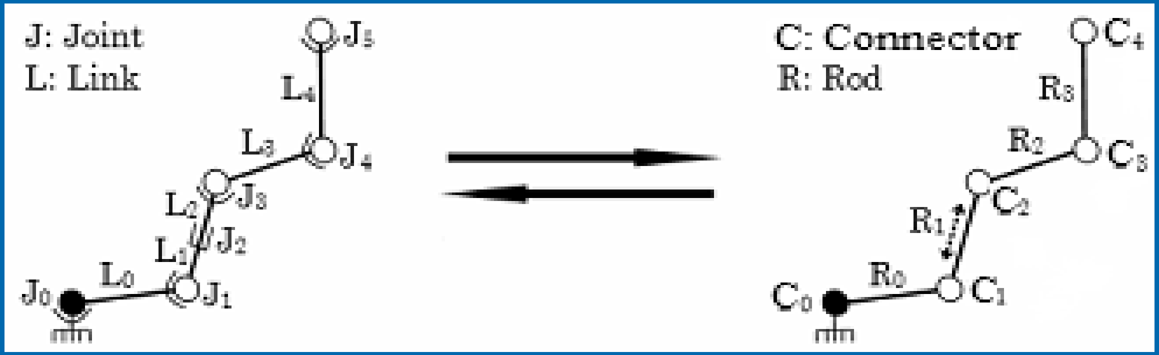
- agent = an entity
  - able to live and act in an environment,
  - able to sense its local environment,
  - driven by certain objectives,
  - has some reactive behaviour
- multi-agent system = a system
  - has an environment (space where agents live),
  - has a set of reactive rules (governing interaction between agents and environment - laws of agent universe)
  - has a set of agents

Proposed conceptual system model

- control chain's motion mimicking motion of chain of rods, interconnected at their endpoints with connectors (potentially constrained universal joints)
- chain's two endpoints are connectors
- rods can be resizable ([min, max] length constraint)
- any part of chain initiating motion "pushes" or "pulls" other parts of chain
- constraints defining chain structure kept

- high number of rods and connectors: behaves like snake crawling amidst obstacles.

Mapping kinematic to conceptual constraints (for manipulators)



Manipulator chain:

- base connector (links to environment): cater for potentially mobile (free or partially/totally constrained) manipulator base
- rotational joint and its outgoing link: map directly to first connector of a fixed-length rod
- translational joint and both its incoming and outgoing links: map to connector + expandable rod

- Connector agent: position property
- Rod agent: length property

- Constraints: rod length bounds respected by positions of connector agents at rod's two endpoints

- Joint parameter values: calculated geometrically from connector positions

Mapping kinematic to conceptual constraints (for swarms)

Swarm chain formation:

<u>rods</u>: distance constraints between pairs of consecutive robots

<u>connectors</u>: angular constraints between triplets of consecutive robots

Mapping between conceptual and kinematic model done same as for manipulator chains

Inter-agent relationships

Master – Vetoable slave relationship

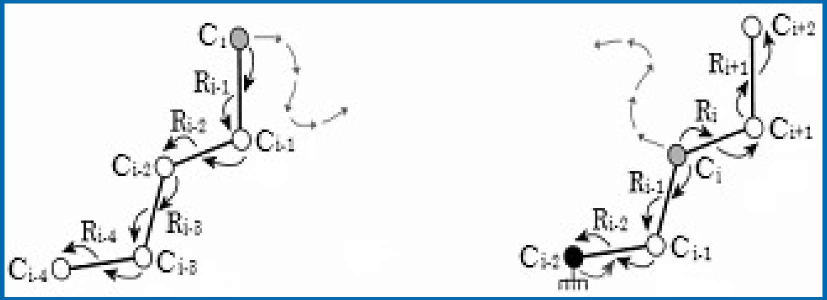
Slave entity:

- listens for changes to master entity's state
- reacts to those changes
  - changing own state or
  - objecting to master changing state (*veto*)
- can take part as master in other relationships (have its own slaves)

Chain of Master – Vetoable Slave relationships:

- finite number of entities
- first entity is master of second, second entity master of third etc.

Change event propagation



- *State change events* at head of master-slave relationship chain propagate towards tail
- Slave reacts and tries to adapt to its master's state change event (changing own state and causing own slave to react too etc.)

- Any part (not just endpoints) can initiate motion
  - acting autonomously to avoid obstacles
  - receiving motion commands from planner module or human operator
- Chain treated as two sub-chains, with their head the part that initiated motion, and tails the head and tail of original chain

Change event back-propagation (veto)

- Slave part can object (veto) to motion of its master part if it can't
  - move to adapt to master's motion and still
  - preserve the chain model and environmental constraints

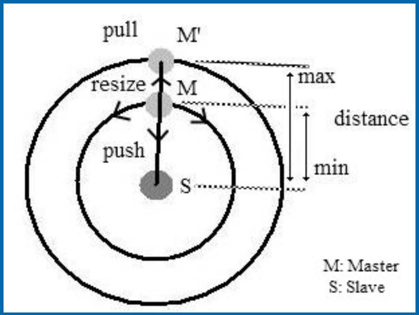
  Vetoing allows for:
  - fixed base manipulators (Connector agent at fixed base always objects to its relocation)
  - replaning in case some slave is trapped in obstacles or malfunctioning

Reactive agent ruleset

Slave entity *ruleset:*

- defines how it reacts to changes of its master entity's state.
- defines *reactions* to preserve *constraints* imposed on slave object and its relation to its master:
  - by design
    - internal, hardware constraints
    - chain structure
  - decided on the fly at runtime
    - environmental constraints
    - obstacles (stationary or moving ones)
    - failure of subparts

Reaction to master motion



Reaction rules exposing *chain structure constraint preservation behaviour: Push, Pull, Resize*

We define a guiding line (g), connecting master's new position and slave's old one:
- Slave connector may move on g, being pushed or pulled by its master
- Interconnecting rod may resize and is always aligned to g

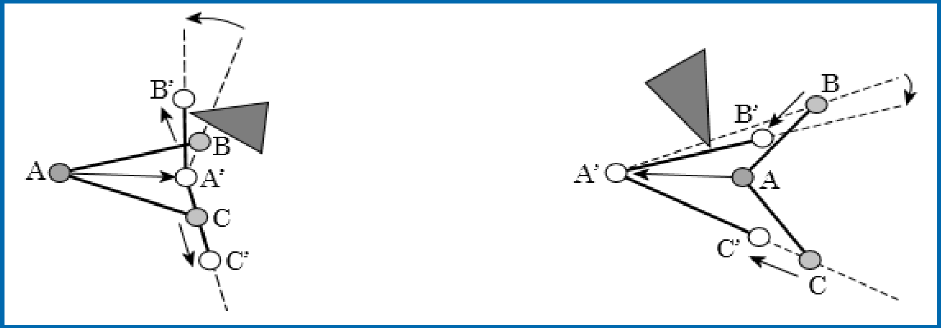
Push rule:
- Condition: master-slave connectors' distance (d) < minimum length (minRL) of interconnecting rod
- Reaction: slave moves on g, master-slave connectors' distance = minRL (rod shrunken)

Pull rule:
- Condition: d > maximum length (maxRL) of interconnecting rod
- Reaction: slave moves on g, master-slave connectors' distance = maxRL (rod stretched)

Resize rule:
- Condition: d in [minRL, maxRL]
- Reaction: slave doesn't move, rod resizes within [minRL, maxRL]

Together called a Push-Pull-Resize behaviour

Reaction to sensed obstacles



Rotate rule:
- Condition: interconnecting rod collision with obstacle
- Reaction: rod rotates around master connector

Motion of master connector resulting in rotation of guiding line at rod-obstacle collision point that is closest to master

This *environmental constraint* preservation behaviour is called *Push-Pull-Rotate*

Autonomous navigation

Navigation of motion initiating agent (controller):

Extended a wall following algorithm with:

- target seeking behaviour
- corridor passage behaviour

Steps:

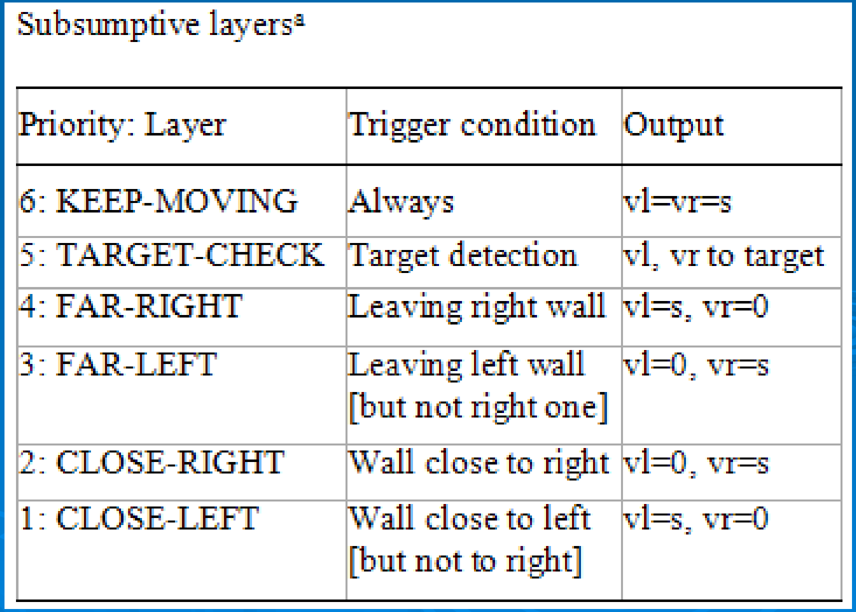
8. reading left-front and right-front side (laser scanner) sensors to get closest distance to visible obstacles at left and right side of controlling agent
9. Based on sensor readings, mover's position adjustment is calculated

Achieve minimum and maximum distance from obstacles (simulated at 0.7 and 0.9 meters respectively): defining mover's reactive behaviour using stack of subsumption layers

- Lower subsumptive layers: potentially replace behaviour exposed by higher layers
- Collision avoidance = lowest behavioural layers

Result:

- Passing through narrow corridors (even of high curvature) = Primary design goal
- Work with dynamically changing environment / moving obstacles when obstacle speed comparable to mover speed



Subsumptive layers[a]

Priority: Layer	Trigger condition	Output
6: KEEP-MOVING	Always	vl=vr=s
5: TARGET-CHECK	Target detection	vl, vr to target
4: FAR-RIGHT	Leaving right wall	vl=s, vr=0
3: FAR-LEFT	Leaving left wall [but not right one]	vl=0, vr=s
2: CLOSE-RIGHT	Wall close to right	vl=0, vr=s
1: CLOSE-LEFT	Wall close to left [but not to right]	vl=s, vr=0

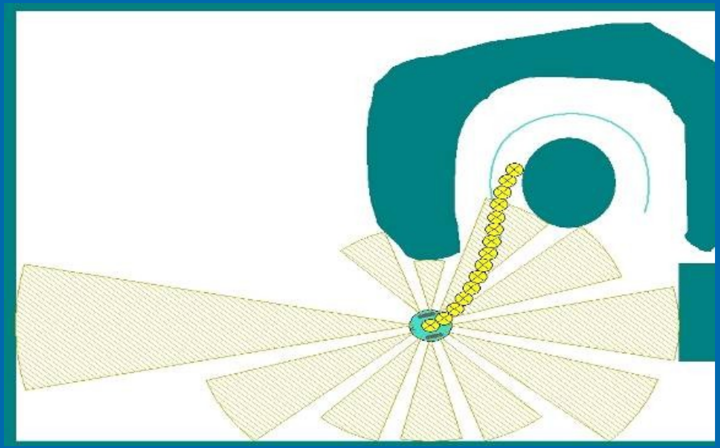
Experiments

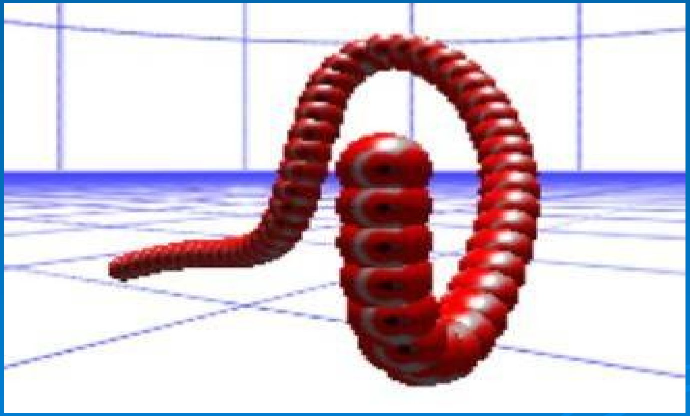
"MachineLab" testbed for 3D simulations:

- RemObjects PascalScript for Delphi / Object Pascal
- Logo to PascalScript transcoding compiler

Pascal script:

- 3D snake-like realtime motion at 450+ agents





MobotSim mobile robots 2D simulator:

SAX Basic script:

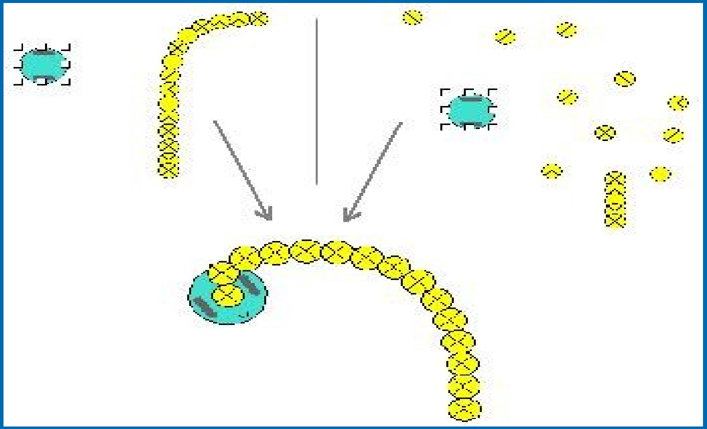
- swarm of mobile robots:
  - wandering controller robot with laser scanner
  - 15 slave robots following in snake like formation using simpler close-collision detection sensors
- obstacle avoidance in narrow curved corridors

Future work

- Design and implement angular constraints and related reactions

- Investigate/measure performance in acquiring solutions for inverse kinematics problem of physical (manipulators) or virtual (swarms) reconfigurable chain linkages.

  Observed that 2D simulation automatically provides:
  - inverse kinematics solution
  - reconstruction of chain manually broken into parts:
    - scattered parts
    - or even fewer parts
    - respecting relative formation of non-broken parts at two ends of chain



- Implement collision detection at 3D simulator:
  - minimal changes to implementation of constraint preservation rules
  - multi-agent system and event propagation design kept intact

Conclusions

Reducing problem of path planning for chain formations:

- Planning only for a master part of chain (usually tool-tip or leading mobile robot).
- Other chain parts:
  - adjust in real-time to master's motion
  - react to avoid sensed obstacles

Can scale up efficiently:

- agent only needs to sense obstacles locally
- agent interacts only with two neighbouring agents in control chain

Conclusions (more)

Best application:

- highly redundant manipulators
- reconfigurable manipulators
- swarms with high number of robots
- trying to move through narrow corridors

Not concerned with:

Veto from slave agent back-propagating to <u>original motion initiating agent</u> (rest of chain can't follow):

- autonomous mover: replan its motion
- controlled by planner module or human operator: let veto propagate further to controller

Acknowledgements

University of Patras is partner of the EU-funded FP6 Innovative Production Machines and Systems (I*PROMS) Network of Excellence.

http://www.iproms.org

References (1/2)

- [1] Hwang, Y., and Ahuja, N. Gross Motion Planning – A Survey. ACM Computing Surveys, no 3, vol 24 (1992) 219-291.
- [2] Chen, P., and Hwang, Y. Sandros, a motion planner with performance proportional to task difficulty. Proceedings of IEEE Int. Conf. on Robotics and Automation (1992).
- [3] Challou, D., Boley, D., Gini, M., Kumar, V., Olson, C. In: Kamal Gupta and Angel P. del Pobil (Eds.) Practical Motion Planning in Robotics: Current Approaches and Future Directions, 1998.
- [4] Liu, J. Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization, and Adaptive Computation. World Scientific, Singapore (2001).
- [5] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, NY, 1999.
- [6] Dorigo, M., Trianni, V., Sahin, E., Labella, T., Grossy R., Baldassarre, G., Nolfi, S., Deneubourg J-L., Mondada, F., Floreano D., Gambardella, L.M. Evolving Self-Organizing Behaviours for a Swarm-bot. Swarm Robotics special issue of the Autonomous Robots journal, 17(2-3) (2004) 223-245.

References (2/2)

- [7] Overgaard, L., Petersen, H., and Perram, J. Reactive Motion Planning: A Multi-agent Approach. Applied Artificial Intelligence, no 10 (1996) 35-51.
- [8] Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotic Research, no. 5 (1986) 90-98.
- [9] Bohner, P., and Lüppen, R. Reactive Multi-Agent Based Control of Redundant Manipulators. Proceedings of the 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, New Mexico (1997).
- [10] Birbilis, G. and Aspragathos N. In: Lenarcic J. and Galletti C. (Eds.) On Advances in Robot Kinematics. Kluwer Academic, Netherlands, 2004, pp 441-448.
- [11] Liu, J., Jing, H., and Tang Y., Multi-agent oriented constraint satisfaction. Artificial Intelligence no.136 (2002) 101-144.
- [12] Brooks, R. A. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, RA-2, April (1986) 14-23.

- Delphi / Object Pascal: www.borland.com/delphi
- PascalScript: www.remobjects.com
- MobotSim: www.mobotsoft.com