Multi-robot systems

Miroslav Kulich

Intelligent and Mobile Robotics Group
Czech Institute of Informatics, Robotics and Cybernetics
Czech Technical University in Prague

Wednesday 13/1/2016

Multi-robot systems

Systems of cooperating robots

- Knowledge about existence of other robots.
- Solving joint tasks based on a current state, actions, and other robots.
- Strong cooperation – communication, synchronization.
- Weak cooperation - periods, when robots operate independently.

Robotic swarms

- A huge number of typically homogeneous robots.
- Local control.
- Little/no explicit communication.

Is N robots better than one?

Pros

- More powerful, faster
- It is possible to solve complex tasks.
- Robustness (lower probability of failure)
- Distributed sensing
- Actions can be done at distant places in parallel.
- It can be cheaper to build a team of unsophisticated robots than a single robot.

Cons

- Robots can interfere and disturb each other.
- Communication is not for free.
- Complexity/maintenance.
- Price
- Uncertainty of knowledge about intentions of other robots.

Studied problems of multi-robot teams

- Exploration and coverage: collecting of randomly placed objects
- Formation creation/keeping, clustering: the team keeps a given geometrical shape while moving.
- Cooperative manipulation (box pushing): the team cooperatively manipulates with objects.
- Multiple object tracking: the teams moves in order to keep a set of objects in a filed of view.
- Traffic control/multi-robot path planning: action coordination in a joint space.
- Multi-robot localization: use a knowledge of other robots to localize itself.
- Multi-robot football: game/competition, where all key aspects of cooperation and coordination are needed to solve.

Architectures of multi-robot systems

Centralized

- A team is coordinated from a single “center”.
- Vulnerability to failure of the central point.
- The central point has to keep information about the current state (communication needed).
- Applicable when the central point can be placed appropriately.

Hierarchical

- Inspired by organization of military units.
- Bad recovery from failures/unavailability of a robot, which is at a high level of hierarchy.

Architectures of multi-robot systems

Decentralized

- Robot's decisions are typically based on local knowledge.
- Nobody is responsible for control of another robot.
- Goals have to be included in local control of every robot.
- The most widely used architecture.

Hybrid

- Combination of local control and high-level control.
- Robustness
- Potentiality to influence behavior of the whole team globally.
- Widely used.

Communication

- Aim: to enable robots to exchange information about their inner states and about surrounding environment with minimal resources (transmission medium) used.
- Implicit communication - via the environment
  - Observation of environment changes, which are consequences of other robots' activities.
  - Observation of actions of other robots.
- Explicit communication - information transmission directly among robots.
  - Includes: occasional requests, information about inner states, sensor measurements
  - Must be specified: WHAT, WHEN, HOW, WITH WHOM to communicate
  - Communication medium: range, bandwidth, reliability
- Examples:
  - Pheromones ants
  - „Posing“ of animals during mating, before the fight
  - Writing, drawing

Is communication needed?

- Good to know:
  - Communication is not for free and can be unreliable.
  - It can be disturbed in hostile environments.
- Role of communication:
  - Actions synchronization - coordination in action ordering
  - Information exchange - information sharing obtained from various sources and perspectives.
  - Negotiation - what will be done and by whom.
- Studies show:
  - Explicit communication improves team efficiency in case of little implicit communication.
  - Explicit communication is not needed when implicit communication is available.
  - Complex communication strategies bring a little benefit in comparison to standard communication.

Communication range

Categories

- No communication.
- Limited range (local communication).
- Unlimited range.

Properties

- General idea: unlimited communication range is better.
- Valid for small teams, but does not hold for big teams:
    - Efficiency of information transfer is small if a single communication medium is used by many robots (radio).
    - A central element providing communication is a weak point and increases possibility of failure and error rates.
    - Local communication is enough to plan a next action in majority of cases.

Communication topologies

Categories

- Broadcast - everyone hears what was sent.
- Addressed (peer-to-peer)
- Tree (supervision)
- Graph - more general, more robust

Properties

- Tree and pear-to-pear are prone to failures of particular robots.
- Addressing leads to differentiation (and therefore to more complicated interchangeability) of robots $\rightsquigarrow$ dynamic change of a role (auction).
- A set of robots able to communicate is defined by:
  - communication topology,
  - communication range, and
  - displacement of the robots in an environment.

Communication - bandwidth

Categories

- High (unlimited) - assumption in theoretical reasoning
- Equal to movement - communication and movement prices are comparable (bee's dance)
- Low - price of communication is high (highly independent robots)
- None

Properties

- A robot can either communicate or solve a task in some cases.
- Low bandwidth is acceptable if the reason to use multiple robots is redundancy instead of efficiency.

Team structure

Identical

- Both hardware and software are identical.
- Robots can have different roles (depending on an environment and chance).
- Robots can have a unique identification (deterministically assigned).

Homogeneous

- Identical hardware, different software.

Heterogeneous

- Physically different robots.

Task allocation

- Task allocation is a problem how to define which task(s) will be performed by which robot(s)
- Assume n robots $\{r_1, r_2, \ldots, r_n\}$
  m tasks $\{t_1, t_2, \ldots, t_m\}$
- Goal: to find a mapping of tasks onto robots so that every tasks is done by best possible way.
- Majority of specific problems is NP-hard.
- Taxonomy: (Gerkey, Mataric, 2004)
  - Task: single-robot (SR) $\times$ multi-robot (MR)
  - Robots: single-task (ST) $\times$ multi-task (MT)
  - Assignment: instantaneous (IA) $\times$ time extended (TA)
- Combination of the criteria into a uniform description:
  - SR-ST-TA: single-robot, single-task, time extended assignment.
  - MR-ST-IA: multi-robot, single-task, instantaneous assignment

ST-SR-IA

- Instance of the Optimal Assignment Problem (OAP)
- The simplest problem, studied in operational research (Gale 1960)
- Formulation: Given m workers, each looking for one job and n jobs with a priority w_j each requiring one worker. Also given for each worker is a nonnegative skill rating (i.e., utility estimate) U_{ij} that predicts his/her performance for each job. The goal is to assign workers to jobs so as to maximize overall expected performance, taking into account the priorities of the jobs and the skill ratings of the workers:

$$U = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} U_{ij} w_j \quad \text{so that}$$

$$\sum_{i=1}^{m} \alpha_{ij} = 1, 1 \leq j \leq n \quad \text{and} \quad \sum_{j=1}^{n} \alpha_{ij} = 1, 1 \leq i \leq m$$

ST-SR-IA - approaches

- Linear programming - Hungarian method, Kuhn(1955), complexity $O(mn^2)$
- Solving a dual problem, Gale (1960):

$$P = \sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j, \text{ where}$$

$$u_i + v_j \geq U_{ij}, \forall i, j$$

- Aucion: a broker offers tasks, robots choose $t_i = \arg\max_j \{h_{ij} - p_j\}$
- BLE - Broadcast of Local Eligibility (iterative assignment):
  1. If any robot remains unassigned, find the robot-task pair (i, j) with the highest utility. Quit otherwise.
  2. Assign robot i to task j and remove them from consideration.
  3. Go to step 1.

ST-SR-IA - approaches

- On-line assignment (MURDOCH) – tasks are randomly injected into the system over time:
    1. When a new task is introduced, assign it to the most fit robot that is currently available.
- Without a model of the tasks that are to be introduced, and without the option of reassigning robots that have already been assigned, it is impossible to construct a better task allocator than MURDOCH.

ST-SR-TA

- The system consists of more tasks than robots, or
- We have a model of how tasks will arrive.
- Robots' future utilities for the tasks can be predicted with some accuracy.
- Problem (scheduling) is NP-hard => approximation algorithm:
    1. Optimally solve the initial $m \times n$ assignment problem (OAP).
    2. Use the Greedy algorithm to assign the remaining tasks in an online fashion, as the robots become available.
- The more tasks that are assigned in the first step, the better this algorithm will perform.
- Other approaches: iterative task allocation: price-based market (Dias, Stentz, 2001), in exploration: Zlot et.al (2002)
- Without knowledge of the exact criteria used to decide when and with whom each robot will trade, it is impossible to determine the algorithmic characteristics (including solution quality) of this method.

ST-MR-IA

- A single task requires several robots => *combined* utility of a set of robots.
- Multi-agent approaches - coalition formation.
- Set theory - Set Partitioning Problem (SPP): Given a finite set E, a family F of acceptable subsets of E, and a utility function $u : F \rightarrow \mathbb{R}_+$, find a maximum-utility family X of elements in F such that X is a partition of E.
- NP-hard, but good heuristics exist (problems with hundreds of tasks can be solved in seconds).
- Large number of „coalitions“ has to be enumerated => pruning is necessary (mutual distances of robots).

ST-MR-TA

- coalition formation + scheduling
- NP-hard
- Example: delivering a number of packages of various sizes from a single distribution center to different destinations.
- Solution: ST-MR-IA + a greedy algorithm for the rest.
- Result quality is unpredictable.
- Another approach: Several leaders dynamically form coalitions and build task schedules for them.

MT-SR-IA a MT-SR-TA

- These problems are not important in robotics.
- Can be solved be as ST-MR-IA a ST-MR-TA (with interchanging robots and tasks).

MT-MR-IA

- Example (surveillance): each robot continuously patrols a fixed portion of the building, but can simultaneously detect only a limited number of environmental events (e.g., suspicious person, smoke, open door).
- Set Covering Problem from the Set theory:
  Given a finite set E, a family F of acceptable subsets of E, and a cost function $u : F \to \mathbb{R}_+$, find a minimum-cost family X of elements in F such that X is a cover of E.
- Contrary to partitioning, sets in covering are not necessarily disjunctive.
- There exist heuristics working well when the space of feasible subsets is limited.
- MT-MR-TA - nobody is interested in this problem.

Coordination of multiple robots

- Assume p robots, every with n DOF.
- Planning methods are same for both mobile robots and manipulators.

Centralized planning

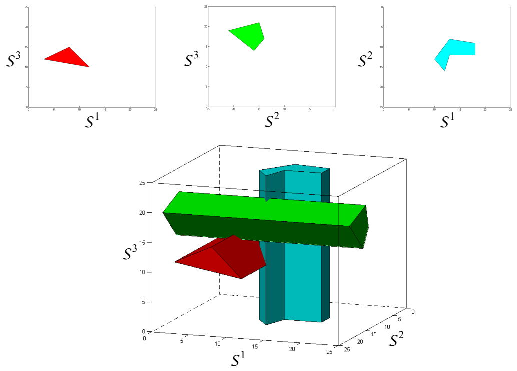
- Multiple robots are assumed as a single robot with many DOFs.
- Planning is done in a composite C-space.
- Complete, i.e. it always finds a solution (if it exists).
- Complexity $\approx e^{pn}$.
- Cann't be applied, when central knowledge about robots is not available.

Decoupled planning

- Planning for each robot separately.
- Correction of particular plans (coordination) is done later on.
- Complexity $\approx pe^{n}$.
- Not complete, not optimal.

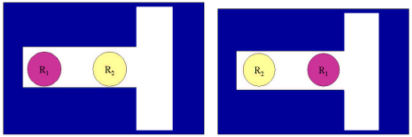
Centralized planning

- Planning of obstacle-free path τ v $C_1 \times C_2 \times \cdots \times C_p$.
- Obstacles in a composite C-space are all configurations, where a robot collides with some obstacle or with another robot.
- Projection τ onto C_i is a path of i-th robot.



Decoupled planning - phase two

- Speed tuning.
    - Every robot performs a trajectory generated in the first phase.
    - The robot can stop, change speed, or go back.
    - Pairwise planning (coordination)
    - Global coordination
- Prioritized planning

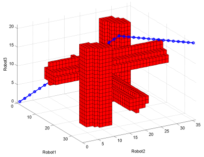


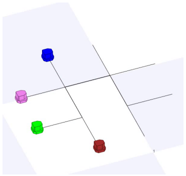
Decoupled planning - pairwise cooperation

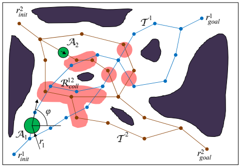
- Trajectories τ_1 and τ_2 of first two robots are coordinated in a 2D coordination space.
- Robot configuration is a one-dimensional (position on a trajectory in time).
- Path planning between $(0, 0)$ a $(1, 1)$.
- Result is a collision-free coordinated trajectory $\tau_{1,2}$.
- Coordination of $\tau_{1,2}$ and τ_3 is done next.
- The process is repeated until $\tau_{1,2,\dots p}$ is determined.

Decoupled planning -global coordination

- P-dimensional coordination space is created.
- Path planning between $(0, 0, 0, \dots)$ and $(1, 1, 1, , \dots)$.
- Simplification: a discrete coordination space, discrete configuration trajectory.
- It is possible to extend for the case robots move on a graph.

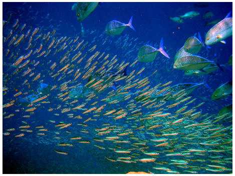






Movement coordination

- With respect to other robots: formation keeping, flocking, clustering, dispersal
- With respect to an environment: exploration, search, foraging, coverage
- With respect to external factors: pursuit, object tracking
- With respect to other robots and an environment: patrolling
- With respect to other robots, external factors, and an environment: robotic soccer

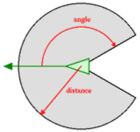


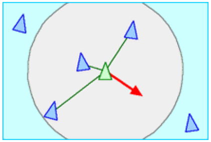
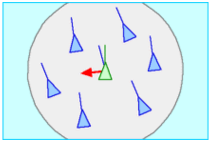
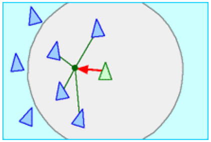
zdroj: http://en.wikipedia.org

Boids (Craig Reynolds)

Basic types of behavior (steering behaviors) based on local information:

- Separation - steer to avoid crowding local flockmates
- Alignment - steer towards the average heading of local flockmates.
- Cohesion - steer to move toward the average position of local flockmates



The Nerd Herd

- Principle: basic behaviors as building blocks for complex strategies.
- Basic behaviors:
  - Avoidance
  - Save-wandering
  - Following
  - Aggregation
  - Dispersion
  - Homing
- Complex behaviors:
  - Flocking
  - Foraging

Safe-Wandering algorithm

Avoid-Kin

- Whenever an agent is within d_{avoid}
    - If the nearest agent is on the left
        - Turn right
        - Otherwise, turn left

Avoid-Everything-Else

- Whenever an obstacle is within d_{avoid}
    - If obstacle is on right only, turn left
    - If obstacle is on left only, turn right
    - After 3 consecutive identical turns, backup and turn
    - If an obstacle is on both sides, stop and wait.
    - If an obstacle persists on both sides, turn randomly and back up

Move-Around

- Otherwise move forward by $d_{forward}$, turn randomly

Following algorithm

Follow

- Whenever an agent is within d_{follow}
    - If an agent is on the right only, turn right
    - If an agent is on the left only, turn left

Dispersion algorithm

Dispersion

- Whenever one or more agents are within $d_{disperse}$
    - Move away from $centroid_{disperse}$

Aggregation algorithm

Aggregation

- Whenever nearest agent is outside $d_{aggregate}$
    - Turn toward the local $centroid_{aggregate}$, go.
- Otherwise, stop.

Homing algorithm

Home

- Whenever at home
    - Stop
- Otherwise, turn toward home, go.