SENSOR NETWORK BASED CONFLICT RESOLUTION IN AUTONOMOUS MULTIAGENT SYSTEMS

Witold Jacak, [(a)] **Karin Pröll** [(b)]

[(a)] Department of Software Engineering
Upper Austria University of Applied Sciences
Hagenberg, Softwarepark 11, Austria
[(b)] Department of Bioinformatics
Upper Austria University of Applied Sciences
Hagenberg, Softwarepark 11, Austria

[(a)] Witold.Jacak@fh-hagenberg.at,[(b)] Karin.Proell@fh-hagenberg.at

ABSTRACT

The design of intelligent and sensor-based autonomous agents learning by themselves to perform complex real-world tasks is a still-open challenge for artificial and computational intelligence. In this paper a concept of a framework for an autonomous robotic agent is presented. The structure of an intelligent robotic agent consists of two independent subsystems: the action and motion planning system and the action and motion reactive control system with integrated conflict resolution methods. The action planning system uses an aggregated world model storing knowledge about all static and dynamic objects in the surrounding environment. The action controller solves space conflicts in a reactive manner making use of information from local sensors and a distributed sensor network. Each dynamic object registered from sensor network field is inserted into the world model as a new obstacle in 2,5D form. Based on the updated world model a conflictfree robot motion is calculated in a one step motion planning cycle.

Keywords: autonomous robotic agent, sensor-based conflict resolution

1. INTRODUCTION

The design of intelligent and sensor-based autonomous systems (agent type) that learn by themselves to perform complex real-world tasks is a still-open challenge for the fields of system and control theory, robotics and artificial and computational intelligence.

In this paper we present the concept of a framework for an autonomous robotic agent that is capable of showing both local sensor-based reactive behavior and global action planning based on external sensor network. Past experience has shown that neither purely reactive nor purely machine learning-based approaches suffice to meet the requirements imposed by real-world environments.

In multi-agent robotic systems, one is primarily interested in the behavior and interactions of a group of agents and a dynamic surrounded world, based on the models of the agents themselves and environment stimuli. With every perceptual input, one associates a certain action that is expressed in the form of rules or procedures that calculate the reaction of the agent. Reactive systems have no internal history or long term plans, but calculate or choose their next action based solely upon the current perceptual situation.

On the other hand, machine learning-based models are motivated by the representation of the system's knowledge. The adaptation of symbolic AI techniques has led to the introduction of believes and intentions into the reasoning processes of the system. Such models permit to use more powerful and more general methods than reactive models; this, however, makes them inadequate for many real-time applications where a dynamic change in the environment occurs.

Usually an agent has only partial information about the world state obtained from own perception system (sensors system). Machine learning aims to overcome the limitations such as knowledge bottleneck, engineering and tractability bottleneck, by enabling an agent to collect its knowledge on-the-fly, through real-world experimentation. Processing, storing and using information obtained during several task executions is called lifelong learning. For this reason it is necessary to extend the reactive control system of sensor-based global preplanning system by omitting the knowledge bottleneck in classical machine learning approach.

2. CONFLICT RESOLUTION SYSTEM FOR ROBOTIC AGENT

In our concept, the structure of an intelligent robotic agent consists of two independent subsystems: the action and motion planning system and the action and motion reactive control system with integrated conflict resolution method (Jacak, Proell, and Dreiseitl 2001, Jacak and Proell 2007). The action planning system uses an aggregated world model storing knowledge

about all conflicts which occurred in the past. Conflicts occur when a collision between a robotic agent and an unknown dynamical object or another agent in work space is possible. The action controller is able to solve the conflict situation (space conflict for autonomous robotic agent) in a reactive manner. The general conflict resolution part of the agent makes use of knowledge not only from sensors mounted on active agent (local sensor network) but also from distributed global sensor network (see Figure 1). A sensor network is a collection of sensor nodes deployed in an adhoc fashion in the work space.

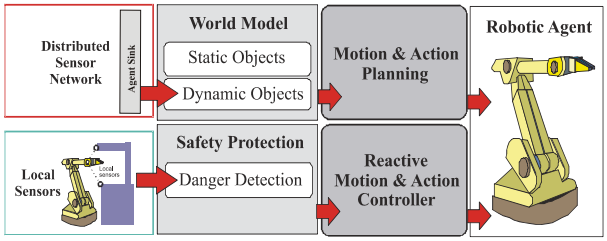

Figure 1: Structure of sensor-based conflict resolution system

Being battery powered and deployed in remote areas they have limited energy resource and hence limited lifetime. Other constraints include limited memory, processing power, and band-width. The accuracy of information is location dependent. Due to these limitations data aggregation is an important consideration for sensor networks. The idea is to combine the data coming from different sources and reroute it further, after eliminating redundancy, minimizing number of transmissions and thus saving energy.

Sensors will be used for inventory maintenance and unknown object recognition and motion tracking. The agent requests a monitoring of some segment of space in which the next parts of motion are supposed to take place from the global sensor network. If the global sensor network identifies unknown objects in these segments then this information can be used by the agent's safety system to preplan a reaction prior to recognition of these objects by the local perception units. This can help to avoid possible conflict situations in advance.

3. SENSOR NETWORK BASED ONE STEP MOTION PLANNING SYSTEM

3.1. Static and Dynamic World Model

The knowledge represented here is the geometrical model of the robotic agent environment. Many different methods can be used for geometrical representation of the agent service space. One of them is the triangle approximation another is cubic approximation.

In a model with triangle approximation, the points in the triangle net (lying on service space border) are coupled in triangle walls. The walls represent the data objects of the world model.

The other model describes the service space of the robot manipulator as cubic approximation. The service space of robotic agent can be discretized in the form of the cubic raster (voxels). The number of voxels depends of the accuracy of approximation. If the voxel is occupied by an obstacle then it obtains the value 1 of the space occupancy function.

The level of fullness of knowledge leads to two different methods of resolving of conflict situation. The both models are convenient to represent geometrical environment of known objects in agent workspace. The model is used for collision-freeness testing between agents and surrounding world. We can say that the position of an agent is collision free if it does not collide with any static or dynamic obstacle in its workspace. To test such conditions we should have full knowledge about the surrounding static and dynamic world, that means that the geometrical model of agent's environment should be completely known. To obtain fast and fully computerized methods for collision detection, we use additional geometric representation of each object on the scene.

We introduce the ellipsoidal representation of 3D objects, which uses ellipsoids for filling the volume. The ellipsoidal representation of an object is convenient to test collision freeness of agent positions. The 3D models of objects represent the static part of agent's world model. To construct the model of unknown objects, which penetrate the agent's environment, it is necessary to continuously modify the geometrical representation. The dynamic part of model is modified based on sensor data coming from wireless sensors network.

3.2. Sensor Network

A sensor network is composed of a large number of tiny autonomous devices, called sensor nodes. Each sensor node has four basic components: a sensing unit, a processing unit, a radio unit, and a power unit. Since a sensor node has limited sensing and computational capabilities and can communicate only within short distances. The nodes are deployed densely and coordinate amongst themselves to achieve common information (Karl and Willig 2005). Some examples of sensor network applications are as follows:

- *Intrusion detection and tracking*: Sensors are deployed along the border to detect, classify, and track intruding objects (Dousse, Tavoularis, and Thiran, 2006).
- *Environmental monitoring:* Specialized sensor nodes that are able to detect changes in environment (Tzung-Shi Chen, Yi-Shiang Chang, Hua-Wen Tsai, Chih-Ping Chu 2007).

These sensor networks applications differ significantly. However, the tasks performed by the sensors are similar: sensing the environment, processing the

information, and sending information to the base station(s). A node in a sensor network has essentially three different tasks (Al-Karaki and Kamal 2004):

1. Sensing: detecting changes of environment;
2. Communicating: forwarding information, acting as an intermediate relay in a path;
3. Computing: data aggregation, processing, and compression.

Routing techniques are needed to send data between sensor nodes and the base station. Several routing protocols are proposed for sensor networks. These protocols can be divided into the following categories: data-centric protocols, hierarchical protocols, location based protocols, and some QoS-aware protocols (Dousse, Tavoularis, and Thiran, 2006).

For monitoring the surrounding environment of the robotic agent we propose a sensor network based on a virtual grid representation of the monitored area and for data delivery a combination of event driven and query-driven data delivery protocol between sensor network and a mobile sink component (AS) of the robotic agent is used (Tzung-Shi Chen, Yi-Shiang Chang, Hua-Wen Tsai, and Chih-Ping Chu 2007). The mobility of AS results from movements of the robotic agent.

Virtual Grid Structure of Sensor Network: The monitored area is divided into virtual grids. We notate *G(x,y)* as the grid coordinates, where *x* is grid x-coordinate and *y* is grid y-coordinate in Cartesian space. Let *R* be the transmission distance of the radio signal and *d* be the side length of grid. After sensors have been deployed, a node is selected to act as grid head. The grid head's task is to record information about events and disseminate it to other nodes for collaborative signal and information processing. At first, each node obtains neighboring information by start message. Utilizing this information, a node that is closest to the center of the grid is selected as head. If a head has not enough resources, one of other nodes in the same grid will be selected to replace it.

We assume that the side length of grids to guarantee that the grid heads can communicate with neighboring grids directly. Here, we also assume that the communication range of node is able to communicate with the neighboring grids. The relationship between *d* and *R* is predefined.

The grids are grouped in sub-networks in hierarchical way. Each sub-network obtains one head selected from the set of grid's heads. The sub-network head collects the messages from grid's heads within the sub-network.

The mobile agent sink (AS) stores the topology of the virtual grid and uses this information to perform the queries to the sensor network.

The task of AS is to find the virtual cell of grid where the intrusion of a new dynamic object is registered. An AS expects to obtain the event information instantly when such event occurred. An event usually happens unexpected. Therefore, a sensor has to signal an event after it was detected. This sensor is called source node. The source node propagates a register packet to all grid heads. The format of register packet is *<P_type, Src_id, Src_G(x,y), hc, event_type, time_to_expire>*, where *P_type* is packet type, *Src_id* is the identifier of the source node, *Src_G(x,y)* is the source's grid location, and *hc* is the hop count. When heads receive this packet, they store the register packet in their register table and route it to the sub-network head. This information is kept within a certain period of time (*time_to_expire)*. If a head does not receive any further register packets and *time_to_expire* is elapsed, it removes the information from the register table.

The information of an event is distributed to the grid heads and summarized to the sub-network head in the following way: The grid heads store the *x,y* positions of active sensor nodes (an event has been signaled) whereas in the subnet-work heads only grid numbers with active sensor nodes are registered (active grids).

For the next query the AS maintains a list containing all sub-network heads surrounding the next segment of its motion path and all sub-networks with active grids.

AS can now decide to obtain detailed information about all active sensor nodes by querying the grid heads of all active grids or summarized information about all active grids by querying only the sub-network heads. For reconstructing the geometric model of the intruding object both - detailed or summarized - information can be used. Using detailed information a more accurate shape of the base of the intruding object can be deduced. The use of summarized information leads to a very approximate representation based on shapes of active grids. (see Figure 2)

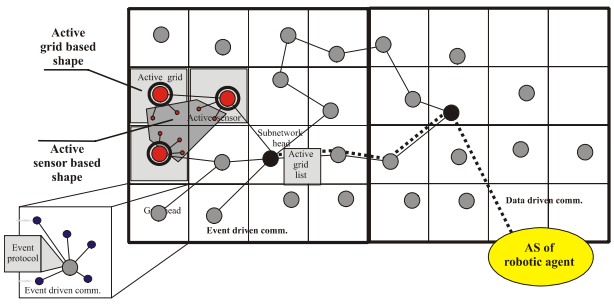

Figure 2: Use of summarized or detailed information from virtual grid for reconstructing shape of intruding object

By maintaining a dynamic list of sub-network heads for querying, a general broadcast to all sub-network heads in the virtual grid can be avoided. The possibility to query either sub-network heads or grid heads further reduces transmission activities in the network.

World Model Updating: The path planner of robotic agent sends a query to AS to check if any grid cell in

sensor network has an active signal of a new object intrusion. AS uses the sub-network heads and grid heads and hierarchical protocol to collect the *G(x,y)* position of grids to be active. The *G(x,y)* and side length *d* will be used to approximation of object shape and volume to be intruded in the work space of robotic agent. The 3D geometrical model of an intruding object is constructed in a 2,5D representation of the shape registered by sensors (see Figure 3).

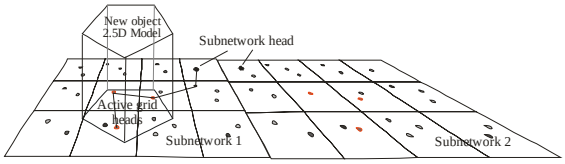

Figure 3: Virtual Grid Structure of Sensor Network and 2,5 D representation of intruding object

3.3. Global Motion Planner

Agent Model: The most suitable model of hardware component of robotic agent is a discrete dynamic system. One of the ways to construct such model of robot's kinematics is based on an arbitrary discretization of angle increments of the agent mechanical joints (Jacak 1999). Using the fact that all the angles can change only by a defined increment, we define the input set U of the model as: $U= \times\{u_i|i=1,..,n\}$ where $u_i =\{-\delta q,0,\delta q\}$ is the set of possible (admissible) directions of changes of the *i-th* joint angle. Having defined the set *U*, it is possible to describe the changes of successive configurations of the agent's link as discrete linear system with the state transition function as:

$$q(k+1)=q(k) + \Lambda\, u(k) \qquad (1)$$

In order to make it possible to check the configuration with respect to obstacles locations, it is necessary to create an output function *q*. A skeleton of agent´s arm represents the agent position in the base frame. The *i*-th joint's position in Cartesian base space (skeleton point), assuming that all the joint variables *q* are known, is described by the Denavit-Hartenberg matrix. Checking for the collision-freeness of the agent configuration (skeleton) can be reduced to the "brokenline ellipsoid" (skeleton ellipsoid) intersection detection problem, which has an easy analytical solution. The complete formal explanation of the FSM model of agent kinematics, is presented in (Jacak 1999, Proell 2002).

Motion Planner: For the robotic agent we can define the problem of achievement the goal position as the problem of reachability of the final state set from the agent's current state (current position).

In order to solve this problem we apply graph searching in the state transition graph. The process of applying the transition function to a current state we term expanding the graph node. Expanding current state *qc*, successors of *qc* etc. ad infinitum, makes explicit the graph that implicitly is defined by current state and transition function. The way of expanding the graph will depend on the form of the cost function using to evaluate each node. As the evaluation function we can use the sum of the cost function *c(qc,q)* and a cost estimate function *h(q,qf)*, for example the rectilinear distance between agent position and terminal position *qf*.

Using the standard A* procedure we can find the state trajectory (if exists) q*=*(qc, q(2),..,q(k),.., qf)* from current state to final state *qf* which includes only feasible states. In order to solve the path-planning problem we apply the graph searching procedure to the agent's state transition graph. The development of the search graph will start from the node (configuration) *qc*, by the action for all possible inputs from the set U. Thus, it becomes essential to quickly check for the non-repeatability of the nodes generated, and their feasibility. A configuration *q* is said to be feasible if it does not collide with any object in the surrounding world.

3.4. One step motion path planning

The continuously adapted world model is used to test the collision freeness. Each dynamic object registered from sensor network field is inserted into the world model as a new obstacle in 2,5D form.

The task of the motion planning and execution component is to plan the collision free configuration of the robot's manipulator based on information coming from the world model from knowledge base. To realize this task, the on line motion planner calculates the changes of robot configuration to avoid the obstacle in the best possible way. This knowledge is represented as geometrical model of work scene and mathematical model of agent's actor direct and inverse kinematics.

The one step action planner of the agent generates the new movement in following steps:

step 1: The motion planner requests the current world model from AS of sensors network,

step 2: Based on the positions of obstacles motion planner recognizes its own current position and establishes the parameters for motion search algorithm, such as type of evaluation function.

step 3: Action planner starts the stategraph searching algorithm A* from its current position with previously established evaluation. The searching stops if the goal position is reached or if the OPEN set is empty or if the OPEN set has more as N elements. To calculate the successors set planner uses the FSM - model of the agent.

step 4: The temporary path of motion is calculated and the new configuration of motion is chosen and realized. The motion planner started step 1 again.

3.5. Example

The following example (Figure 4) shows the trajectories of the robotic agent at times T_1, T_2, …T_8 avoiding the cylindrical object crossing its initial path. At each point of time T_i the motion planner uses the information about

the obstacle's current position from the sensor network for recalculating a new collision free motion as a temporary path from current to final position. When the object leaves, the robotic agent does not return to its initial trajectory at time T_1 but takes the shortest way from the current to the final position.

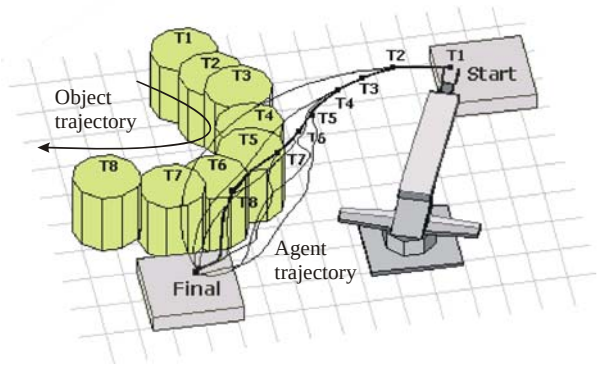

Figure 4: One step motion planning to avoid collision with cylindrical object

4. REACTIVE LOCAL CONTROLLER OF MOTION

4.1. Neural network based agent model

The agent model contains the knowledge about construction, properties and structure of a hardware agent. Here, knowledge of the kinematical properties of the robotic agent is provided in order to decide about collision avoidance mode and avoidance path. Therefore, the forward and the inverse kinematics models of the robot should be known.

The planning of the new configuration is based on the computation of robot kinematics and is computationally expensive. Therefore it is attractive to apply a neural network model of robot kinematics, stored in the knowledge base of the agent, which automatically generates safe configurations of the robot. This model uses a multilayer feedforward neural network with hidden units having sinusoidal activation functions (Jacak 1999).

Each element of manipulator direct kinematics can be represented in the form of $\Pi^{i=1}_{n}\, a_i\, sc(q_i)$ where $sc(q_i)$ is either $sin\ q_i$, $cos\ q_i$ or 1. Then, after simplification, the forward kinematics of a robot manipulator with n revolute joints can be described by the weighted sum of sinusoidal functions: $t_i^s(q)=\Sigma\, sin(w_j^T q)$ and $s=x,y,z$ where $t_i^s(q)$ defines the output of neural network representing s-th Cartesian variable of i-th joint position. Based on the neural direct kinematics model it is easy to generate the inverse kinematics (Jacak 1999, Proell 2002). The plant model of the robot kinematics is presented in Figure 5.

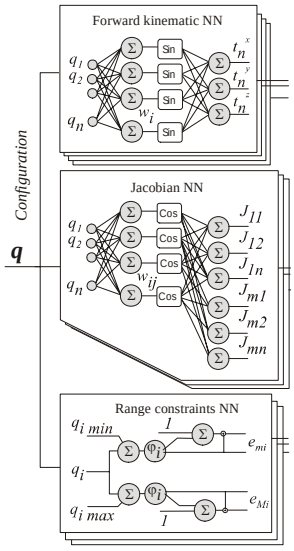

Figure 5: Neural network model of the robot kinematics

4.2. Safe local motion planning and execution

To calculate the safe configuration, the planner uses information from the local sensors of danger recognition component and combines it in the inverse kinematics computation. We use the method, which requires only the computation of direct kinematics based on neural processing. Such the solution of inverse kinematics problem can be obtained by attaching a feedback network around a forward to form a recurrent loop, such that, given a desired Cartesian pose P of a feedforward network, the feedback network iteratively generates joint angle correction terms to move the output of forward network toward the given pose. This coupled neural network is the neural implementation of a gradient method of position error minimization.

Let $e_n(q)$ denote the position-error between the current position of effector-end in current configuration q, calculated by forward kinematics and a desired next Cartesian position on the executed path. The obstacle avoidance can be achieved by introducing the additional errors for each joint, i.e. the errors between virtual points p_i and the joints positions, where the virtual points represent the wished position of the i-th link that achieves collision-avoidance. The virtual points are placed on the opposite side of the joint with respect to the obstacle. The virtual points are calculated based on signals from local sensor system, mounted directly on the robotic agent. To solve the inverse kinematics problem in our particular case we transform it to the optimisation problem. Note, that the solution of the inverse kinematics problem uses only direct kinematics models, and Jacobian. Both models can be implemented as neural networks (Jacak 1999).

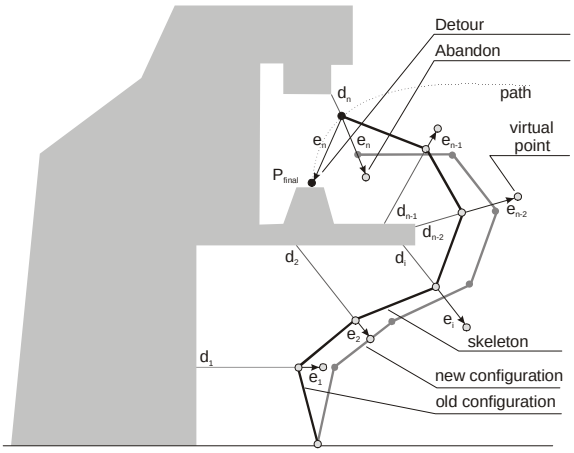

Figure 5: virtual points and reactive motion control

For calculation of virtual points, we propose the installation of ultrasonic sensors and a sensitive skin on each manipulator link and then use a neural network to estimate the proximity of the objects with the link of question. The resulting distances are compared with the world model of the robot environment to recognize the new obstacle within the radius of the security zone ρ.

When a manipulator is equipped with many sensors and these sensors are mounted on different manipulator links the problem arises how to fuse sensors readings to obtain useful information. The results of the fusion action are vectors $d=(d_i|i=1,..,n)$, representing the minimal distances to objects in robot environment for each joint of the manipulator. The distances are used to calculate the virtual points for online inverse kinematics method. When the obstacle penetrates the local security zone the virtual points are calculated and additional errors are introduced into inverse kinematics. The reactive motion controller calculates the new position of agent which minimizes additional errors i.e. maximizes the distances d to obstacles. (see Figure 5)

CONCLUSION

Using global sensor network unknown objects in a robotic agent's workspace can be identified prior to recognition of these objects by the local perception units. This information is stored in the world model of the robotic agent which is continuously updated. The aggregated world model is used by the agent's safety system to preplan a reaction to avoid possible conflict situations. Especially for mobile robots with a manipulator fixed on top of a mobile base unit the combination of a global sensor network and local perception units improves their mobility.

REFERENCES

Al-Karaki, J.N. Kamal, A.E. 2004. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE* Volume 11, Issue 6:6- 28.

Dousse, O., Tavoularis, C., and Thiran, P. 2006. Delay of intrusion detection in wireless sensor networks. *Proceedings of the 7th ACM international Symposium on Mobile ad hoc Networking and Computing*, 155-165. May 22 - 25, 2006, Florence, Italy.

Jacak W., Proell K., 2007. Heuristic Approach to Conflict Problem Solving in an Intelligent Multiagent System Heuristic Approach to Conflict Problem Solving in an Intelligent Multiagent System. In: *Computer Aided Systems Theory — EUROCAST 2007, Lecture Notes in Computer Science*. Heidelberg:Springer, 772-779.

Jacak W., Proell K., Dreiseitl S., 2001. Conflict Management in Intelligent Robotic System based on FSM Approach. In: *Computer Aided Systems Theory — EUROCAST 2001, Lecture Notes in Computer Science*. Heidelberg:Springer, 359-386.

Jacak, W., 1999. *Intelligent Robotic Systems: Design, Planning and Control.* New York, Boston, USA: Kluwer Academic/Plenum Publishers.

Karl H, Willig A., 2005. *Protocols and Architectures for Wireless Sensor Networks*. USA: Wiley.

Proell K., 2002. *Intelligent Multi-Agent Robotic Systems: Contract and Conflict Management*. PhD Thesis, Johannes Kepler University Linz /Austria.

Tzung-Shi Chen, Yi-Shiang Chang, Hua-Wen Tsai, Chih-Ping Chu, 2007. Data Aggregation for Range Query in Wireless Sensor Networks. *Journal of Information Science and Engineering* Volume 23, Number 4:1103-1121.