Journal of ELECTRICAL ENGINEERING, VOL. 60, NO. 3, 2009, 143–148

MULTI–ROBOT CONTROL SYSTEM FOR PURSUIT–EVASION PROBLEM

Daniel Hladek — Ján Vaščák — Peter Sinčák *

We propose a hierarchical multi-agent control system with a rule based fuzzy system for a pursuit-evasion problem. We state a new representation of this type of problem that is based on fuzzy logic. We can express parts of the available space as fuzzy relations and assign them a name. Fuzzy logic allows easy expression of rules and the multi-agent structure supports separation of team and individual knowledge. An example application domain includes reckon and guard robots, research space probes, coordination of multiple mine sweeping devices or automatic rescue teams.

K e y w o r d s: control of multiple robots, pursuit-evasion games

INTRODUCTION

Cooperation of multiple robots executing common task might be difficult. It requires ability of the robots to deal with unknown and uncertain situation by themselves and on the other hand take into account success of the whole team.

An example of such a task is a pursuit-evasion problem. In this type of game a group of mobile agents pursues other agents. The environment is usually filled with obstacles and thus the pursuers must avoid them. Besides avoiding obstacles they also need to catch the evading agents in the shortest possible time. We can see, that fulfilling this objective can serve as a good test-bed for solving more difficult missions involving multiple agents.

We propose a multi-agent control system based on a fuzzy inference system for a group of two wheeled mobile robots executing a common task. Possible applications of our approach are in the control of robotic formations moving on the plane such as a group of guard robots taking care of one area and dealing with potential intruders [7].

2 PROBLEM DESCRIPTION

2.1 Pursuit Evasion Problem Definition

According to the available literature, pursuit-evasion game is defined as follows [7]

Pursuit-evasion games are a mathematical abstraction arising from numerous situations, which address the problem of controlling a swarm of autonomous agents in the pursuit of one or more evaders.

We can see that this definition states the pursuit-evasion problem as an abstract mathematical problem. Many existing papers (*eg* [9, 1]) solve the problem of pursuing agents from this perspective. 7 However, we think that besides a general solution suitable for any possible map it is also important to provide a working approach, capable of deployment on real robots, sufficiently dealing with just one given environment.

From the robotic perspective, we can formalize pursuit-evasion problem as

Pursuit-evasion is a problem of finding trajectories for a group of agents in known environment such that one of the agents is able to approach the critical distance from one randomly moving target in the guarded area in the lowest possible time

If the map of the area is available and we have some other preliminary information about the places of potential intrusion, the problem is to find the trajectory for every robot that will lead to catching the intruder in the shortest possible time.

2.2 State of the Art

The existing approaches can be basically divided according to the internal representation of the world state - positions of the pursuers, evaders and obstacles.

- **Graph representation** [5] The pursuit-evasion happens in an oriented graph. Nodes represent available agent positions and arcs are connections between them. Then the problem is solved using graph search methods. This is the most basic and historically first form. The evader is caught, when it occupies the same node as the pursuer.
- **Polygonal representation** [9, 1] Obstacles and border in the area are expressed as polygonal objects. Pursuers can move along the arcs with a given speed. The area that can contain the evader is called "contaminated".

  Every pursuer has some type of visibility region. The goal of this game is to "clear" the whole area, by

* Center for Intelligent Technologies, Department of Cybernetics and Artificial Intelligence, Technical University Košice, Slovakia, daniel.hladek@tuke.sk

ISSN 1335-3632

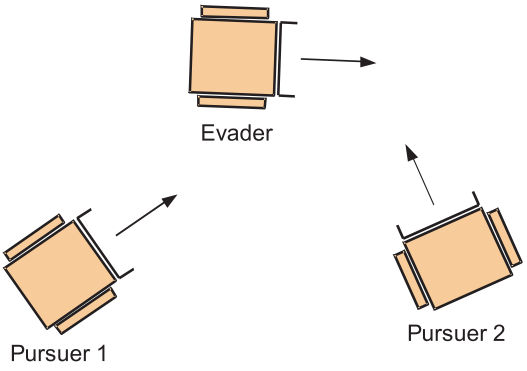

Fig. 1. Sample pursuit-evasion scenario

examining it by pursuers' visibility region, such that the possible evader is caught in a finite time.

The simplest case of visibility region is a thin "laser". When a pursuer moves the laser with the given speed, it clears the area. A pursuer with k laser lines is called k-searcher. When we think of visibility regions as an angular area denoted by its size δ, we talk about a δ-searcher.

The evader is caught when it appears in the pursuer's visibility region or the whole area is cleared.

To solve the problem in this type of representation, we need to transform it to the graph representation. The graph is constructed by calculating all possible states in the polygonal area. The graph shape depends on the type of pursuer's visibility region.

- **Probabilistic representation** [8, 2] The environment is modelled using grid world. One grid cell can contain the evader, pursuer or obstacle.

  According to the obtained sensory information we can assign the probability of evader presence to the grid cells.

  The pursuers are moving on the grid and trying to approach the evader. The evader is caught, when pursuer approaches to the critical distance.

  This approach is the closest to the common methods that are in use for navigation of multiple robots.

2.3 Situation Scenario

Our scenario includes a group of mobile robots guarding a area to be secured (*eg* warehouse or industrial complex) with a map of the environment available. The task for the robots is to patrol in the area and when an intruder appears, to contact it in the shortest possible time. The situation is depicted in Fig. 1.

The pursuit-evasion problem in the real world environment puts various demands on the deployed control system. Some situations, especially those unexpected, need to be handled by each robot by itself, obstacle avoidance. Another kind of situation is the one that needs the whole team cooperation when an intruder appears, some robots need to handle the situation, and some others need to take care of the rest of the area.

3 PROBLEM SOLUTION

To provide a working solution for the pursuit-evasion problem, we need to specify the internal representation of the situation and an algorithm capable of working with this representation.

Available information, such as obstacle or evader position then need to be expressed in this representation and processed. The result of the algorithm is the demanded speed vector for every robot in the group.

3.1 Situation Representation

In the area of pursuit-evasion, we need to find a way to express positions of various objects.

We believe that fuzzy logic is the right tool to handle situations that can emerge in the pursuit-evasion problem. Its main advantage, when compared with existing approaches, is ability to simulate the human way of thinking, generalization and handling vague and uncertain terms [10].

By using fuzzy sets we can express vague knowledge from the problem domain in the form of rules and process it using a rule-based system proposed by Mamdani and Asilian in [4].

We assume the existence of a planar world with several obstacles. The whole world is bounded by a rectangle with sides $[X, Y]$. The area in the map of the world can be viewed as a fuzzy relation $m(x, y)$, $x \in X$, $y \in Y$, where the degree of membership denotes the membership of the point p with coordinates $[x, y]$ to the area m. The map of the world is then defined as a set of named relations M, where each relation $m(x, y)$ marks some specific area.

This kind of fuzzy relation can be used to assign some meaning to certain parts of the map. Thanks to this framework, we can plausibly describe terms like 'area close to the enter' or 'some free space in the centre' in a way that is natural for human working with the system and also for the machine that executes inserted commands.

There are many ways of constructing this kind of fuzzy relation both autonomous or human-made. In this paper we want to provide the simplest one using linguistic variables [10]. The area in the map can be constructed by creating a linguistic variable for each axis and partitioning it to parts using triangle-shaped fuzzy sets.

An example for partitioning of x axis of the map is depicted in Fig. 2. We divide the area in five parts and each one has assigned a label and one fuzzy set. We continue by creating fuzzy partition for y axis. We can label the sets on the y axis by numbers.

When the x and y axes are partitioned by triangle-shaped fuzzy sets, we can obtain fuzzy relations expressing pyramid-shaped areas in the map, where each point

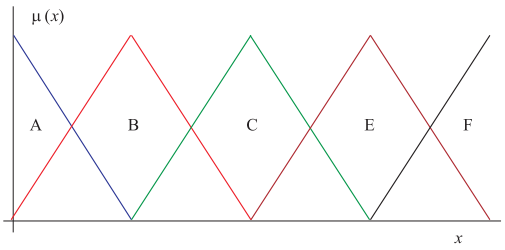

Fig. 2. X axis partition

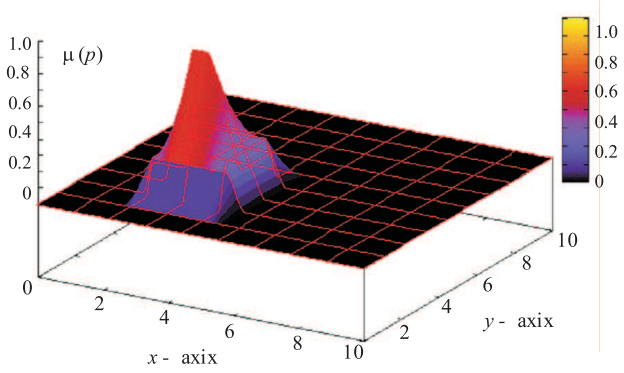

Fig. 3. Area as a fuzzy relation

in the plane has a degree of membership to this relation. This can be expressed as a membership function

$$\mu_A(p) = min(\mu_{Ax}(p_x), \mu_{Ay}(p_y))\,. \tag{1}$$

Using this expression we obtain a fuzzy relation (Fig. 3)

$$Area_A(p) = \{(p, \mu_A(p))\,, p \in X \times Y\} \tag{2}$$

where each point p in the plane with sides X, Y denotes a degree of membership to the $Area_A$.

Constructing the fuzzy relations by expert is not the only way to obtain such information. It would be very interesting to propose a method for autonomous map construction using robot's sensors, or to have a method for transformation of the available maps represented in a different way.

The constructed map areas then can be used in the rules of the knowledge base of the proposed system. When the map areas are allocated, the goal is to control the whole team of pursuers in a way that minimizes the time to catch the evader.

The evader is caught, when the closest pursuer and evader reach a critical distance, such that

$$|pa_{closest}(p) - ea(p)| \leq d \tag{3}$$

where $pa_{closest}$ is the area of the closest pursuer, ea is the position of the evader and d is the critical distance.

Positions of the pursuer and evader are expressed as a fuzzy relation. This raises a question what the expression $|pa_{closest}(p) - ea(p)| \leq d$ means. One of the possible solutions is to use a defuzzification operator and perform calculation of the distance in the crisp domain. By this transformation and using Euclidean metrics we get that the evader is caught if

$$\min(\sqrt{((P_{ix} - E_x)^2 + (P_{iy} - E_y)^2)}) \leq d \tag{4}$$

where

$$P_i = Defuzz(pa_i(p))\,, \tag{5}$$

$$E = Defuzz(ea(p)) \tag{6}$$

are positions of the evader and pursuers expressed as crisp points in the map.

It is good to choose a defuzzification operator with low computational demands. In the case of simple pyramid-shaped fuzzy relations the result could be just the maximum, such that

$$Defuzz(m(p)) = arg\max(\mu_m(p))\,. \tag{7}$$

Result of this defuzzification operator is point P with coordinates $[P_x, P_y]$ of the centre of the fuzzy relation $m(p)$. In the case of more complex shapes of the area we need to find another operator to find a good crisp point describing the relation.

3.2 Multi-Agent Control Algorithm

We propose a hierarchical multi-agent control system based on fuzzy rules. Our reasons to choose this kind of systems are

- fuzzy logic enables usage of expert knowledge in the task domain and its mechanism can easily deal with uncertain inputs
- hierarchical structure can involve knowledge both on a single robot level and on the whole team level
- parameters of the fuzzy systems can be improved through numerical optimization, *eg* using the approach described in [6].

Inspiration for our approach comes from a soccer match. Soccer is played by the players that are spread around the playground. Every player has his own position and can play different role. *The Coach* assigns positions for the players and determines the team strategy. The original idea for this approach was used for a control of the robotic soccer in [3].

Basic parts of our system are in Fig. 4 *The player* agent that handles the knowledge and reasoning on the local level (related to one robot) and the *Coach* processes the knowledge on the team strategy level. The *Player* has capability do decide if to obey or not to obey *Coach's* action recommendation (*eg* to move to the proposed location), according to the current situation.

Inputs of the whole system are the position of the evader and positions of the pursuers. The most important input to the system is the position of the evader and can be obtained from the camera system. It is taken into account by both *Coach* and *Player* agents. Recognized actual positions of the pursuing robots can be used as a feedback for their navigation. The output of the system is a set of demanded wheel speeds of the mobile robots.

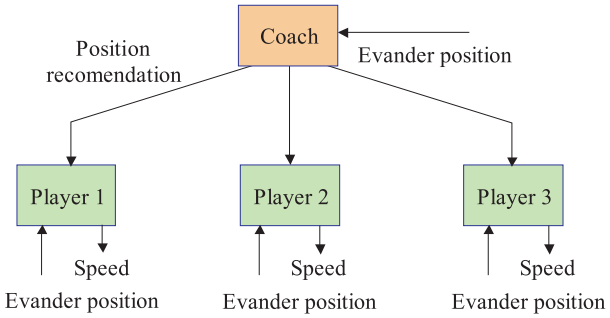

Fig. 4. Structure of the proposed system.

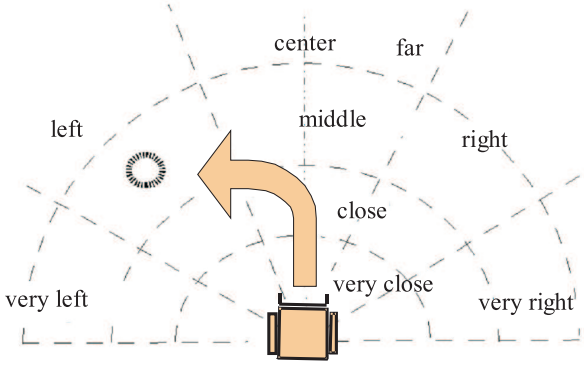

Fig. 5. Local area partition for agent *Player*

3.3 Agents and Their Knowledge

The knowledge base for our system is divided into two parts. *Coach's* knowledge base contains the rules about approximate displacement of the members of the team. On the other hand, the knowledge base of the *Player* is focused on individual robot and has rules about robot's movement in the space.

3.3.1 Agent coach

The input of *Coach's* inference systems is the position of the evader. This position is fuzzified and inserted to the inference system. The result of its inference is a set of recommended positions for the whole team.

To create the knowledge base of the *Coach*, we need to express areas in the map of the environment. Then we will specify interesting input data *eg* position of the evading object.

Rule for the *Coach's* inference system will then look like

```
IF evader IS in area A2 THEN
robot1 IS in area A2, AND robot2 IS in area A1
AND robot3 is in area B1
```

The *Coach* inference process can be described as sequence

1. Input of the position of the evader
2. Fuzzification
3. Internal inference process
4. Sending the output data to the team members

3.3.2 Agent Player

The input of agent *Player* is the position of the evader again. Also, we can take care of sensory data, like information about obstacles from a sonar. The *Player* takes these assumptions into account and resolves it into speed of its wheels or other actuators.

`Player's` knowledge base is constructed in a similar manner. The major difference is that the natural way for coding position information is using radial coordinates, *ie* angle and distance coordinates. We can partition the area around the robot in this way. We also need to label and partition the speed range of the robot, *eg* slow, medium, fast. An example of area partitioning is drafted in Fig. 5.

The *Player* receives this data Position recommendation from the *Coach*, position of the evading robot from an image recognition system and position of the obstacles from other sensors (laser or sonar range sensors).

Rules for its inference system look like this one

```
IF evader IS far left
AND recommended position is middle centre
THEN speed IS slow forward
```

We can see, that agent *Player* has information about the position of the evader and also information about recommended position. His rule base can be divided in two parts — one that takes care about matters of the whole team – catching the prey, and the second part can care about his own goals – reckoning or obstacle avoidance.

Based on its knowledge base, the *Player* can choose whether to obey or not to obey *Coach's* position recommendation. When the information about the evader is not available, the *Player* can move to the recommended position. When the evader suddenly appears, he can decide to follow him.

The inference of the *Player* agent can be described as

1. Input of the data evader position, obstacle position
2. Fuzzification
3. Internal inference process
   (a) Decide whether to follow own or team goal
   (b) Perform goal (pursue evader or recommended position)
3. Wheel speed computation

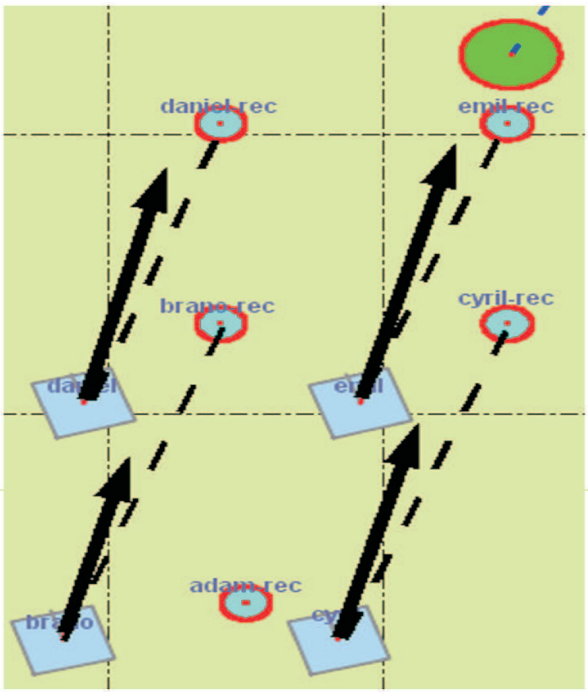

Fig. 6. Accuracy of the *Player's* inference

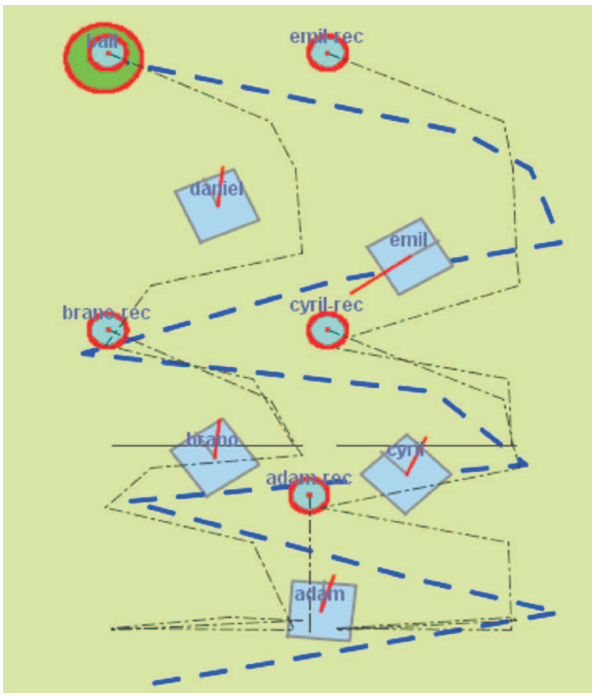

Fig. 7. Team coordination

4 EXPERIMENTS

4.1 Methodology

In the first stage of experiments we want to prove usability of this approach for a control of a group of mobile robots. For this, we have used a simplified simulation environment. We have set up some simple situations that can occur and observed the system behaviour.

4.2 Implementation

The control software is implemented using a combination of well-known tools for multi-agent systems, expert systems and fuzzy logic.

As a base we have chosen the JADE multi-agent platform. JADE is a reference implementation of de-facto standard of inter-agent communication language called FIPA (Foundations of Intelligent Physical Agents). Each basic component of the control system corresponds to one JADE agent.

Knowledge handling and reasoning is handled by JESS expert system shell with Fuzzy-J extension. JESS is similar to the well-known expert system shell CLIPS and uses the same syntax. The rule for agent *Coach* from previous example

```
IF evader IS in area A2 THEN
robot1 IS in area A2,
AND robot2 IS in area A1
AND robot3 is in area B1
```

can be written using equation (1) as two rules of the fuzzy rule based system

```
IF evader_position_x IS A2_x THEN
robot1 IS A2_x, AND
robot2 IS A1_x AND
robot3 IS B1_x

IF evader_position_y IS A2_y
THEN robot1 IS A2_y,
AND robot2 IS A1_y
AND robot3 IS B1_y
```

and in the CLIPS/JESS syntax as

```
(defrule evader-A2-x
(width-position evader ?wp&(fuzzy-match ?wp
    "A2-x"))
=>
(go-x robot1 "A2-x")
(go-x robot2 "A1-x")
(go-x robot3 "B1-x")
)

(defrule evader-A2-y
(width-position evader ?wp&(fuzzy-match ?wp
    "A2-y"))
=>
(go-y robot1 "A2-y")
(go-y robot2 "A1-y")
(go-y robot3 "B1-y")
)
```

6 RESULTS

We have displayed the results of simulation on the screen. Rectangles in the screenshots mark positions of the pursuing robots and their orientation. The big circle means the position of the evader. Small circles represent recommended positions for the members of the team.

In the performed experiments we have observed the following areas

1. Accuracy of the resolved *Player's* speed vector
2. *Coach's* ability of team coordination

In the first experiment we have set up a pursuit-evasion problem and observed the behavior of pursuers. As we were interested in the *Player's* speed vector, we have graphically denoted it with an arrow. Straight direction is marked with a dash line. An example situation is shown in Fig. 6.

We can see, that the *Player's* speed vector has some inaccuracy. However, agents were able to find a way to the evader anyways in all cases. We have deduced that accuracy of the speed vector is sufficient.

In the second experiment the evader was running along a sine-shaped line. We have recorded trajectories of the whole team and also trajectories of the recommended positions. Results are shown in Fig. 7.

Results of the performed experiments show that this approach is usable in the control of multiple robots. Major enhancements are good group cooperation, conflict resolution and intelligent behaviour of robots in the pursuit-evasion problem.

5 CONCLUSIONS

Our system allows execution of difficult tasks for teams of mobile robots. Its main advantage is simple knowledge expressing from the problem domain in the form of rules in a way that is natural to human reasoning.

We hope that our approach would be useful not only in the field of pursuit-evasion problems, but also for research of multi-agent technologies, expert systems and fuzzy logic. This methodology could be used as a starting point in the development of arbitrary multi-agent systems for group coordination, decision support or cooperation of multiple entities on a common task.

References

[1] GERKEY, B.—THRUN, S.—GORDON, G.: Visibility-Based Pursuit-Evasion with Limited Field of View, The International Journal of Robotics Research **25** No. 4 (2006), 299.

[2] HESPANHA, J.—KIM, H.—SASTRY, S.: Multiple-Agent Probabilistic Pursuit-Evasion Games, Decision and Control, 1999. Proceedings of the 38th IEEE Conference on 3 (1999).

[3] HLÁDEK, D.: Multi-Agent Fuzzy Control of the Robotic Soccer, SAMI 2007 Proceedings, 2007, pp. 329–341.

[4] MAMDANI, E. H.—ASSILIAN, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Human-Computer Studies **51** No. 2 (Aug 1999), 135–147.

[5] PARSONS, T.: Pursuit-Evasion in a Graph, Theor. Appl. Graphs, Proc. Kalamazoo, 1976.

[6] PREITL, S.—PRECUP, R.—FODOR, J.—BEDE, B.: Iterative Feedback Tuning in Fuzzy Control Systems. Theory and Applications, Acta Polytechnica Hungarica **3** (2006), 81–96.

[7] SCHENATO, L.—OH, S.—SASTRY, S.—BOSE, P.: Swarm Coordination for Pursuit Evasion Games using Sensor Networks, Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on (2005), pp. 2493–2498.

[8] VIDAL, R.—SHAKERNIA, O.—KIM, H.—SHIM, D.—SASTRY, S.: Probabilistic Pursuit-Evasion Games Theory, Implementation, and Experimental Evaluation, Robotics and Automation, IEEE Transactions on **18** No. 5 (2002), 662–669.

[9] YAMASHITA, M.—UMEMOTO, H.—SUZUKI, I.—KAMEDA, T.: Searching for Mobile Intruders in a Polygonal Region by a Group of Mobile Searchers (Extended Abstract), In Symposium on Computational Geometry, 1997, pp. 448–450.

[10] ZADEH, L.: The Concept of a Linguistic Variable and its Application to Approximate Reasoning-II, Information Sciences **8** No. 4 (1975), 301–357.

Received 3 September 2008

Daniel Hládek is a PhD student at the Centre for Intelligent Technologies at Technical University of Košice, Slovakia. He graduated in the field of artifficial intelligence and his research is focused on the fuzzy systems and mobile robotics. He is author of several conference papers and posters in this area.

Ján Vaščák graduated in technical cybernetics from the Faculty of Electrical Engineering and Informatics of Technical University in Košice in 1990, gained his PhD in informatics and automation from the same university in 1996. At present he is an Assistant Professor at the department of Cybernetics and Artificial Intelligence of his home university. He is an IEEE member as well as a member of the Slovak Association for Artificial Intelligence. His research is in the field of artificial intelligence (AI), especially fuzzy systems, neuro-fuzzy systems and utilization of AI means in mobile robotics.

Peter Sinčák is a University Professor in the branch of AI at Technical University of Košice, Slovakia. He is a head of Center for Intelligent Technologies and Program director for the branch of Artificial Intelligence at Technical University of Košice, Slovakia.