Agent Control for Reconfigurable Open Kinematic Chain Manipulators

Janez Sluga1, Viktor Zaletelj2 and Andrej Žemva3,*

1 Trimo d.d., Trebnje, Slovenia
2 EVORO d.o.o., Novo mesto, Slovenia
3 University of Ljubljana, Faculty of Electrical Engineering, Ljubljana, Slovenia
* Corresponding author E-mail: andrej.zemva@fe.uni-lj.si

Received 07 Jan 2013; Accepted 09 Aug 2013

© 2013 Sluga et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract This paper presents a method for the autonomous control of differently structured open kinematic chains based on multi-agent system technology. The appropriate level of distributing local autonomy (agents) to a manipulative structure is defined, which makes it possible to dynamically change the number, type and structure of manipulative components without modifying their behavioural logic. To achieve fast reconfigurable and scalable manipulative systems, a new multi-agent method is developed for controlling the manipulator kinematics. The new method enables independent manipulator structure from the control system because of its structural and system modularity. The proposed method consists of kinematic equations for use in an agent environment, agent motion-planning algorithms, evaluation functions, agent control logic and kinematic algorithms. The results of simulations and real-world experiments demonstrate the usefulness of the approach for different non-redundant and redundant manipulation structures.

Keywords Reconfiguration, Multi-Agent System (MAS), Autonomy, Distributed Environment, Manipulators

1. Introduction

Most manipulation control systems are designed to perform predefined functions and sets of tasks. These systems have difficulty operating in an unstable and dynamic environment in which different unpredicted situations can occur. It is thus challenging to ensure rapid and effective system adaptation to changes in the environment, changes in the structure of the manipulator and changes in the requirements. Manipulation systems designed using centralized control methods are not capable of meeting these requirements without being properly adjusted at the logical, informational and structural level, which requires many resources. Thus, alternative methods should be provided. These alternatives include parallel algorithms, distributed control, autonomy distribution and others, for which there are no explicit and universally useful solutions available. Additionally, according to the authors in [1,2], there is a lack of tools and methodologies for planning, modelling and controlling highly flexible manipulation systems.

Several authors have proposed solutions to the issues of reconfigurable manipulators. For Pan et al. [3], a viable solution was a distributed control environment for which
they present the software architecture design and system implementation of a reconfigurable system for real-time multiple robot control. The system is partitioned into autonomous functional units and data modules manipulated by the units. A decentralized control algorithm for modular and reconfigurable robots, with each joint acting as an independent subsystem, is proposed by [4]. Good position tracking with no velocity control is achieved and tested with a linear PD controller. There are several problems with dynamic conditions, nonlinear models, payloads and modifying robot configuration. Similar issues are reported by [5]; they present a modular distributed control technique (based on torque sensing) for modular and reconfigurable robots. A method for designing reconfigurable assembly systems is described by [6]. They discuss constraint logic programming for dealing with a combinatorial explosion. This gives them the possibility of dynamic adaptation and emergent generation of optimal configurations for different requirements and constraints.

In [7] the authors propose a distributed control system based on the general suppression framework to control a set of MSR robots configured into a planar manipulator arm, consisting of autonomous modules that evaluate and react to the environment independently. The modules communicate with their immediate neighbours and exhibit either aggressive or tolerant behaviour. Similarly, in [8] the authors describe a control architecture for autonomous multi-robot systems. They address task description, distribution, allocation and coordination of the system components. They implement local intelligence but provide no solution for quick component substitution and system adaptation to new circumstances in the environment.

One of the possible distributed control approaches, which introduces system modularity and thus fast structural reconfigurability and scalability, is the multi-agent based method. It has many properties that can be used in the reconfigurable and highly adaptive manipulators field. According to [9,10], the agent is an autonomous entity that acts independently, communicates with other units and negotiates with them to execute the demanded tasks. The agent also coordinates its activities with other units to achieve an individual or systematic goal. Using agent-based decision-making and interactions while performing tasks allows us to build systems that can dynamically respond and adapt to unpredictable events/changes in the environment. The ability of the agent to adapt its behaviour during operation reduces the need to predict all the possible scenarios and changes that the designed system might encounter. Different types of agents have been developed, which vary in terms of their architecture, behaviour, performance and usability.

In [11] the author presents a failure-tolerant architecture in which the loss of one or several agents may be supported by the whole system. He describes the humanoid robot HRP2 built of actuators associated with agents acting independently from each other. Some problems with speed control of the end-effector are discussed. Another example of using agent-based architecture is given in [12]. They describe the software architecture in the agent framework for mobile manipulators. They established cooperation between the arm and the mobile base to accomplish tasks for mobile manipulators. There are no possibilities mentioned for manipulator structural reconfiguration.

The focus of our previous work presented in [13] was on developing a special modelling environment and architecture based on multi-agent systems (MASs). This makes it possible to assign functionalities among autonomous system components, called agents. The manipulation system built with the proposed approach gives each of the active components (joints) a certain degree of autonomy and enables real-time reconfiguration of the manipulator structure. It also offers full control of the system redundancy, requiring no modification in the component logic behaviour. The authors describe the development of reactive agents to control different manipulator structures with no speed control (differential kinematics) of end-effectors implemented. Additionally, no methods are suggested to define the component behaviour and its properties.

In this paper, a new method for autonomous control of differently structured open kinematic chains is presented. The method is based on the multi-agent system technology, in which agents act autonomously with some data interchange. In the following sections, a multi-agent architectural design for differently structured manipulators is described. Different types of agents are defined, which are assigned to basic components of the physical manipulation structure. Section 3 introduces a new multi-agent method to control the manipulator kinematics, which allows dynamic changes and scalability of the physical manipulative structure without any modification of the control system architecture. The method consists of direct and differential kinematics and algorithms for movement planning and evaluation, where each agent autonomously generates its motion plans, evaluates them and then chooses the optimal one according to its goals for agent motion execution. In Section 4, the proposed approach was simulated and evaluated using different non-redundant and redundant structural configurations of the manipulator, where the agent control system remains the same. Solutions for point-to-point movement with the demanded end-effector velocity are presented. To prove the proposed control system flexibility some real-world experiments...
were performed on a manipulator composed of translational joints. The presented work is summarized in the final section, where the performance of the current multi-agent control system is discussed and possible improvements are suggested.

2. Design of the MAS environment

This section outlines the design of the control architecture of manipulators by using the multi-agent approach. To achieve system modularity and thus fast reconfigurability and scalability of the manipulation system, the appropriate level of its dividing and partitioning must be defined on both the control and physical level.

In our approach, the physical manipulation structure is divided into fixed and Dynamic Structural Components (DSCs). The dynamic components can be the rotational and translational joints that might change the pose and kinematics of the manipulator end-effector. They are further divided into active (actuators) and passive (free joints) components. Any other component that does not introduce changes to the manipulation structure or working environment is considered a fixed structural component.

Figure 1 gives an architectural overview of the multi-agent control system of a simple RTR manipulator. The control system is formed of several autonomous control units (agents) of different types. All dynamic structural components have assigned autonomous DSC agents. The agent has control over one active DSC or represents one passive DSC. The Work Piece is represented by the WP agent and the Tool is represented by the T agent. The interactions needed between the WP and the T agents introduce new goals and constraints to the manipulation system for certain applications. The role of the DSC, WP and T agent types is to connect and represent the physical world in the virtual multi-agent environment. The role of the moderator agent in an MAS is to represent one of the application goals and share it with other agents. Its role is also to notify the other agents of the overall manipulative structure and to coordinate (if needed) the agent cooperation to achieve the manipulator or final application goals. The authors of [13] describe the informational architecture of the above agents in detail.

The proposed design, in which one DSC is always represented by its own DSC agent, allows dynamic changes and scalability of the physical manipulative structure without any modification of the control system architecture. In this way, the control system does not depend on the manipulator structure and can support any structural configuration.

3. Multi-agent kinematic control

The main goal of the manipulation system is to correctly execute a desired motion path that comes from T and WP interaction. For the manipulation structures known when designing a control system, the joint values for any desired movement can be computed with classical methods of inverse kinematics. In case of any structural or kinematic changes, the inverse kinematics analytical problem should be solved again. This process is mathematically demanding and does not necessarily provide an explicit solution. Moreover, its complexity increases with the structural diversity and the number of elements (multi m-DOF, redundancy).

To avoid problems with computing inverse kinematics, an approach such as closed-loop inverse kinematics with a Jacobian transpose, which is described in [14], could be considered. This method requires knowledge of the direct kinematic equations and of the Jacobian of the manipulator. As a result, joint values are obtained, with no end-effector velocity control. This is suitable for predesigned centralized control systems in which low requirements for system scalability, reconfigurability and adaptability exist. Other, more flexible methods should be considered.

In the following sections, an alternative approach using the multi-agent method to control the kinematics of the end-effector for any manipulation structure is presented. It is composed of DSC agent control logic and an agent kinematic algorithm with a new solution for computing kinematics, movement planning and evaluation within a multi-agent environment.

3.1 DSC agent control logic

In general, the structure of a manipulator consists of a sequence of fixed structural components (links) interconnected by the dynamic components (joints) shown in Figure 2. It involves everything from the base to the end-effector (tool) and includes different combinations and types of joints and links. To achieve movement of end-effectors in space, according to the architectural design (Section 2), each joint is controlled by a DSC agent. All DSC agents use identical agent control logic, which determines the procedure of the agent motion planning, simulation, evaluation and execution.
Figure 3 shows the block scheme of the agent control logic, which is executed in parallel by different DSC agents. After the DSC agent receives a movement execution task for the current manipulation structure, it autonomously computes its own motion plans $MP = (mp_{Ai})$, which are described in the following section. To prepare motion plans, DSC agents need to exchange their optimal motion plans $(mp_{A0}...mp_{Ai})$ with each other through a common communication interface. The MP of the current agent are evaluated in comparison to the end-effector position, orientation and velocity errors (Err). If all the errors of the best $mp_{Ai}$ are within the tolerances, then this $mp_{Ai}$ becomes the solution for the current task. The DSC agent takes a new task and repeats the entire procedure. Because all the DSC agents share their optimal $mp$, the overall solution for a certain task is achieved almost at the same time. As soon as this happens, the task is prepared for execution.

3.2 Agent kinematic algorithm

In the currently discussed multi-agent manipulation control scheme, each agent makes decisions about its movements in the environment on its own without negotiation with other participants. However, there can be some feedback loops between them, such as sensing through the environment and sharing their internal states. Each agent autonomously prepares its possible future activities, evaluates them and then chooses the optimal one according to its goals. In this section, the agent kinematic algorithm is defined.

The algorithm is composed of five main operations (Figure 4): computing direct and differential kinematics, determining position and velocity errors, calculating the evaluation function, choosing an optimal motion plan and generating new motion plans (MP). Each DSC agent computes the direct and differential kinematics of the manipulation system on its own. Its computation depends on the current manipulation structure, kinematic conditions and current internal and environment states. To compute the manipulator kinematics, the agent formulates equations in accordance with [14], which describes the kinematic equations. The equations are
needed to evaluate the generated motion plans. The computation is described in the following section.

3.2.1 Agent activity planning

The motion plans (MP) for each DSC agent are composed of its possible joint values. To derive the possible agent joint values (position or rotation) for the next time step, the velocity is needed:

\[
\dot{q}(t + 1) = \dot{q}(t + 1) \cdot t + \dot{q}(t)
\]

\[
\dot{q}(t + 1) = \dot{q}\text{max} \cdot \mathbf{v}_b
\]

\[
\mathbf{v}_b = \mathbf{W}_p \cdot \mathbf{v}_{p_b} + \mathbf{W}_o \cdot \mathbf{v}_{w_b}; \sum W = 1
\]

where \(\dot{q}\text{max}\) is the maximum joint velocity, \(\mathbf{v}_b\) is the relative velocity and \(\mathbf{W}_p\) and \(\mathbf{W}_o\) are the prioritizing position and orientation task weights. The relative linear \((\mathbf{v}_{p_b})\) and angular \((\mathbf{v}_{w_b})\) velocities are:

\[
\mathbf{v}_{p_b} = \min \left[ \frac{P_{\text{ERR}}}{K_p \cdot p_{\text{tol}}}; 1 \right]
\]

\[
\mathbf{v}_{w_b} = \min \left[ \frac{\varphi_{\text{ERR}}}{K_p \cdot \varphi_{\text{tol}}}; 1 \right]
\]

where \(K_p\) is the gradient of the approaching speed range to its target and \(p_{\text{tol}}\) and \(\varphi_{\text{tol}}\) are the position and orientation tolerance, respectively. \(P_{\text{ERR}}\) and \(\varphi_{\text{ERR}}\) are the absolute values of the end-effector linear and angular velocity, computed according to the direct kinematic equations in [14]. In Equations 4-5, the relative velocity depends only on the end-effector linear and angular velocity error. Thus, it can be used only for velocity control and not in combination with the pose. To plan the agent motion, we propose a new method. To determine a set of the possible joint values and velocities (Equations 1, 6), a set of possible gradients \(K_p\) and \(K_q\) must be computed:

\[
K = \{K_i; i = 0, 1, 2\}
\]

At the initial time step \((t = 0)\), they are:

\[
K_0(t = 0) = K_{\text{min}}
\]

\[
K_1(t = 0) = \frac{K_{\text{max}} - K_{\text{min}}}{2}
\]

\[
K_2(t = 0) = K_{\text{max}}
\]

\[
K_{\text{min}}; K_{\text{max}} > 0
\]

For any later time step \((t > 0)\), they are calculated as:

\[
K_0(t + 1) = \max \left( K_{i-1}(t) + \frac{K_i(t) - K_{i-1}(t)}{2}; K_{\text{min}} \right)
\]

\[
K_1(t + 1) = K_i(t)
\]

\[
K_2(t + 1) = \min \left( K_{i+1}(t) - \frac{K_{i+1}(t) - K_i(t)}{2}; K_{\text{max}} \right)
\]

with rules:

- If \(K_i(t) = K_{\text{min}}\) then \(K_{i-1}(t) = K_{\text{min}}\)
- If \(K_i(t) = K_{\text{max}}\) then \(K_{i+1}(t) = K_{\text{max}}\)

Index \(i\) can equal zero, one or two, depending on the agent evaluation of the optimal motion plan made for current time step \(t\). Equations 12-18 are designed based on a bisection method and are used to obtain a rough approximation of \(K_0\) and \(K_2\) gradients. The purpose of choosing this method is to give the control system the ability to dynamically change gradients of approaching
speed and acceleration range to its target and thus lower overlapping. The agent motion plans for time \( t+1 \) have a set of possible joint values and velocities with their gradients:

\[
q(t + 1) = \{q_i; i = 0,1,2\}
\]

(19)

\[
K_p(t + 1) = \{K_{ii}; i = 0,1,2\}
\]

(20)

\[
\dot{q}(t + 1) = \{\dot{q}_i; j = 0,1,2\}
\]

(21)

\[
\dot{K}_p(t + 1) = \{\dot{K}_{ii}; j = 0,1,2\}
\]

(22)

The combination of the joint value and the velocity defines the motion plan:

\[
mp_{ij} = \{q_i, \dot{q}_j\}; i, j = 0 \ldots 2
\]

(23)

The agent motion simulation for computing direct and differential kinematics is made based on a set of the motion plans (Figure 4):

\[
MP(t + 1) = \{mp_{0,0}, mp_{0,1}, mp_{0,2}, ..., mp_{i,j}, ..., mp_{p,2}\}
\]

(24)

3.2.2 Evaluation function

To execute motions according to its global and local goals, each agent evaluates motion plans and chooses the optimal one. For this purpose, the evaluation function is defined and it consists of goal functions (G), and their weights (A):

\[
C = \sum_g A_g \cdot G_g
\]

(25)

\[
\sum_g A_g = 1
\]

(26)

where \( g \) is the number of the goals to be fulfilled and the corresponding weights define priorities between the goals. The goals (end-effector pose, velocity, system behaviour energy, high manipulability of the component, etc.) for the entire manipulation system and for each agent can differ. The goal function indicates the extent to which a certain goal has been fulfilled. The goal functions for the end-effector position, orientation, linear and angular velocity errors are defined as:

\[
G_p = \frac{p_{err}}{p_{tol}}
\]

(27)

\[
G_o = \frac{\Theta_{err}}{\Theta_{tol}}
\]

(28)

\[
G_p = \frac{\dot{p}_{err}}{\dot{p}_{tol}}
\]

(29)

\[
G_o = \frac{\dot{\Theta}_{err}}{\dot{\Theta}_{tol}}
\]

(30)

Combining the goal functions given above with Equation 25 yields the evaluation function for assessing the end-effector pose and velocity with respect to the target:

\[
C = A_p \cdot G_p + A_o \cdot G_o + A_p \cdot G_p + A_o \cdot G_o
\]

(31)

where \( A_p \), \( A_o \), \( A_p \), and \( A_o \) are the corresponding priority weights.

The simulated motion plan with the minimal value of the evaluation function is chosen for agent motion execution (Figure 4):

\[
\min\{C(MP(t + 1))\} \rightarrow mp_{i,j}(t + 1)
\]

(32)

The corresponding i, j motion plan indexes determine the gradients (Equations 16-18) used in computing the motion plans at the next time step.

In the following section, experimental results obtained with the proposed methodology are shown.

4. Approach evaluation

This section presents the results of experiments conducted on different structures of non-redundant and redundant manipulators.

4.1 Non-redundant manipulation structures

Figure 5 shows experimental non-redundant 3-DOF manipulation structures. The structures are composed of translational and/or rotational joints. All joints of the same type are identical in their properties and in their agent control logic. To compare their usability, they have to perform the same task on the presented manipulation structures. The task for this experiment is described as follows:

\[
T = \{(P_i, [v_{F_i}]), i = 1 ... n\}
\]

(33)

where \( P_i [\mu m] \) and \( v_{F_i} [\mu m/s] \) are the target position and the linear velocity vectors with respect to the x, y and z-axes. The first experiment to evaluate is the point-to-point movement of the end-effector with the following position and linear velocity requirements (Equation 33):
According to the proposed approach, motion plans for the position and linear velocity are prepared separately. Figure 6 shows the end-effector position and linear velocity in the process of agent kinematic algorithms searching for joint solutions for the described tasks. All DSC agents start searching from the initial point and with a linear velocity equal to zero.

The global end-effector xyz coordinates depend on the combination of computed agent motion plans and their joint values. From Figure 6, it can be seen that some target coordinates are reached before others and that it takes some time to acquire the right joint combination. In this process, the agent system control tries to preserve the already reached target goals by slightly modifying the joint values to compensate for the new errors. When all the errors are within the tolerances, the agent’s joint values become the solution of the current task point. The solution is achieved at time $T_{STx(Pi)}$ (highlighted by vertical lines) and used for later execution.

The ST1 manipulation structure in Figure 5 represents the orthogonal 3D coordinate system with axes aligned to the global coordinate system. Every component can make contributions to the end-effector only in its own direction without interfering with the other components. It can be seen (Figures 6ab - ST1) that the curves of the end-effector position and the linear velocity are almost identical in all directions and in the corresponding position gradients and the overlapping size.

Multi-agent searching of the correct end-effector position is independent from searching the linear velocity and it

$$T_1 = \begin{pmatrix} 1000 \\ 1000 \\ 1000 \end{pmatrix} \times 10^8, \begin{pmatrix} 1000 \\ -1000 \\ -750 \end{pmatrix}$$

Figure 6. End-effector position (a) and linear velocity (b) in the process of agent kinematic algorithms searching for joint solutions for ST1-ST4
has its own searching speed. According to the kinematic equations in [14], the current end-effector velocity depends on the current position. Because of this, we can see some deviations in Figure 6b in the searching solution for the linear velocity before the position stabilization shown in Figure 6a. This is evident in the more demanding manipulative structures ST3 and ST4. For manipulation structures ST2 and ST3 (Figure 5), component movement might influence the end-effector motion in multiple directions. Figure 6 shows that the same agent control system is capable of performing this compensation. Some deviations around the target point can be seen due to the non-cooperative DSC agents. Nevertheless, the proposed approach still converges to a solution. The time steps needed to acquire the solution (Figure 6 - TSTx(Pi)) grows with the structural complexity of the manipulator (Table 1).

|   | P1  | P2  |
|---|-----|-----|
| ST1| 41  | 61  |
| ST2| 47  | 68  |
| ST3| 58  | 90  |
| ST4| 59  | 81  |

Table 1. No. of time steps for the agent kinematic algorithm to acquire a solution for ST1-ST4

Figure 7 shows the end-effector movement execution of the acquired solutions for the corresponding structures. Because of the unsynchronized DSC agents exchanging their current states through the info bus, the smoothness of the calculated end-effector path depends on the control loop time. There may be some deviations around the start and end target points due to communication delays; however, the final goal is nevertheless achieved. For the ST1, ST2 and ST3 structures, the mapping between the joints and end-effector is a linear function. The ST4 structure has a rotational joint. Thus, the nonlinear mapping of the ST4 structure results in an incomplete solution (velocities) for T1, as seen in Figures 6b and 7. However, the solution for the next point (T2) is fully acquired. This can be solved by the agent system by defining the path between the points more precisely (adding subpoints), which is presented in Figure 8 for ST5.

The task is:

\[ T_2 = \left\{ \begin{bmatrix} 10 \\ 10 \\ 7 \end{bmatrix} \times 10^4, \begin{bmatrix} -1000 \\ 1000 \\ -750 \end{bmatrix} \right\} \]

Figure 8. End-effector position (a) and linear velocity (b) in a process of agent kinematic algorithms searching joint solutions for experimental non-redundant manipulation structure ST5 (c)
with initial end-effector position \( [20 \ 0 \ 0] \times 10^4 \mu m \) and linear velocity 0. Experiments vary in terms of the number of subpoints (3, 6, 12) added to the original motion task. Subpoints helped the agent kinematic algorithms converge to target points (Figure 8). The time steps taken to find solutions \( (T_{SP}(P_i)) \) increase with the number of added subpoints (Table 2). The increased number of subpoints makes the end-effector movement straighter, which can be seen in the ST5 end-effector movement execution in Figure 9.

![Figure 9. Movement execution of the acquired solution for ST5 with added subpoints](image)

| Subpoints  | P1 | P2 |
|------------|----|----|
| Subpoints 3| 95 | 116|
| Subpoints 6| 147| 188|
| Subpoints 12| 227| 286|

Table 2. No. of time steps for the agent kinematic algorithm to acquire a solution for added subpoints

It can be seen that identical multi-agent control systems used in different experimental manipulation structures (ST1-ST5) are independent of such structural changes.

### 4.2 Redundant manipulation structures

This section presents the experiments conducted on redundant manipulation structures (Figure 10). They have more degrees of freedom than are necessary to execute a certain task. All demonstrated structures have 6-DOF with different joint types. ST6 is based on ST1 with added axes redundancy. ST7 and ST8 introduce non-orthogonal linear and nonlinear superposition. ST9 is a prototype of the rotational structural redundancy. The motion task for all the manipulators is T1 and the initial point and linear velocity are equal to zero.

As long as the systems of the non-cooperative DSC agents are redundant, each of the errors is trying to be minimized by several components at the same time. Thus, in general, a larger overlap and more time steps are needed to find a proper solution, which can be seen in Figure 11. Agent kinematic algorithms converge to target points and successfully find appropriate joint solutions. The time steps for acquiring solutions \( (T_{ST}(P_i)) \) grow with the structural complexity (Table 3) and also depend on the communication speed for exchanging information about current motion plans between DSC agents. With an increased number of DSC agents controlling the manipulator, the data quantity for exchanging also increases, which can affect system performance. The rotational redundancy of ST9 has a larger influence on the deviations around the target point than the other structures. This is mainly because small rotational moves result in large position changes of the end-effector.

![Figure 10. Experimental redundant manipulation structures](image)

| ST6 | P1 | P2 |
|-----|----|----|
|     | 42 | 100|
| ST7 | 29 | 51 |
| ST8 | 70 | 105|
| ST9 | 239| 259|

Table 3. No. of time steps required for the agent kinematic algorithm to acquire a solution for ST6-ST9

Figure 12 shows the corresponding movement execution of the solutions acquired by the same agent kinematic algorithms, which managed to solve the problems introduced by structural and system redundancy. The proposed multi-agent control approach is, in case of redundancy, also resistant to component failures. The still-working units perform the same control procedure as before because the autonomy of the agent system supports such events by default.
5. Real-world experiments

This section describes experiments conducted in the real world. The usability of the proposed approach is demonstrated for fast system reconfiguration and its flexibility to adapt to new circumstances with no changes made at the logic and/or informational level. In this way, a highly adaptable manipulation system can be constructed.
Our experimental manipulation structure consists of three identical translational components and a pencil as a tool (Figure 13). Each component was given autonomous control with the associated DSC agent. Each DSC agent was implemented on the real-time controller (NI-sbRIO). They represented one manipulator component composed of the ISEL linear drive and servo actuator CoolMuscle with integrated position, velocity and torque sensors. The moderator agent was running on a PC; the purpose of the moderator agent was to submit tasks to the DSC agents. All the agents communicated and exchanged information through TCP/IP. Communication between the agent control and actuator layer was implemented by serial communication. The experiments varied with manipulator structural configuration (Figures 13 and 14b). The task in both examples was to execute the end-effector (tip of the pencil) movement in the shape of a circle defined as a set of points:

\[
T_3 = \begin{bmatrix}
  [r \cdot \sin \varphi] \\
  [r \cdot \cos \varphi] \\
  [z]
\end{bmatrix}, \quad \begin{bmatrix}
  [r \cdot \sin \varphi] \\
  [r \cdot \cos \varphi] \\
  [0]
\end{bmatrix}; \quad \varphi_i = \varphi_{i-1} + 10^\circ
\]

In the experiment, the radius of the circle (r) is 50mm and the number of target points (i) is 36. In the first example, all the components were orthogonal to the global xyz coordinate axes (Figures 13 and 14b). The second experiment differs in the structural configuration between the components by changing the orientation angle of one of the translational components.

First, the moderator agent formed a model of the manipulation structure and sent it together with the movement task to all the DSC agents. The DSC agents immediately started searching for joint solutions for the demanded task points. Figure 14a shows the end-effector position and joint values as the agent kinematic algorithm searched joint solutions for target points in the shape of a circle. It can be seen that in both experiments, the DSC agent used different joint values to achieve the specified target because of its current structural configuration. In the second example, the component movement of DSC agent 2 results in end-effector motion in both the x and y directions. To reach the target position, DSC agent 1 had to compensate for the movement from DSC agent 2. After all the joint solutions had been obtained, the movement was executed in the real world. The results of both experimental movement executions are shown in Figure 14b. From the photos, it can be concluded that multi-agent execution of the end-effector in the shape of a circle was successfully achieved. Experiments were conducted for two different structural configurations with the identical agent control system.

Figure 14. End-effector position and joint values in the process of agent kinematic algorithms searching for joint solutions (a) and photos of the performed multi-agent execution in the real world (b)
6. Conclusions

In this paper, a new multi-agent control method is presented. It is designed to control modular, reconfigurable and scalable manipulative systems. The control system of distributed and autonomous units dynamically responds to changes in the environment and/or to its own ones, while trying to fulfill the goals of the manipulative system. In contrast to existing approaches, the presented level of control autonomy allows us to dynamically change the number, type and structure of manipulative components with no change at the logical and architectural levels. Additionally, the proposed agent system control can manage redundancy in the system and in the manipulation structure.

New agent control logic dedicated to solving kinematic problems of dynamically composed manipulative structures is demonstrated. Based on direct and differential kinematic equations, correct end-effector pose and velocity computations are guaranteed. This approach avoids some of the problems that occur in computing inverse kinematics (e.g., singularity, inflection points). The agent kinematic algorithm makes it possible to expand the number of possible motion plans and thus increases the quality of the motion solution. However, this also increases the computation time, so the proper ratio between quality and time should be identified. The time step in the planning loop is crucial to guarantee non-outdated agent actions. A new evaluation function was defined to assess the motion plans generated by the agents. It can support different sets of goal functions for each agent and thus easily adjust the system behaviour to specific needs.

It is shown by the demonstrations on a wide range of manipulators, which are dynamically recomposed of the same components, that the proposed multi-agent system can successfully control any open kinematic chain for the demonstrated tasks. A virtual agent environment was developed within the NI-LabView programming software. Autonomous agents were deployed on a PC and embedded platforms, which communicated over TCP/IP. Real-world experiments confirm the usability of the proposed approach and are in agreement with the simulation results. The experiments show that the presented agent control logic can be used in a wide range of different applications (milling, reforming machines, etc.). Of course, the specifics of particular technological processes might require adaptation of the agent system goal definitions (adding pressure, force contact, human safety and other system interactions).

Our future work will focus on multi-agent control of closed kinematic chains with human-robot interactions. New agent cooperation methods will be investigated, which might help us to solve some currently unresolved issues with advanced automatic motion planning and adaptive behaviour in an environment that is not known in advance.

7. Acknowledgements

The work presented in this paper is financially supported by the Slovenian Technology Agency, Ministry of Higher Education, Science and Technology of the Republic of Slovenia and by the European Union through the European Social Fund. The operation was implemented in the framework of the Operational Programme for Human Resources Development for the Period 2007-2013, Priority axis I: Promoting entrepreneurship and adaptability. Main type of activity 1.1: Experts and researchers for competitive enterprises. Acknowledgement for support also goes to the company Trimo, d.d..

8. References

[1] H. G. Stork, “Cognitive Systems and Robotics” in the ICT Programme of the European Commission,” ERCIM NEWS, vol. 84, pp. 2-3, 2011.
[2] D. Rüdiger, A. Tamin and A. Antonis, “Intelligent and Cognitive Systems,” ERCIM NEWS, vol. 84, pp. 12-13, 2011.
[3] J. Z. Pan and R. V. Patel, "Reconfigurable distributed real-time processing for multi-robot control: Design, implementation and experimentation," Robotica, vol. 22, pp. 661-679, 2004.
[4] Z. Li, W. W. Melek and C. Clark, "Decentralized robust control of robot manipulators with harmonic drive transmission and application to modular and reconfigurable serial arms," Robotica, vol. 27, pp. 291-302, 2009.
[5] G. Liu, S. Abdul and A. A. Goldenberg, "Distributed control of modular and reconfigurable robot with torque sensing," Robotica, vol. 26, pp. 75-84, 2008.
[6] D. Krizaj and P. Butala, "Design of reconfigurable assembly systems based on logical reasoning and constraints," CIRP Journal of Manufacturing Systems, vol. 34, no. 5, pp. 475-484, 2005.
[7] H. Y. K. Lau, A. Ko A and T. L. Lau, "A decentralized control framework for modular robots," Proceedings 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1774, 2004.
[8] T.C. Lueth and T. Laengle, "Task Description, Decomposition, and Allocation in a Distributed Autonomous Multi-Agent Robot System," Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94, vol. 3, pp. 1516, 1994.
[9] J. Ferber, Multi-agent systems – An introduction to distributed artificial intelligence. Pearson Education, 2005.
[10] M. Wooldridge, *An Introduction to Multi-agent Systems*, 2nd ed.: John Wiley & Sons Ltd, 2009.

[11] P. Lucidarme, "Distributed multi-agent architecture for humanoid robots," *3rd National Conference on “Control Architectures of Robots” Bourges, Presses Universitaires d’Orléans*, pp. 67-75, 2008.

[12] P. Nebot and E. Cervera, "An integrated agent-based software architecture for mobile and manipulator systems," *Robotica*, vol. 25, pp. 213-220, 2007.

[13] J. Sluga, V. Zaletelj and A. Žemva, "Development of a modelling environment for efficient system reconfiguration based on MAS," *Electrotechnical Review: Ljubljana, Slovenija*, vol. 77, no. 4, pp. 179-187, 2010.

[14] B. Siciliano, L. Sciavicco, L. Villani and G. Oriolo, *Robotics: modelling, planning and control*. Springer-Verlag London Limited, 2009.