Synchronizing Robot Motions in Cooperative Tasks

Anton Florin*, Anton Silvia*

**Automation and Applied Informatics Department, University Politehnica of Bucharest, RO 060032 Bucharest, Romania (Tel: 021-421-9314; e-mail: anton@cimr.pub.ro).*

Abstract: This paper discusses the problem of synchronizing multiple robots in cooperative tasks. The problem is divided in two parts: first the problem of synchronizing robots which are handling large objects that cannot be manipulated only by one robot (hard connected robots) is discussed; the problem is approached from the point of view of path planning, kinematics, and movement synchronization. An alternative of control using the force control and dynamics which solved by using a decentralized control structure is also presented. The second problem concerns robot synchronization in a shared workspace; here is presented a method of control for collision avoidance and time optimization for the robot tasks (assembly / disassembly). In the conclusion section the results from the experiments conducted on two SCARA type robots (Adept Cobra s600) are presented.

Keywords: Motion Synchronization, Robot control, Cooperative robots, I/O and Ethernet Communication.

1. INTRODUCTION

Recent research activities have been directed toward designing multiple robot systems engaged in collective behaviour. Such systems are of interest for several reasons:

- tasks may be inherently too complex for a single robot to be accomplished (or impossible to be realized), or performance benefits can be gained from using multiple robots;
- building and using several simple robots can be easier, cheaper, more flexible and more fault tolerant than having a single powerful robot for each separate task.

Studies concerning multiple-robot systems naturally extend research on single-robot systems, but represent also a discipline itself: multiple-robot systems can accomplish tasks that no single robot can accomplish, since ultimately a single robot, no matter how capable, is spatially limited, Caccavale and Uchiyama (2008).

Modelling and controlling multiple robotic manipulators handling a constrained object requires more sophisticated techniques compared with a single robot working alone. Since the theory employed for cooperative robots is independent of their size, one can think of them as mechanical hands.

Robot hands (as well as cooperative robots), may find many areas of application nowadays. Many benefits can be obtained by using them in industrial manufacturing. A typical example is in a flexible assembly, where the robots assemble two parts into a product, Guidino-Lau and Arteaga (2006).

Robots working in a cooperative manner can also be used in material handling, e.g., transporting objects beyond the load carrying capacity of a single robot. Furthermore, their employment allows improving the quality of tasks in the manufacturing industry requiring high precision.

From another point of view, cooperative robots are indispensable for skilful grasping and dexterous manipulation of objects. However, the literature about experimental results on the modelling, simulation and control of systems of multiple manipulators holding a common object is rather sparse.

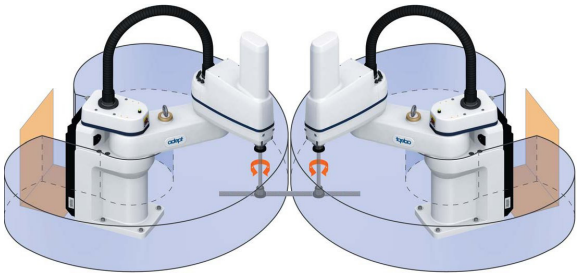

Fig. 1. Robot cooperative task.

Our research is based on two SCARA type Adept Cobra s600 robots (Fig. 1). The robots are used in tasks of cooperative movement (object handling) and in tasks where the robots share a common workspace. The paper gives a solution for movement synchronization based on Ethernet and I/O communication as an alternative to force control techniques, Anton (2008).

2. VIRTUAL LINKAGE

The task of manipulating objects requires accurate control of internal forces. Williams and Khatib (1993) proposed the virtual linkage, as a model of internal forces associated with multi-grasp manipulation. In this model, grasp points are connected by a closed, non-intersecting set of virtual links.

The same approach was used in this paper, but in our case the robots are in a client-server relationship (Fig. 2). Based on the spatial position and orientation of the robots, and using the kinematic models of the robots, the *server robot* computes the trajectory for both robots and the *client* only execute the moves in a synchronized manner. In this approach, the robots can be controlled without force sensors because the movements are composed by small segments of linear movements synchronized using the I/O lines.

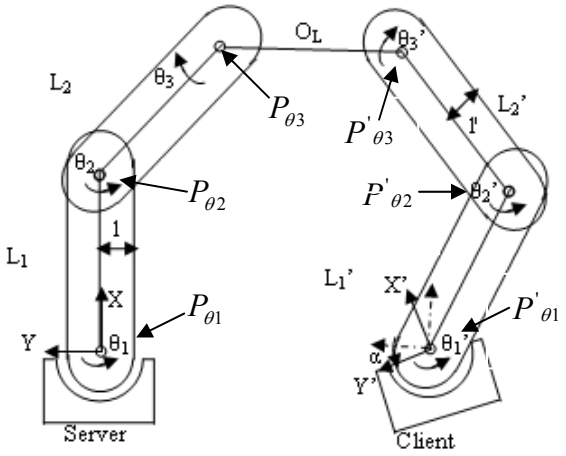

Fig. 2. The virtual linkage.

The position and orientation of the client robot base (relative to the base coordinate system of the server robot) ($X^{'}, Y^{'}, Z^{'}$ and α) and the dimensions of the handled object are well known, also the grasping positions are precisely trained.

Using only small linear movements the two robots have three types of cooperative movements expressed relative to the handled object:

- *translations* (in this mode both robots are moving in the same direction using the same speed and path length);
- *rotations* (here the robots have different directions of movement, different speeds, and different path lengths). The robots start the movement and are stopping it at the same moment of time, so the duration of the movement is the same despite the different path lengths;
- movements *composed* from the two above basic types.

In the first case the problem is very simple:

Having a small movement Δx on X axis of the server robot, a movement of the client robot results, being composed on the two axes:

$$\Delta x \rightarrow \begin{cases} \Delta x' = \Delta x \cos(\alpha) \\ \Delta y' = -\Delta x \sin(\alpha) \end{cases} \tag{1}$$

and a small movement Δy on Y axis generates:

$$\Delta y \rightarrow \begin{cases} \Delta x' = -\Delta y \sin(\alpha) \\ \Delta y' = \Delta y \cos(\alpha) \end{cases} \tag{2}$$

In the second case, the rotations can be executed around any point between the grasping points (included in the O_L segment). Let that point be x, then the O_L segment is spitted in two segments: $l_{x_1} = dist(G_S, x)$ and $l_{x_2} = dist(x, G_C)$, where G_S and G_C are the grasping points of the robots relative to the object $O_L = dist(G_S, G_C)$. If the object must be rotated with the amount $\Delta\theta_L$ in counter clockwise direction, this rotation will be generated by a rotation θ_{4_S} and a movement P_{x_S}, P_{y_S} of the server robot:

$$\Delta\theta_L \rightarrow \begin{cases} P_{x_S} = P_{x_{Si}} - l_{x_1} \sin(\Delta\theta_L) \\ P_{y_S} = P_{y_{Si}} + l_{x_1} \cos(\Delta\theta_L), \\ \theta_{4_S} = \theta_{4_{Si}} - \Delta\theta_L \end{cases} \tag{3}$$

where $P_{x_{Si}}$, $P_{Y_{Si}}$ and $\theta_{4_{Si}}$ is the initial position of the server robot and the rotation of the 4th segment:

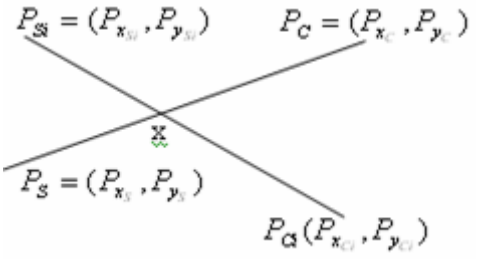

Fig. 3. Initial and final points for the segment rotation.

In the triangles $\Delta(x, P_S, P_{Si})$ and $\Delta(x, P_C, P_{Ci})$ (Fig. 3) the relation below holds:

$$\frac{l_{x_1}}{l_{x_2}} = \frac{\Delta x_S}{\Delta x_C} = \frac{\Delta y_S}{\Delta y_C}, \tag{4}$$

where:

$$\begin{cases} \Delta x_S = P_{x_S} - P_{x_{Si}} \\ \Delta y_S = P_{y_S} - P_{y_{Si}} \end{cases} \text{ and } \begin{cases} \Delta x_C = P_{x_C} - P_{x_{Ci}} \\ \Delta y_C = P_{y_C} - P_{y_{Ci}} \end{cases} \tag{5}$$

From Eq. 4 and Eq. 5 it is:

$$\Delta\theta_L \rightarrow \begin{cases} P_{x_C} = \dfrac{l_{x_1} P_{x_{Ci}} + (P_{x_S} - P_{x_{Si}}) l_{x_2}}{l_{x_1}} \\ P_{y_C} = \dfrac{l_{x_1} P_{y_{Ci}} + (P_{y_S} - P_{y_{Si}}) l_{x_2}}{l_{x_1}} \end{cases} \tag{6}$$

But P_{x_C} and P_{Y_C} are expressed in the coordinate system of the server robot; to compute the correct coordinates we replace Eq. 5 and Eq. 4 in Eq. 1 and Eq. 2 and then add the results:

$$\Delta\theta_L \rightarrow \begin{cases} P_{x'_C} = P_{x'_{Ci}} - l_{x_2} \sin(\Delta\theta_L + \alpha) \\ P_{y'_C} = P_{y'_{Ci}} + l_{x_2} \cos(\Delta\theta_L - \alpha) \end{cases} \tag{7}$$

The rotation of the 4th link is:

$$\theta_{4_C} = \theta_{4_{Ci}} - \Delta\theta_L \tag{8}$$

3. AN ALTERNATIVE TO DECENTRALIZED CONTROL

In the force control approach a control scheme of a decentralized architecture has to be used, so that the input torque for each robot is calculated in its own joint space and

takes into account motion constraints rather than the held object dynamics.

In the case of a multiple robot system, each robot has real-time access only to its own state information and can only infer information about the other robots' grasp forces through their combined action on the object. In the decentralized control structure, the object level specifications of the task are transformed into individual tasks for each of the cooperative robots. Local feedback control loops are then developed at each grasp point.

The task transformation and the design of the local controllers are accomplished in consistency with the augmented object and virtual linkage models, Braun et. al. (2004). The overall structure of the decentralized control structure introduced by Khatib et. Al (1995) is shown in Fig. 4, where F_{force} and F_{int} are the input force and the input internal forces, f_i and $f_{s,i}$ are the forces at the i^{th} grasp point and the sensed forces at the i^{th} grasp point.

The above control structure will work correctly if the object is rigid (interraction via environment) and there is no slippage at the grasp points. Gripper slip in the real system will result in errors in the grasp kinematic computation and inconsistencies with the virtual linkage model.

To compensate for these effects, some level of communication between the different platforms is needed for updating the robot state and modifying the task specifications. The rate at which this communication is required is much slower than the local servo control rate. Such communication can be achieved over a radio Ethernet link (at 10-20 Hz).

Our method of control uses only the communication in order to move the objects in a synchronized manner (interraction via communication). This approach is based on knowing the initial positions of the robots, the dimensions and the desired final position of the handled object.

The control structure is based on the client-server architecture. Here one robot (the server) knows all the above initial data, also knows the kinematics of the client robot and computes off-line the path which the client robot must follow. The path is decomposed in small segments and the robots synchronize the motion using high speed I/O lines.

This approach allows us to solve the following problems: it is not required that the object be rigid (there is no force sensing), and there is no slippage of the object because the small segments of synchronized movements. Also because all the computation (path planning) is executed off-line (when the robots are not moving) and the movement control programs are almost identical (the single difference are the values stored in the position variables), a faulty synchronization due to instructions execution time is prevented.

4. SHARED WORKSPACE

In the case of multiple robots moving within a common environment, they typically attempt to avoid collisions. This may be viewed as a problem of resource conflict, which may be resolved by introducing, e.g., traffic rules, priorities, or communication architectures.

From another perspective, path planning must be performed taking into consideration other robots and the global environment; this multiple-robot path planning is an intrinsically *geometric* problem in space-time configuration. Prioritization and communication protocols – as well as the internal modelling of other robots – all reflect possible variants of the *group architecture* of the robots.

As an example, traffic rules are commonly used to reduce planning cost for avoiding collision and deadlock in a real-world environment, such as a network of roads.

For multiple robots to inhabit a shared environment, manipulate objects in the environment, and possibly communicate with each other, a mechanism is needed to resolve resource conflicts.

Resource conflict arises when a single indivisible resource is requested by multiple robots. This issue has been studied in many works, notably the mutual exclusion problem in distributed algorithms and the multi-access problem in computer networks. With multiple robots, resource conflict occurs when there is a need to share space, manipulable objects or communication media.

Therefore the discussion is centred on the space sharing problem, which has been studied primarily via multiple-robot path planning and the collision and deadlock avoidance problems.

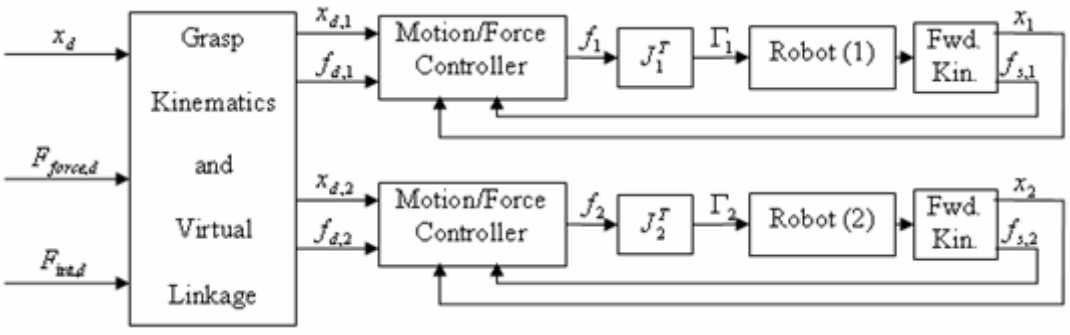

Fig. 4. Decentralized control structure.

Rude, (1994) considers the problem of crossing an intersection: event transforms into the local space-time coordinate frame of a robot are applied, and each robot (i) iteratively updates the local frame and its objects, (ii) evaluates collision risk, and (iii) generates a modified path depending on the collision risk. However, researchers considering real-world multi-robot systems typically conclude that planning paths in advance is impossible. Thus, robots are often restricted to prescribed paths or roads, with rules (like real life traffic laws) and communications used to avoid collision and deadlock, Stroupe et. al. (2005).

Grossman (1988) classifies instances of the traffic control problem into three types: (i) restricted roads, (ii) multiple possible roads with robots selecting autonomously between them, and (iii) multiple possible roads with centralized traffic control. When individual robots possess unique roads from one point to another, no conflict is possible; when there is global knowledge and centralized control, it is easy to prevent conflict. Thus, the interesting case is (ii), where robots are allowed to autonomously select roads. Restricted roads are highly suboptimal, and that the autonomous road choice coupled with a greedy policy for escaping blocked situations is far more effective (i.e. "modest cooperation", where robots are assumed to be benevolent for the common good of the system).

In our case the robots share a common workspace for assembly/disassembly purposes. The parts which compose the assembly are stored in stacks in the private workspace while the objects are assembled in the shared workspace. Here, because the robots must access the same position in space or very close positions at different speeds (the aim is to assemble as many object as it is possible), there is no possibility to plan an optimal path for the robots and then synchronize them.

The idea is to create two movement modes:

- A free movement mode, where the robot has no movement constraints relative to the other robot. This regime is used in the private workspace.
- A second movement mode where the robot is tying to access the shared workspace.

The second movement mode is achieved by using the kinematic model of the robots and also knowing the positions and orientation of each robot base relative to the other robot. In this case one can compute the positions $P_{\theta 1}, P_{\theta 2}, P_{\theta 3}$ and

of the first three robot articulations (for SCARA robots) where $P_{\theta 1}$ and $P'_{\theta 1}$ are the positions of the robots base. Then a volume is defined around each segment; the shortest distance from the border of this volume to the link is l (see also Fig. 2).

At run time, the robot which is accessing the shared workspace has a privileged status until it has done the job. The robot (R2) which wants to access the shared workspace checks the status of the other robot (R1). If R1 is accessing the shared workspace, then R2 computes the distances from the second and the third articulations ($P'_{\theta 2}, P'_{d3}$) to the links L_1 and L_2 of R1 and the distance from each point $P(x, y)$ to each link $L(P_0, P_1)$, where the points P_0 and P_1 are the locations of the articulations computed using the kinematic model ($P_{\theta 1}, P_{\theta 2}, P_{\theta 3}$,and have the coordinates $P_0(x_0, y_0)$ and $P_1(x_1, y_1)$); this distance is given by equation (9).

$$d(P, L) = \frac{(y_0 - y_1)x + (x_1 - x_0)y + (x_0 y_1 - x_1 y_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}} \tag{9}$$

If the distances d are greater than $2l$ then the movement can be executed, else the projection point of $P(x, y)$ on $L(P_0, P_1)$ is computed, and if the projection point is included in the segment $L(P_0, P_1)$, then the movement cannot be done. If the projection point is not included in the L segment, then the distance between $P(x, y)$ and the closest point of the segment ($P_0(x_0, y_0)$ or $P_1(x_1, y_1)$) is computed. If this distance is greater than $2l$, then the movement can be executed. If the second robot cannot execute the movement it will compute the closest point form the destination point which is out of the reach of the privileged robot and moves there. After that, it monitors the activity of the privileged robot and tries again to reach the destination. All movements are decomposed into small movement segments; in this way the robots have more flexibility and the risk to collide is reduced.

The same approach is used when the robot R1 (with privileged status) is moving. In this case, if the movement cannot be done due to collision risk, the robot R2 will be informed and will clear the path.

This type of movement testing and synchronization can be compared with the behaviour of magnets. The privileged robot is not influenced by the other robot; instead the other robot is pushed by the "magnetic force" of the privileged robot and is "attracted" by the destination position. This behaviour can be described by the algorithm given in Fig. 5 (SH_WS is the shared workspace).

5. COMMUNICATION

5.1 Interaction via environment

Modelling the behaviour (intentions, beliefs, actions, capabilities, and states) of other robots can lead to more effective cooperation between robots.
Communications requirements can also be lowered if each robot has the capability to model the behaviour of other cooperative robots.

Modelling of other robots entails more than implicit communication via the environment or perception: it requires that the modeller has some representation of another robot, and that this representation can be used to make inferences about the actions of the other robot.

In applications where robots must cooperate, modelling has been explored most extensively in the context of manipulating a large object.

Many solutions have exploited the fact that the object can serve as a common medium by which the robots can model each other.

In a rigid body carrying task, each robot uses a probabilistic model of the other agent. When a risk threshold is exceeded, an agent communicates with its partner to maintain coordination, Miyabe et. al. (2004).

The simplest, most limited type of interaction occurs when the environment itself is the communication medium, and there is no explicit communication or interaction between robots. This modality has also been called "cooperation without communication" by some researchers. Systems that depend on this form of interaction have been discussed in Gueaieb et. al. (2003).

This type of interaction is suited for robots with force sensors which are connected in virtual links using the handled objects. In the case of cooperative handling due to problems presented in section 3 (gripper slip, non rigid objects) an Ethernet link is also required.

5.2 Interaction via communications

This form of interaction involves a type of explicit communication; in which an media access protocol is used for inter-robot communication, Kawasaki et. al. (2006), Martinez-Rosas et. al. (2006).

Our approach uses only I/O and Ethernet communication in order to synchronize the robot movements. There are two types of strategies used:

- In the case of collaborative object handling the first robot computes offline a set of points which the second robot will follow; here before the movement begins the robots make a communication over Ethernet for sending/receiving the set of points; at online stage movements are synchronized by activating/deactivating an I/O line to signal the movement start/stop.
- The other case when the robots use a shared workspace assumes that I/O lines are used to signal if a robot has a privileged status (whether it is the first which accesses the shared workspace), and to signal the beginning and the end of each movement, as well as the Ethernet line highly used to transmit the current position.

Fig. 5 presents an algorithm wich allow the access to a shared workspace for two robots without the risk of collision. Using the kinematics, each robot is aware of the position of the other robot in space and the risk of collision is determined. Based on the robot priority and the communication, the workspace is cleared in order that the robot with the biggest priority to access the workspace.

6. CONCLUSIONS

The paper discusses two types of robot applications with robot movement synchronization needed.

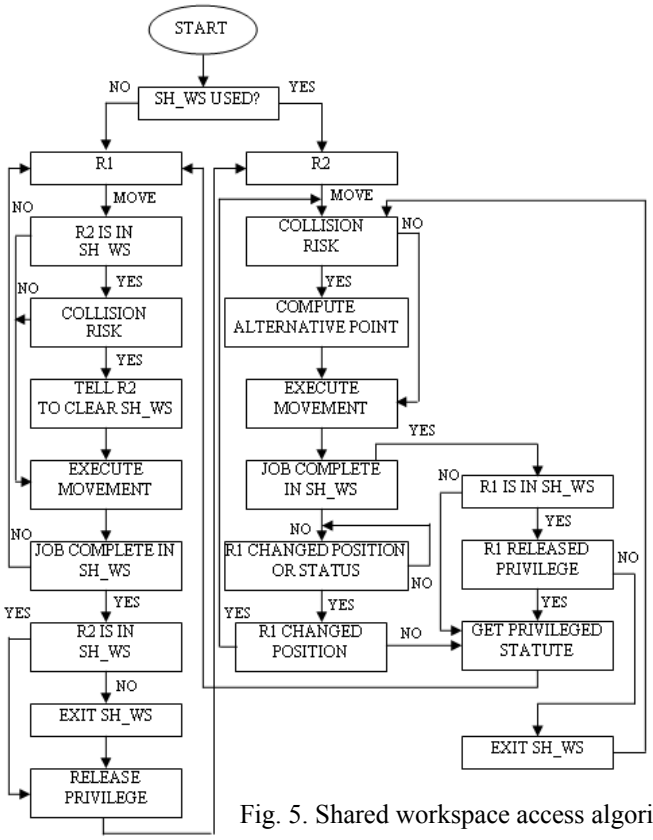

Fig. 5. Shared workspace access algorithm.

The first discussion case (Fig. 6) can be applied in cases where the robots are handling an object which can be heavy enough that a robot could not handle by itself, or the object has a shape that is very difficult to be grasped and handled by one robot; the robots are connected and each movement must be synchronized.

Fig. 6. Collaborative object handling.

In this first case we managed to use two Adept Cobra s600 industrial robots to handle an object without sensing the forces using force sensors, but by using high speed I/O lines to synchronize robot movements. One robot which knows how the handled object will be moved, computes the trajectory for both robots based on the virtual linkage and the kinematics of robots, and then each movement is synchronized using the I/O lines.

This method is much cheaper than using force sensors, is reliable and solves some problems which appear to decentralized force control (here the object can be made of any material rigid or not, the grippers can be vacuum grippers, etc.) all that because the force sensing is not needed anymore.

Fig. 7. Shared workspace access.

In the second case (Fig. 7) the problem was to synchronize the movements of robots in such way that the robots will access a shared workspace without the risk of collision using an optimal method which simulates the behaviour in "magnetic fields" (the low privilege robot is pushed back by the high privilege robot but is attracted by the destination position).

The implementation of this second case is based on high volume of Ethernet communication. During the tests it has been observed that due to communication delays, in some situations where the robots access the shared workspace, there is a small delay between consecutive movements of the robots (despite this problem synchronization depending only on the high speed I/O lines).

The problem can be solved using the method of interaction via environment - in our case the controller memory. The robots can be controlled using a single Adept Smart Controller CX in dual configuration. This solution solves the communication delay adding a plus of performance.

REFERENCES

Anton, S. (2008). *Integrating multiple robot-vision systems in intelligent assembly/disassembly structures*. PhD thesis, UPB Automatics and Industrial Informatics Department.

Braun, B.M., Starr, G.P., Wood, J.E. and Lumia, R. (2004). A framework for implementing cooperative motion on industrial controllers. *IEEE Trans. Robot. Autom.* **20**, 583-589

Caccavale, F. and Uchiyama M. (2008). Cooperative Manipulators. *Springer Handbook of Robotics,* 701-716

Grossman, D. (1988). Traffic control of multiple robot vehicles. *IEEE Journal of Robotics and Automation.* 4, 491–497.

Gudiño-Lau, J. and Arteaga, M.A. (2006). Dynamic Model, Control and Simulation of Cooperative Robots: A Case Study, *Mobile Robots, Moving Intelligence,* ARS/pIV, Germany

Gueaieb, W., Karray, F., and Al-Sharhan, S. (2003). A robust adaptive fuzzy position/force control scheme for cooperative manipulators. *IEEE Trans. Contr. Syst. Technol.* **11**, 516-528

Kawasaki, H., Ueki, S., and Ito, S. (2006). Decentralized adaptive coordinated control of multiple robot arms without using a force sensor. *Automatica* **42**, 481-488

Khatib, O., Yokoi, K., Chang, K., Ruspini, D.C., Holmberg, R., Casal, A., and Baader, A. (1995). Force Strategies for Cooperative Tasks in Multiple Mobile Manipulation Systems. *Proc. of the International Symposium on Robotics Research*, 333-342.

Martinez-Rosas, J.C., Arteaga, M.A., and Castillo-Sanchez, A.M. (2006). Decentralized control of cooperative robots without velocity-force measurements. *Automatica* **42**, 329-336

Miyabe, T., Konno, A., Uchiyama, M., and Yamano, M. (2004). An approach toward an automated object retrieval operation with a two-arm flexible manipulator. *Int. J. Robot. Res.* **23**, 275-291

Rude, M. (1994). Cooperation of mobile robots by event transforms into local space-time. *IEEE/RSJ IROS*. 1501–1507.

Stroupe, A., Huntsberger, T., Okon, A., and Aghazarian, H. (2005). Precision Manipulation with Cooperative Robots. *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, 235-248

Williams, D. and Khatib, O. (1993). The Virtual Linkage: A Model for Internal Forces in Multi-Grasp Manipulation, *Proc. IEEE Int. Conf. Robotics and Automation*, 1025-1030.