Synchronization of Heterogeneous Multi-Robotic Cell with Emphasis on Low Computing Power

Martin Juhás * and Bohuslava Juhásová

Faculty of Materials Science and Technology in Trnava, Slovak University of Technology in Bratislava, 917 24 Trnava, Slovakia; bohuslava.juhasova@stuba.sk
* Correspondence: martin_juhas@stuba.sk

Received: 26 June 2020; Accepted: 23 July 2020; Published: 27 July 2020

Abstract: This paper presents a time-synchronization solution for operations performed by a heterogeneous set of robotic manipulators grouped into a production cell. The cell control is realized using master–slave architecture without an external control element. Information transmission in a cell is provided by a TCP/IP channel in which communication is ensured via sockets. The proposed problem solution includes an algorithm, which is verified and validated by simulation and tested in real environment. This algorithm requires minimal computational power thanks to an empirically oriented approach, which enables its processing directly by the control unit of each participating element of the robotic cell. The algorithm works on the basis of monitoring and evaluating time differences among sub-operations of master and slave devices. This ensures defined production cycle milestones of each robotic manipulator in the cell at the same time are attained. Dynamic speed adaptation of slave manipulators utilizing standard instructions of their native language is used. The proposed algorithm also includes a feedforward form of operations synchronization which responds to changes in the operating cycle of the master manipulator. The application of the solution proposal is supplemented with a visualization part. This part represents a complementary form of designed solution implementation.

Keywords: time-based synchronization; heterogeneous multi-robotic cell; socket communication; low computational power; native language application

1. Introduction

A current significant challenge for industrial production is vision substantiation of a future factory within the concept of Industry 4.0, with the aid of the Internet of Things (IoT) concept. The implementation of this concept into production requires high flexibility and adaptability of production lines and their smaller units. The deployment of this concept in practice is widely supported in Germany, and the first solutions in isolated production systems were developed there [1,2]. Qin et al. [3] note that the Internet of Things is a well-known concept that represents the next generation of products and communication among them. It has a direct correlation with the Industry 4.0 standard, where the existence of smart products is one of the prerequisites for intelligent manufacturing implementation [4].

In a smart factory, individual customer orders determine manufacturing processes and the associated supply chains. This results in the need for high production flexibility with shorter production times, which require the implementation of measures to improve production efficiency, often at a low cost associated with solving these problems. The new term “smart factory” is introduced here to refer to the current trend of automation and data exchange in manufacturing technologies. It includes cyber-physical systems, the Internet of Things, and cloud computing, as declared by Hermann et al. [5], Jasperneite [6], and Kagermann et al. [7]. One of the important
indicators for such production is decentralized decision making, that is, the ability of equipment to make its own decisions, which are the most independent inside closed production complexes. In the case of essential decisions and conflicts, it is clearly necessary to assign tasks to the next level of production control [5]. Closed production complexes require continuity of individual manufacturing operations, synchronization of individual production facilities, and the possibility of rapid adaptation and production time changes of individual production facilities, depending on the control structure of such complexes.

An alternative deployment area of robotic manipulators with a potential requirement for synchronization of their activities is healthcare, particularly related to the rehabilitation of patients. Techniques dealing with robot-assisted therapy are included, for example, in the works of authors Chang and Kim or Yoon et al. [8,9].

Various methods of synchronization and control of individual production operations and processes are currently used. It is necessary to differentiate whether these are autonomous mobile systems or manipulators and what usage is expected of them. Many published works deal with the synchronization of mobile platform activities. Rubenstein et al. [10] offer a solution for the synchronization of a large group of mobile robots as an open-source, low cost robot, designed to test collective algorithms on hundreds or thousands of robots accessible to robotics researchers. This solution allows for easier testing of algorithms designed to control robot groups because these control algorithms, due to their cost, time, and complexity, are confirmed only through simulations. Popular synchronization methods are also based on the observation of nature, for example, fireflies as presented by Werner-Allen et al. [11]. A modification of this approach is intended to operate on systems that use a communication channel where contention and delays are possible. In addition, the coordination mechanisms that enable the execution of cooperative tasks with modular robotic systems are presented in the contribution of Baca et al. [12]. They describe the implementation of a tight cooperation strategy through Intra M-Robot communication based on a closed-loop discrete time method and the remote clock across the robot configuration enables proper coordination inside the colony.

The work of Chung and Slotine [13] presents a new synchronization tracking control law that can be directly applied to the cooperative control of multi-robot systems and oscillation synchronization in robotic manipulation and locomotion, where a common desired trajectory can be explicitly given.

Rodriguez-Angeles et al. [14] describe a controller utilization that solves the problem of position synchronization of two (or more) robotic systems, under a cooperative scheme, in the case when only position measurements are available. In the work of Yasuda [15], a Petri-net-based prototyping tool is presented to implement the control flow of parallel processes in multiple robot systems. The next variant of synchronization is presented by Markus et al. [16], where the coordination control of two flexible joint robotic manipulators using flat outputs is implemented by means of simulations. The differential flatness technique of trajectory generation enables easy estimation of synchronization parameters and trivializes stabilization of these trajectories around predefined points. Bouteraa et al. [17] describe a new adaptive algorithm, which was proposed for synchronization and trajectory tracking of multiple robot manipulators. The same authors also discuss other techniques in this problem area. They describe the possibility of designing decentralized control laws to cooperatively command a team of general actuated manipulators in the article “Distributed synchronization control to trajectory tracking of multiple robot manipulators” [18], or an approach to position synchronization of multiple robot manipulators based on emergent consensus algorithms [19]. Synchronization of activities in task-oriented robotic rehabilitation training using iterative learning synchronization (ILS) and immediate error correction (IEC) techniques is addressed by Duschau-Wicke et al. [20].

In the above-mentioned cases, the authors based their solutions on complex mathematical algorithms and derivation of complex relations, or by exploring new approaches using specialized hardware. These methods require investment in hardware infrastructure, which is their major disadvantage.
In some cases in homogenous production complexes, it is possible to use tools that are directly implemented in control systems, such as the MultiMove option of ABB robots [21,22] or the RoboTeam software of KUKA [22,23].

However, if limited possibilities exist to upgrade infrastructure and it is desired to use available hardware that may not be from the latest production series, there could be a serious problem with the use of the previously described solutions. We used an alternative approach in a production or technical process of a simple solution implementation even in production facilities without the possibility of using sophisticated methods of robotic set or robotic cell synchronization. Our aim is the simplest solution possible in terms of computing power demands. The basis is to design the least complex algorithms that can be easily implemented in existing controllers in their native programming languages. Therefore, our goal is to develop a solution in which synchronization algorithms can be performed directly on the control units of robotic manipulators. This is based on the utilization of our rich empirical knowledge and experiences in algorithms, in addition to the implementation of various tasks in the field of robotics or modeling and visualization of processes.

The main idea does not lead to a specific use or solution to a precisely specified problem, for example in a production process. The aim of the proposed solution can have a wide range. From the analyzed areas it is possible to use this solution either in production or in healthcare in rehabilitation. Another area of application could be the control of collaboration-oriented workplaces with a master manipulator connected to movements of a human as a cell control element. Finally, it could be used in presentation events oriented to Industry 4.0 or the latest trend, Internet-of-Robotic-Things (IoRT).

2. Problem Area Definition

The multi-robotic cell consists of a set of $n$ robotic manipulators $M = \{M_i, ..., M_i, ..., M_n\}$. In the organizational structure of cell control, manipulator $M_1$ represents the master element and a set of manipulators $MS \subset M$, $MS = \{M_2, ..., M_i, ..., M_n\}$ represents the slave elements. Each manipulator $M_i$ performs a set of $m$ operations $O = \{O_1, ..., O_i, ..., O_n\}$ cyclically. Each operation $O_i$ is in one cycle executed by manipulator $M_i$. The duration time of this operation is $T_{i,j} \in T = \{T_{1,1} \cdots T_{1,m} \cdots T_{n,1} \cdots T_{n,m}\}$.

It is clear that duration times $T_{i,j}$ for $x = (1, \ldots, n)$ of operation $O_i$ for every manipulator $M_i$ are different without using a synchronization algorithm. Duration time depends on a movement speed $V_{i,j} \in V = \{V_{1,1} \cdots V_{1,m} \cdots V_{n,1} \cdots V_{n,m}\}$ of manipulator $M_i$ endpoint, defined as a parameter of a movement instruction.

The goal is to modify the movement speed of manipulators $MS$ separately for every operation $O_i$ in such a way that the duration times of the same operation for every slave manipulator $MS$ will be equal or very similar.

The main requirements for the synchronization algorithm design are listed in Figure 1.
3. Design of Multi-Robotic Cell Synchronization Algorithm

The principle of the synchronization algorithm design is to dynamically adjust the speed of operation performing $V_{ij}$ on the slave side. The speed adjustment means the change of manipulator endpoint movement speed, based on the synchronization coefficient $K_{ij} \in K = \{K_{11}, ..., K_{nm}\}$. This coefficient $K_{ij}$ is evaluated for each operation $O_i$ for each slave manipulator $MS_i$ based on Equation (1).

$$K_{ij} = \frac{T_{1i}}{T_{3i}}$$

The synchronization coefficient $K_{ij}$ therefore represents the ratio of the operation duration $T_{1i}$ of the master element and the operation duration of the slave element $T_{3i}$ for $i = (2, ..., n)$.

It is necessary to emphasize that although the speed $V_{ij}$ is based on coefficient $K_{ij}$ recalculated immediately after operation $O_i$ execution, the new speed value is used only in the next cycle.

The ability of the master manipulator $M_1$ to distribute operation $O_i$ duration time $T_{1i}$ after performing for each slave manipulator $MS_i$ is the obligatory condition for feasibility of this proposal.

3.1. Basic Synchronization Algorithm

The elemental analysis results in the basic synchronization algorithm, which is the same for each slave manipulator $MS_i$; its structure is depicted in Figure 2.
Figure 2. Basic algorithm for synchronization of robotic cell operations.

As is later mentioned in Section 4.1., the main disadvantage of the proposed algorithm is low reaction speed to changes in master or slave behavior. This is contrary to the requirement R08 from Figure 1. Every change of master manipulator M1 (or slave manipulator MSi) activity is captured immediately so the synchronization coefficient Ki,j is evaluated. This leads to the adjustment of endpoint movement speed Vi,j of each slave manipulator MSi. However, the adjusted endpoint movement speed Vi,j is actually used during movement of the slave manipulator MSi in the next production cycle.

3.2. Advanced Synchronization Algorithm

If the master manipulator M1 (or slave manipulator MSi) activity change has a global character (endpoint movement speed increasing/decreasing), then it is possible to indicate this change using variable D. This variable represents the percentage proportion deviations among operation durations of master and slave manipulators, where

\[ D_{i,j} \in \{ D_{2,1}, \ldots, D_{2,m}, \ldots, D_{n,1}, \ldots, D_{n,m} \} \]

is calculated based on Equation (2).

\[ D_{i,j} = (1 - (T_{i,j}/T_{1,j})) * 100 \]  \hspace{1cm} (2)

The variable D is calculated in each step of the algorithm, but is only usable from the second production cycle. The first production cycle has diverse values of deviations D_{i,j} from first step of second cycle, D_{i,j} within defined tolerance represent synchronized state. Every change of D_{i,j} values exceeding the limit can be considered as a desynchronization indicator.

If two (or another specified number d) differences \( D_{i,j}; D_{i,j-1}; \ldots; D_{i,j-d+1} \) of consecutive operations \( O_{i+d} \ldots O_i \) are sufficiently similar based on the similarity threshold \( D_{ratio \_limit} \) (Equation (3))

\[ (|1 - (D_{i,j-d}/D_{i,j})| * 100 < D_{ratio \_limit}) \land \ldots \land (|1 - (D_{i,j-d+1}/D_{i,j-d+2})| * 100 < D_{ratio \_limit}) \]  \hspace{1cm} (3)

and this difference exceeds the given threshold of change \( D_{limit} \) (Equation (4)),

\[ |D_{i,j}| > D_{limit} \]  \hspace{1cm} (4)

then advanced synchronization coefficient \( K_i \) (Equation (5)) is applied to all other operation execution speeds Vi,j except the actual operation (which is already adjusted in this step) and previous d−1 values (which were adjusted in previous d−1 steps).

---

**Algorithm 1: Slave Speed Synchronization**

Data: \( m \) - number of operations in one cycle
Data: \( V \) - set of initial speed for each operation
Data: \( i \) - id of slave manipulator

begin
  while not end of production do
    for \( j = 1 \) to \( m \) do
      perform operation \( O_j \) by speed \( V_{i,j} \);
      \( T_{i,j} \) = operation \( O_j \) duration;
      get \( T_{1,j} \) from master;
      change speed \( V_{i,j} = V_{i,j} * T_{i,j}/T_{1,j} \);
    end
  end

---
Both algorithms were designed for (theoretically) endless repetition of production cycle. The above-mentioned previous values of the current $D_{i,j}$ in the initial steps of every cycle are based on an endless loop of variable $j$. For example, in the case of 10 operations in one production cycle, after the initialization cycle, the previous two elements $V_{2,j}$ (endpoint speed of slave manipulator $M_2$) to element $V_{2,1}$ are $V_{2,10}$ and $V_{2,9}$.

The synchronization algorithm with a feedforward reaction to the general change of the master manipulator $M_1$ (or slave manipulators $MS_i$) endpoint movement speed is identical for every slave manipulator $MS_i$ and is depicted in Figure 3.

### Algorithm 2: Slave Advanced Speed Synchronization

- **Data:**
  - $m$ - number of operations in one cycle
  - $V$ - set of default speed for each operation
  - $i$ - id of slave manipulator
  - $d$ - number of compared values
  - $D_{j,off}$ - deviation offset

- **Begin**

  - **While not end of production do**
    1. **For $j = 1 \text{ to } m$ do**
      a. perform operation $O_j$ by speed $O_{i,j}$; 
      b. $T_{i,j}$ = operation $O_j$ duration;
      c. get $T_{1,j}$ from master;
      d. change speed $V_{i,j} = V_{i,j} \times T_{i,j}/T_{1,j}$;
      e. $D_{i,j}$ = percentage proportion deviation between $T_{i,j}$ and $T_{1,j}$;
      f. If $D_{i,j} < D_{j,off}$ then
        - if $D_{i,j}$ is significantly similar to previous $d$ values of $D_{i,j}$ then
          - evaluate $K_{i}$ based on last $d$ values of $D_{i,j}$;
          - change speed $V_{i,x} = V_{i,x} \times K_{i}$
          - where $x \in (1, \ldots, m)$ - (last $d$ values of $j$)

- **End**

- **End**

**Figure 3.** Advanced algorithm for synchronization of robotic cell operations.

## 4. Validation of the Proposed Solution

The proposed solution was experimentally validated in two phases. In the first phase, the algorithms were implemented as scripts in the MATLAB software environment. To evaluate the efficiency of the algorithms, monitoring of the percentage proportion deviation $D_{i,j}$ (Equation (2)) was used. The aim of the solution was to reach a $D_{i,j}$ value as close as possible to zero. The synchronization state was indicated by a value close to zero within the specified tolerance. In contrast, a value exceeding the limit can be considered as a desynchronization indicator.

In the second phase, the designed algorithms were implemented in the native language of real robotic manipulators in a specific multi-robotic cell.

### 4.1. Simulation Validation of Solution Functionality

The functionality of the proposed solution was validated by simulations using the MATLAB software tool (R2018a, The MathWorks, Inc., Natic, MA, United States, 2018). The designed algorithms were processed by a script and the results were represented in graphical form.
Duration times $T_{ij}$ of master manipulator $M_i$ operations were simulated as generated random values of a vector in the range of 0–5000 ms. Each value $T_{ij}$ was modified during its processing with a random element from the range $(-2\%, +2\%)$, that represents a stochastic process part. Because the synchronization algorithm for all slave manipulators $MS_i$ is identical, the case with one slave manipulator $MS_i$ where $i = 2$, was validated by simulation. Operation durations $T_{ij}$ of this slave manipulator were simulated on the basis of Equation (6).

$$T_{ij} = S_{ij}/V_{ij}$$  \hspace{1cm} (6)

In Equation (6) the variable $S_{ij} \in S = \{S_{1,1}, \ldots, S_{n,m}\}$ represents the path length of operation $O_j$ performed by manipulator $MS_i$. An idealized kinematic model of a manipulator with omission of the non-linear character of robot arm movement in acceleration and deceleration was used for simulation validation purposes. Elements of the $S$ set were generated as random values of a vector in the range of 0–1000 mm in the simulation validation process. The default endpoint movement speed $V_{ij}$ of the slave manipulator was set to 200 mm/s. In processing of each operation duration $T_{ij}$, an additional modification with the stochastic part from the interval $(-2\%, +2\%)$ of $T_{ij}$ was also used.

4.1.1. Experiment 1a—Master Speed Change, Basic Algorithm

Simulation Experiment 1a includes:
- eight production cycles of production cell
- five operations in every production cycle ($m = 5$)
- change of master manipulator $M_i$ operation speed
  - in the 3rd production cycle by +20% of current speed value
  - in the 4th production cycle by +30% of current speed value
  - in the 7th production cycle by −50% of current speed value
- master manipulator $M_i$ speed change in different time of production cycle

4.1.2. Experiment 2a—Slave Speed Change, Basic Algorithm

Simulation Experiment 2a includes:
- six production cycles of production cell
- 10 operations in every production cycle ($m = 10$)
- change of slave manipulator $MS_i$ operation speed
  - in the 2nd production cycle by −30% of current speed value
  - in the 4th production cycle by +30% of current speed value
- slave manipulator $MS_i$ speed change in different time of production cycle

4.1.3. Summary of Experiment 1a and Experiment 2a

The obtained simulation results of the experiment with master speed change are depicted in Figures 4 and 5. Results of both experiments, experiment with master speed change and the experiment with slave speed change, is aggregated in Figure 6, and the results of the experiment with slave speed change in Figures 7 and 8. Figure 9 also presents results of both experiments, experiment with master speed change and the experiment with slave speed change.
Figure 4. Operation duration times of master and slave manipulators without feedforward synchronization—Experiment 1a.

![Figure 4](image)

Figure 5. Percentage proportion deviation without feedforward synchronization—Experiment 1a.

![Figure 5](image)

**Figure 6.** Brief summary of Experiment 1a results.
4.1.4. Experiment 1b—Master Speed Change, Advanced Algorithm

The case with a 10% limit for the error of the difference between operation duration performed by master and slave manipulators and a 15% significance limit for the similarity of two successive deviations $D_{ij}$ and $D_{ij+1}$ was validated via simulation.
In simulation Experiment 1b, equal parameters of the whole simulated system were used as in simulation Experiment 1a, with the addition of the feedforward synchronization feature to the synchronization algorithm.

4.1.5. Experiment 2b—Slave Speed Change, Advanced Algorithm

Identical parameters of the whole system as in simulation Experiment 2a were used in simulation Experiment 2b. The synchronization algorithm was supplemented by a feedforward synchronization function.

4.1.6. Summary of Experiment 1b and Experiment 2b

The simulation results of the experiment with a change of master manipulator speed are depicted in Figures 10 and 11. Results of the experiment with a change of slave manipulator speed are shown in Figures 13 and 14. Both experiments results, where an advanced synchronization algorithm was used, are aggregated in Figures 12 and 15. These results prove that the proposed algorithm can identify a global change of speed of the master element (Experiment 2a) and also of the slave element (Experiment 2b). Most importantly, the algorithm is able to modify the speed of the slave element (elements) feedforward unlike in the case of the standard algorithm. Based on monitoring the operation duration of master and slave manipulators (Figures 10 and 13), and according to evaluation of percentage proportion deviations (Figures 11 and 14), it is clear that the algorithm ensured the time-synchronized movement in the 3rd step of the production cycle from such a change. This resynchronized activity is indicated by a minimized absolute value of percentage proportion deviation $D_{ij}$ in the two operations after this change.

![Figure 10](image1.png)

**Figure 10.** Operation duration times of master and slave manipulators with feedforward synchronization—Experiment 1b.

![Figure 11](image2.png)

**Figure 11.** Percentage proportion deviation with feedforward synchronization—Experiment 1b.
Figure 12. Brief summary of Experiment 1b results.

Figure 13. Operation duration times of master and slave manipulators with feedforward synchronization—Experiment 2b.

Figure 14. Percentage proportion deviation with feedforward synchronization—Experiment 2b.
Figure 15. Brief summary of Experiment 2b results.

4.2. Implementation of the Proposed Solution and Discussion

The functionality of the designed algorithms was tested in the final phase on real robot manipulators in a multi-robotic cell. The algorithms were processed in the native language of each robotic manipulator controller using its built-in standard functions.

The model robotic cell, used for the purposes of this paper, contains a heterogeneous triplet of robotic manipulators as shown in Figure 16.

Figure 16. Heterogeneous multi-robotic cell.

One production cycle consists of five operations \((m = 5)\), and mainly contains instructions for circular interpolation movement in combination with linear interpolation movement. The whole production cycle is schematically shown in Figure 17.

Figure 17. Desired set of operations in one production cycle.
The organizational structure of this robotic cell control was realized as master–slave [24,25] without an external control element. The master object (master manipulator) chosen was a Mitsubishi Melfa RV-2FB-D robot with the Mitsubishi CR750-D control system [26]. Robots ABB IRB 120 and ABB IRB 140, with IRC5 Compact control systems, were the child objects-slaves [27]. A personal computer was considered an element of the slave group and provided visualization of the cell activity synchronization process. Configurations of every cell element are listed in Tables 1 and 2. The TCP/IP channel provided communication among cell elements on the basis of socket exchange [25,28].

| Table 1. Slave Configuration. |
|-------------------------------|
| **Object** | **IP Address** |
| Robot ABB IRB 120 | 192.168.1.120 |
| Controller ABB IRC5 COMPACT | 192.168.1.140 |
| Robot ABB IRB 140 | 192.168.1.140 |
| Controller ABB IRC5 COMPACT | 192.168.1.111 |

| Table 2. Master Configuration. |
|-------------------------------|
| **Parameter** | **Value** |
| Object | Robot Mitsubishi Melfa RV-2FB-D |
| Controller | Mitsubishi CR750-D |
| NETIP | 192.168.1.20 |
| COMDEV | 5, 6, 7 (OPT16, OPT17, OPT18) |
| NETMODE | 1 (SERVER) |
| NETPORT | 10006, 10007, 10008 |
| CRPCE | 2 (DATALINK) |
| PORT | COM6, COM7, COM8 |

Confirmed coordination of operations control among robotic manipulators and unconfirmed communication between master and monitoring computer were used in the cell. Confirmed coordination means that after every operation is executed by the master element, a terminating message is sent to the slave side and the master element waits for a confirmation message. The next operation can only be initiated after receiving a confirmation message from the slave object. Unconfirmed communication in this case involves simply receiving messages about the synchronization process by the slave object, without sending any confirmation message to the master object [25].

4.2.1. Master—Mitsubishi Melfa RV-2FB-D

In the case of the Mitsubishi CR750-D controller of the robotic manipulator Mitsubishi Melfa RV-2FB-D, the algorithm application was implemented in MELFA BASIC V language [29]. Communication with the slave object was realized through the initiated TCP/IP channel (Box 1).

**Box 1.** Communication initiation in MELFA BASIC V.

```
Open "COM7:": As#1 'IRB120' IP:192.168.1.120 port 10007
Open "COM8:": As#2 'IRB140' IP:192.168.1.140 port 10008
Open "COM6:": As#3 'PC' IP:192.168.1.111 port 10006
```

A duration time of each operation was evaluated and this information was, after the message was received on the completion of each operation by all slave objects, sent in a defined format to these slave objects. This message was also a confirmation message that enabled the beginning of the next operation execution. The operation duration time evaluated by the master object (Time) was measured in milliseconds. All of the obtained information was also sent to the visualization part of the application for further processing (Box 2).
Box 2. Socket communication in MELFA BASIC V.

```melfa
M_TIMER(1) = 0
[Operation instructions]
Time = M_TIMER(1)
Input #1, msg1$
Input #2, msg2$
Print #1, STR$(Operation_ID) + " + STR$(Time)
Print #3, " 1 + STR$(ID) + " + STR$(Time)
Print #3, " 2 + msg1$
Print #3, " 3 + msg2$
```

4.2.2. Slaves — ABB IRB 120/IRB 140

In the case of controllers ABB IRC5 Compact of robotic manipulators ABB IRB 120 and ABB IRB 140, the designed algorithms were implemented in RAPID language, which is the native language for ABB robot programming [30]. Communication with master object was realized through the activated TCP/IP channel (Box 3).

Box 3. Communication initiation in RAPID.

```rapid
VAR socketdev client_socket;
SocketCreate client_socket;
SocketConnect client_socket, "192.168.1.20", 10007
Time := 5;
```

Each operation time duration was measured in each production cycle and, on the basis of the decoded information from the received socket sent by the master object (custom function DecodeSocket), an actual speed for that operation was modified (custom function ChangeSpeed). This modified speed was used for the presently evaluated operation in the next production cycle. If there is a need for feedforward form of reaction on speed change then a custom function FeedforwardChange can be used (Box 4).

Box 4. Socket communication in RAPID.

```rapid
ClkStart clock1;
[Operation instructions]
ClkStop clock1;
n_time := ClkRead(clock1);
SocketSend client_socket, Str := ValToStr(op_id) + " + ValToStr(n_time*1000);
SocketReceive client_socket, Str := s_receive_string
Time := 10;
DecodeSocket;
ChangeSpeed;
[FeedforwardChange];
```

The function DecodeSocket provided information extraction from received socket (s_receive_string). This information consisted of operation identifier (op_id) and time of its duration (t_in) for the master object. Built-in functions of RAPID language for string processing StrPart, StrMatch, StrLen and StrToVal were used.

The custom function ChangeSpeed provided the modification of speed used for the current operation execution in the array of all operations speeds. These speed values were used in the next production cycle. A numerical array was also used for storing percentage proportion deviations between durations of the operation performed by master and slave objects. This array can be optionally used for feedforward synchronization. The operation duration time evaluated by a slave object (n_time) was measured in seconds (Box 5).

Box 5. The function ChangeSpeed in RAPID.

```rapid
speed[op_id] := speed[op_id] * (n_time*1000/t_in);
diff[op_id] := (1-(n_time*1000/t_in))*100;
```
The optional function `FeedforwardChange` provides an evaluation of whether the indicator of current operation change is supraliminal and, at the same time, is sufficiently similar to the indicator of the previous operation change. In the case of positive evaluation, all records in the operation speed array, except those of the current and previous operation, are modified utilizing a dynamically modified array of operation indexes (Box 6).

**Box 6.** The feedforward function in RAPID.

```
Box 6. The feedforward function in RAPID.

op_indx = [1, ..., n];
diff_ratio = Abs(1-(diff[op_indx[n]]/diff[op_indx[1]]))*100;
IF (Abs(diff[op_indx[1]])>diff_limit) AND (diff_ratio<diff_ratio_limit) THEN
  k = 1-((diff[op_indx[1]]+diff[op_indx[n]])/2)/100;
  FOR i FROM 2 TO n-1 DO
    speed[op_indx[i]] = speed[op_indx[i]]*k;
  ENDFOR
ENDIF
Rotate_op_indx_left;
```

### 4.2.3. Visualization—MATLAB Application

For the visualization of the robotic cell behavior, any development tool supporting functionalities for sockets processing can be used. In this case, the visualization part of application was implemented using the MATLAB software tool since it was already used for simulation validation of the proposed solution. The capability of MATLAB to access existing system Java classes to be used in the MATLAB workspace [31–34] was utilized in this application part. The communication with the master object was realized through the activated TCP/IP channel (Box 7).

**Box 7.** Communication initiation in MATLAB.

```
import java.net.Socket
import java.io.*
input_socket = java.net.Socket();
input_socket.connect(java.net.InetSocketAddress(server,port));
```

The received message from the master object consisted of cell object identifier (station), operation identifier (index), and operation duration (value). After decoding, the message was used as a data source for graphical representation (3D bar graph) of the production process in the robotic cell, as shown in Figures 18–20 (Box 8).

**Box 8.** Socket communication in MATLAB.

```
input = BufferedReader(InputStreamReader(input_socket.getInputStream));
message = char(input.readLine());
{graph(station)}.ZData(index) = value;
```

![Figure 18](image.png)

**Figure 18.** Visualization of operation duration in the production process.
The graph in Figure 18 was updated after each operation. Information on the operation duration of all participating executive elements was collected and subsequently distributed by the master manipulator.

4.2.4. Implementation Results

Implementation results are represented based on the visualization part of the application. The operation durations of all cell elements during the production process with the implemented basic algorithm without feedforward synchronization are depicted in Figure 19.

*Figure 19.* Implementation and visualization—synchronization without feedforward effect. (a) Initial cycle; (b) Quasi-synchronized state; (c) Nearly-synchronized state; (d) Synchronized state; (e) Master speed change cycle; (f) Resynchronized state.

The change of endpoint movement speed of the master manipulator occurred in this case during operation number two in production cycle five (Figure 19e). The production cell activities were resynchronized after the 2nd operation in the next production cycle (Figure 19f).

In the case of the advanced algorithm with feedforward synchronization implementation (Figure 20), the movement speed change of the master manipulator endpoint occurred in operation one during production cycle five (Figure 20d). The feedforward effect ensured production cell activities were resynchronized in the next two operations in the same production cycle (Figure 20e).

On the basis of the obtained results depicted in Figures 19 and 20, it is clear that the proposed and simulation-validated algorithm for the synchronization of the movement speeds of manipulators
(representing endpoint movement speeds) grouped in heterogeneous robotic work cells is functional and applicable. The deviations in achieved operation duration times of master and slave robotic manipulators in the initial production cycles (Figures 19b,c and 20b) are noticeable. These are caused by non-linear characteristics of the movement of robot arms (acceleration, deceleration), and because operations two and four are composed of several movements of different types (linear, circular). However, this disproportion is fully eliminated by dynamic speed correction of the slave manipulators during subsequent cycles.

In the case of basic non-feedforward synchronization, the system reaction to the endpoint movement speed change of the master manipulator was as expected. The process was resynchronized in the next production cycle in the same operation, where the change occurred.

The results also indicate that the reaction of slave manipulators to the global change of the master manipulator speed with the use of feedforward synchronization persists in real conditions for an additional cycle step (Figure 20c) than is presented in Section 3.1 for the simulated system with idealized conditions. The reason for this difference is that the master manipulator speed change occurred during operation execution. The percentage proportion deviation among these operation durations (between master and each slave) is thus different from the percentage proportion deviation among durations of the next operation that is completely performed at the changed speed.

Figure 20. Implementation and visualization—synchronization with feedforward effect.
(a) Initial cycle; (b) Nearly-synchronized state; (c) Synchronized state; (d) Master speed change cycle; (e) Resynchronized state in the same cycle; (f) Synchronized state.
5. Conclusions

In this study, an algorithms for the time synchronization of operations performed by a heterogeneous set of robotic manipulators grouped into a production cell was proposed, validated, and implemented. The organizational structure of this robotic cell control was realized as master–slave without an external control element. Communication in the cell was provided by a TCP/IP channel via sockets. The proposed problem solution requires minimal computational power due to an empirically oriented approach. We relied on our wide empirical knowledge and experiences in process algorithmizing, as well as on various previously implemented tasks in the field of robotics or the modeling and visualization of processes. This approach enabled the solution to be processed directly by the control unit of each participating element of the robotic cell with utilization of standard instructions in their native language. The main aim was to dynamically adapt the movement speed of slave manipulator endpoints to the master manipulator activity. Therefore, the algorithms ensure the defined milestones of the production cycle of each robotic manipulator in the cell are attained at the same time, while all operations may include various sets of different motion or manipulation instructions.

The proposed solution also includes an advanced feedforward form of operation synchronization which responds to changes in the operating cycle of the master manipulator or slave manipulators more effectively. The main difference between the two proposed algorithms is the number of unsynchronized operations performed after the change of the master or the slave behavior. In the basic algorithm case, after desynchronization, the operations of one cycle are performed unsynchronized. In contrast, the advanced algorithm ensures resynchronization after a defined number (in our case two) of asynchronously performed operations.

The application of the solution proposal is supplemented with a visualization part created using MATLAB software for technical computing. This application illustrates each intervention of the synchronization algorithms, and enables more efficient monitoring and evaluation of the multi-robotic cell activity with a focus on the synchronization process. This application part complements the validation of the functionality of the designed solution.

Finally, it can be stated that all requirements were successfully met and our solution for synchronization of the heterogeneous multi-robotic cell with emphasis on low computing power is functional and feasible.

Our goal in the future is to continue to develop this idea based on current trends in industrial automation [35]. There is a possibility in master–slave architecture to distribute more process or control information among elements, e.g., target position or movement type together with operation duration as used in our solution. Visualization, as an important aspect of the production of tomorrow, can be realized using virtual or augmented reality [35].

**Author Contributions:** Conceptualization, M.J.; formal analysis, B.J.; project administration, M.J.; software, M.J. and B.J.; supervision, M.J.; validation, M.J.; visualization, M.J.; writing—original draft, M.J. and B.J.; writing—review & editing, M.J. and B.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by VEGA agency, grant number 1/0232/18—“Using the methods of multi-objective optimization in production processes control”.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

**References**

1. Weyer, S.; Schmitt, M.; Ohmer, M.; Gorecky, D. Towards Industry 4.0—Standardization as the crucial challenge for highly modular, multi-vendor production systems. In Proceedings of the IFAC Symposium on Information Control in Manufacturing (INCOM), Ottawa, ON, Canada, 11–13 May 2015; doi:10.1016/j.ifacol.2015.06.143.
2. Halenar, I.; Juhasova, B.; Juhas, M. Proposal of communication standardization of industrial networks in Industry 4.0. In Proceedings of the IEEE 20th Jubilee International Conference on Intelligent Engineering Systems (INES), Budapest, Hungary, 30 June–2 July 2016. doi:10.1109/INES.2016.7555105.
3. Qin, Y.; Sheng, Q.Z.; Falkner, N.J.G.; Dustdar, S.; Wang, H.; Vasilakos, A.V. When things matter: A survey on data-centric internet of things. *J. Netw. Comput. Appl.* 2016, 64, 137–153.

4. Leitao, P.; Rodrigues, N.; Barbosa, J.; Turrin, C.; Pagani, A. Intelligent products: The grace experience. *Control Eng. Pract.* 2015, 42, 95–105.

5. Hermann, M.; Pentek, T.; Otto, B. Design Principles for Industrie 4.0 Scenarios. In Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS), Kauai, Hawaii, 5–8 January 2016.

6. Jasperneite, J. Was hinter Begriffen wie Industrie 4.0 steckt. *Comput. Autom.* 2014, 12, 24–28.

7. Kagermann, H.; Wahlster, W.; Helbig. J. Recommendations for implementing the strategic initiative Industrie 4.0. In *Final Report of the Industrie 4.0 Working Group*; National Academy of Science Engineering: Munich, Germany, 2013.

8. Chang, W.H.; Kim, Y.H. Robot-assisted Therapy in Stroke Rehabilitation. *J. Stroke* 2013, 15, 174–181. doi:10.5853/jos.2013.15.3.174.

9. Yoon, J.; Novandy, B.; Yoon, C.; Park, K. A 6-DOF Gait Rehabilitation Robot With Upper and Lower Limb Connections That Allows Walking Velocity Updates on Various Terrains. *IEEE/ASME Trans. Mechatron.* 2010, 15, 201–215. doi:10.1109/TMECH.2010.2040834.

10. Rubenstein, M.; Ahler, Ch.; Hoff, N.; Cabrera, A.; Nagpal, R. Kilobot: A low cost robot with scalable operations designed for collective behaviors. *Robot. Auton. Syst.* 2014, 62, 966–975. doi:10.1016/j.robot.2013.08.006.

11. Werner-Allen, G.; Tewari, G.; Patel, A.; Welsh, M.; Nagpal, R. Firefly-inspired sensor network synchronicity with realistic radio effects. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, San Diego, CA, USA, 2–4 November 2005; ACM: New York, NY, USA, doi:10.1145/1098918.1098934.

12. Baca, J.; Pagala, P.; Rossi, C.; Ferre, M. Modular robot systems towards the execution of cooperative tasks in large facilities. *Robot. Auton. Syst.* 2015, 66, 159–174. doi:10.1016/j.robot.2014.10.008.

13. Chung, S.; Slotine, J.E. Cooperative robot control and concurrent synchronization of Lagrangian systems. *IEEE Trans. Robot.* 2009, 25, 686–700. doi:10.1109/TRO.2009.214125.

14. Rodriguez-Angeles, A.; Nijmeijer, H. Mutual synchronization of robots via estimated state feedback: A cooperative approach. *IEEE Trans. Control. Syst. Technol.* 2004, 12, 542–554. doi:10.1109/TCST.2004.825065.

15. Yasuda, G. A distributed autonomous control architecture for synchronization and coordination of multiple robot systems. In Proceedings of the SICE Annual Conference 2012 (SICE), Akita, Japan, 20–23 August 2012.

16. Markus, E.D.; Yskander, H.; Agee, J.T.; Jimoh, A.A. Coordination control of robot manipulators using flat outputs. *Robot. Auton. Syst.* 2016, 83, 169–176. doi:10.1016/j.robot.2016.05.006.

17. Bouteraa, Y.; Poisson, G.; Ghommem, J.; Derbel, N. Adaptive multi-robots synchronization. In Proceedings of the IEEE International Symposium on Industrial Electronics, Bari, Italy, 4–7 July 2010. doi:10.1109/ISIE.2010.5637712.

18. Bouteraa, Y.; Ghommem, J.; Poisson, G.; Derbel, N. Distributed synchronization control to trajectory tracking of multiple robot manipulators. *J. Robot.* 2011, 9, 1–10. doi:10.1155/2011/652785.

19. Bouteraa, Y.; Ghommem, J. Synchronization control of multiple robots manipulators. In Proceedings of the 6th International Multi-Conference on Systems, Signals and Devices, Djerba, Tunisia, 23–26 March 2009. doi:10.1109/SSD.2009.4956742.

20. Duschau-Wicke, A.; von Zitzewitz, J.; Banz, R.; Riener, R. Iterative Learning Synchronization of Robotic Rehabilitation Tasks. In Proceedings of the 10th International Conference on Rehabilitation Robotics, Noordwijk, The Netherlands, 13–15 June 2007. doi:10.1109/ICORR.2007.4428447.

21. ABB MultiMove Functionality Heralds a New Era in Robot Applications. Available online: https://searchext.abb.com/library/Download.aspx?DocumentID=9AKK105152A2837&Action=Launch (accessed on 1 December 2019).

22. Gan, Y.; Dai, X.; Li, D. Off-Line Programming Techniques for Multirobot Cooperation System. *Int. J. Adv. Robot. Syst.* 2013, 10, 1–17, doi:10.5772/56506.

23. KUKA. KUKA. RoboTeam. Available online: https://www.kuka.com/en-gb/products/robotics-systems/software/hub-technologies/kuka_robotteam (accessed on 1 December 2019). Gan, Y.; Dai, X.; Li, J. Cooperative Path Planning and Constraints Analysis for Master-Slave Industrial Robots. *Int. J. Adv. Robot. Syst.* 2012, 9, 1–13, doi:10.5772/51374.

24. Juhasova, B.; Juhas, M.; Halenar, I. TCP/IP Protocol Utilisation in Process of Dynamic Control of Robotic Cell According Industry 4.0 Concept. In Proceedings of the 15th IEEE International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herfany, Slovakia, 26–28 January 2017; IEEE.
25. Mitsubishi Electric. *MELFA Industrial Robots Instruction manual (Ethernet Interface CRn-500 series)*; Mitsubishi Electric: Ratingen, Germany, 2002.
26. ABB Automation Technologies AB Robotics. *Product Specification Controller Software IRC5 Manual*; ABB: Zurich, Sweden, 2004.
27. Youm, B.; Park, J. TCP/IP protocol over ieee-1394 network for real-time control applications. In Proceedings of the 16th IFAC World Congress, Prague, Czech Republic, 3–8 July 2005; Volume 38, pp. 37–42.
28. Mitsubishi Electric. *MELFA Industrial Robots Instruction manual (Detailed Explanations of Functions and Operations)*; Mitsubishi Electric: Ratingen, Germany, 2005.
29. ABB AB Robotics Products. *Technical Reference Manual—RAPID Instructions, Functions and Data Types*; ABB: Zurich, Sweden, 2010.
30. MathWorks Matlab. Call Java Libraries. Available online: https://www.mathworks.com/help/matlab/using-java-libraries-in-matlab.html (accessed on 1 December 2019).
31. Oracle, Socket (Java Platform SE 7). Available online: https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html (accessed on 1 December 2019).
32. Oracle, BufferedReader (Java Platform SE 7). Available online: https://docs.oracle.com/javase/7/docs/api/java/io/BufferedReader.html (accessed on 1 December 2019).
33. MathWorks Matlab. Plot 3-D Bar Graph. Available online: https://www.mathworks.com/help/matlab/ref/bar3.html (accessed on 1 December 2019).
34. Pérez, L.; Rodríguez-Jiménez, S.; Rodríguez, N.; Usamentiaga, R.; García, D.F. Digital Twin and Virtual Reality Based Methodology for Multi-Robot Manufacturing Cell Commissioning. *Appl. Sci.* 2020, 10, 3633, doi:10.3390/app10103633.
35. Juhás, M.; Juhásová, B.; Halenár, I. Augmented reality in education 4.0. In Proceedings of the 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT 2018), Lviv, Ukraine, 11–14 September 2018. doi:10.1109/STC-CSIT.2018.8526676.