

Predicate-based model of problem-solving for robotic actions planning
Tsymbal, Oleksandr; Mercorelli, Paolo; Sergiyenko, Oleg

Published in:
Mathematics

DOI:
[10.3390/math9233044](https://doi.org/10.3390/math9233044)

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

[Link to publication](#)

Citation for pulished version (APA):
Tsymbal, O., Mercorelli, P., & Sergiyenko, O. (2021). Predicate-based model of problem-solving for robotic actions planning. *Mathematics*, 9(23), Article 3044. <https://doi.org/10.3390/math9233044>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.



- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Article

Predicate-Based Model of Problem-Solving for Robotic Actions Planning

Oleksandr Tsymbal ^{1,†}, Paolo Mercorelli ^{2,*,†}  and Oleg Sergiyenko ^{3,†} 

¹ Faculty of Automatics and Computerized Technologies, Kharkiv National University of Radio Electronics, Nauki Avenue 14, 61166 Kharkiv, Ukraine; oleksandr.tsymbal@nure.ua

² Institute of Product and Process Innovation, Leuphana University of Lüneburg, Universitätsallee 1, D-21335 Lüneburg, Germany

³ Faculty of Engineering, Autonomous University of Baja California, Blvd. Benito Juárez, Mexicali 21280, Mexico; srgnk@uabc.edu.mx

* Correspondence: paolo.mercorelli@leuphana.de; Tel.: +49-4131-677-1896

† The authors contributed equally to this work.

Abstract: The aim of the article is to describe a predicate-based logical model for the problem-solving of robots. The proposed article deals with analyses of trends of problem-solving robotic applications for manufacturing, especially for transportations and manipulations. Intelligent agent-based manufacturing systems with robotic agents are observed. The intelligent cores of them are considered from point of view of ability to propose the plans of problem-solving in the form of strategies. The logical model of adaptive strategies planning for the intelligent robotic system is composed in the form of predicates with a presentation of data processing on a base of set theory. The dynamic structures of workspaces, and a possible change of goals are considered as reasons for functional strategies adaptation.

Keywords: adaptation; problem-solving; robotics; predicates; manufacturing system



Citation: Tsymbal, O.; Mercorelli, P.; Sergiyenko, O. Predicate-Based Model of Problem-Solving for Robotic Actions Planning. *Mathematics* **2021**, *9*, 3044. <https://doi.org/10.3390/math9233044>

Academic Editor: António M. Lopes

Received: 2 October 2021

Accepted: 19 November 2021

Published: 26 November 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Mobile robots are remarkable cases of highly developed technology and systems. The robot community has developed a complex analysis to meet the increased demands of the control challenges pertaining to the movement of robots. The research on mobile robots has attracted many researchers in recent years. In practical application, very often, the robot operating system (ROS) is used for the communication between the robot and its control system. Different control systems are applied in Robotino. For instance, in [1,2], a linear model predictive control is used for optimal motion control, with a great advantage obtained in terms of global optimality and in computational load.

This paper is an extension of work originally presented in SPEEDAM 2020 [3], which proposed an analysis of a computer-integrated system of mobile robots application for transportation and the manipulation of goods inside manufacturing workspaces, and connected to works of P. Mercorelli and O. Sergiyenko: the consideration of a set theory-based dynamic model to describe problem-solving processes in the execution of mobile robots' paths or manipulation tasks. The description of a logical model as a key element of the decision-making system for robotic applications was connected to works of O. Tsymbal.

Modern manufacturing systems are described by an intensive application of information technologies on the base of computer networks, artificial intelligence, and digital technologies, and must correspond to requirements of mobility, of fast responses to the changing quality of products, of small sizes, specific, individual, customer, and environmental demands. In the last two decades, industrial engineers and scientists have spent a substantial amount of time and effort in researching the advanced production systems and their influence in the global market [4,5].

The parts of the mentioned concepts of manufacturing systems are already widely applied in practice, and result in an essential decrease of designer time. Others are still at the research and conceptualization stage. In any case, the core element of manufacturing systems control is in the introduction of the manufacturing intelligence. The participation of numerous units of equipment, including technological, transportation, and warehouse units, as well as humans acting in workspaces, makes it possible to describe and simulate manufacturing systems as multi-agents, as well as their origins and functioning.

Robotic systems of the transportation and manipulation types are key points of manufacturing and other automated systems. The intelligence of robots is defined by their ability to observe the workspaces, to analyze them, to make decisions concerning the problems, and to execute transportation movements/manipulations to reach the proposed goals in industrial areas. In this view, decision-making systems for robotics are important for both theory and experiments.

The aim of this article is to propose the logical approach to the decision-making of robots for manufacturing workspaces, added with adaptation solutions to respond to the dynamics of industrial areas.

2. Agent-Based Systems and Role of Robots

The open architecture of multi-agent systems (MAS) has become the basic direction of distributed artificial intelligence development. From a practical point of view, these agents are considered as self-controlled software objects with self-estimation systems and independent problem-solving tools for their own needs, and for other agents (by requests) [4].

The conceptual agent model consists of four components:

- perception, which is a source for information about the external world;
- effector, which is the agent's interface to change the world or the agent's community;
- communication system, which is the tool for information exchange between other members of the agent's community;
- aims, which are the list of goals for the agent's execution.

Any manufacturing system can be considered as a sort of MAS because they consist of various kinds of technological equipment (with/without intelligent features), possibly including manipulation and transportation robots, and human personnel, which are united by an automated control system. Modern manufacturing systems (which are mainly decentralized) are typical applications of MAS. There are many benefits of such applications for manufacturing systems: distributed processing of information, in a way which is different from united big systems; quality increase, supplied by learning and interaction with objects; support of system integration.

For the condition of manufacturing, the agent is an object with a certain intelligent level, which can be considered as physical (worker, machine) or a logical object (task, directive, order). In this way, a robot with manipulation and transportation functions looks as an ideal manufacturing agent (MA). In [5,6], we can see presentation of such an agent.

The model of MA usually includes the library of procedures (the experience of the system in the form of actions) [6–8], inference drive (problem-solver) [9–12], knowledge base (more general than procedures), and a perception processor with the possibility to communicate with the sensors of robots [11,13]. MA concepts have found their implementations for robotic group tasks [13–17], for manufacturing problems [18–22], and also for social and collaborative robotic problems [23–27]. The simulation component allows the estimation of possible results for the activity of MA [13,14]. Like any computing systems, MA includes memory and communication units [15,16]. The coordination subsystem checks the internal functions of MA, and receives the queries for coordination from other MA [3]. According to the general structure of manufacturing, [7] proposes the architecture of the manufacturing system on an agent-based concept, whereas the core of the system can be based on a number of technologies, presented in [25,26,28–37]. Decision-making systems became an object of software development design in the works [38,39].

From the point of view of multi-agent manufacturing systems functioning, the interaction of agents at every level is not the only key point. Another important thing is the implementation of agents. Other important tasks are resource planning, technological processes design, and schedules development [10,13,18]. MA can be implemented in the forms of virtual (as an automated control system function set) or physical (as industrial or transport robot) agents, able to analyze the constructive and technological specifications of manufacturing for specific workplaces of the flexible manufacturing system (FMS), to monitor the production process, to check-up the manufacturing technology acceptance, to respond to predicted or unpredicted manufacturing situations, and to supply the selected functions of operative manufacturing control [10,11]. The paper is organized in the following way: Part 3 is dedicated to the formal description of strategies planning; Part 4 describes the decision-making model, based on predicates; in Part 5, the adaptation technique of the assembling planning is shown; Part 6 describes practical implementations. The paper closes with a discussion of the results.

3. Formal Description of Strategies Planning

From an analysis of the manufacturing agent conception, we can see the need to describe the creation and activity of the problem-solving component. Such a component must include: the set of operative procedures (actions) with common knowledge support; an inference engine as the core of problem-solving; and a dynamic database with information on the surrounding workspace of the manufacturing system [8]. The actions of the manufacturing robotic agent can be described in the form of procedures, which allow the transformation of states of the robotic platform and of the external workspace (WS) [9].

The robotic agent (RA) can be described by: number of sets, X, D, S ; of platform states; of robotic system decisions; and of WS.

Correspondently, $x_i \in X, d_i \in D, s_i \in S$ can be introduced as atoms for the model, which describes robotic agents and their WS. We can also introduce standard operations $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and well-formed formulas on the base of them:

$$\neg x, x \wedge y, x \vee y, x \rightarrow y, x \leftrightarrow y$$

To describe the theory, sets X, D, S , the functions, and the predicates are introduced.

Transition of states for RA: $x_i = f(x_0, \dots, x_{i-1})$, for i -th element of set X .

Transition of states for WS: $s_i = \psi(s_0, \dots, s_{i-1})$.

To define the logics of the problem-solving system, the set of predicates (pt) is introduced:

$$pt(x_i), pt(s_i), pt(d_i), pt(x_i, s_i), pt(x_i, d_i), pt(d_i, s_i), pt(x_i, d_i, s_i)$$

These predicates define:

$pr(x_i, s_i) \subset pt$ —states of the RA in the workspace (WS),

$ps(x_i, s_i) \subset pt$ —states of the WS in the system of the RA,

$pa(x_i, s_i) \subset pt$ —actions of the RA inside the WS,

$pg(pr, ps) \subset pt$ —goals of the RA inside the WS.

Every goal of the RA is formulated as a new (or existing) state of the RA or WS:

$$pg(pr, ps) \leftarrow (pr(x_i, s_i) \vee ps(x_i, s_i))$$

The database of the RA is combined from:

$$pr(x_i), pr(x_i, s_i), ps(s_i), ps(x_i, s_i)$$

The knowledge base of the RA includes possible action $pa(x_i, s_i)$ s of the RA in the WS.

Predicate $pa(x_i, s_i)$ is a strategy, which solves goal $(pr(x_i, s_i), ps(x_i, s_i))$, if there exists such a conjunction of RA actions $pa(x_i, s_i)$ (with n —total number of actions), which supplies $pg(pr(x_i, s_i), ps(x_i, s_i))$:

$$pg(pr(x_i, s_i), ps(x_i, s_i)) \leftarrow pa^0(x_i, s_i) \wedge pa^1(x_i, s_i) \wedge \dots \wedge pa^{n-1}(x_i, s_i), \text{ or} \\ pg(pr(x_i, s_i), ps(x_i, s_i)) \leftarrow \bigwedge_{i=0}^{n-1} pa^i(x_i, s_i), \quad (1)$$

which defines, that for every robotic platform state x_i and state of workspace s_i , the goal state $pr(x_i, s_i)$ of the robotic agent or the state $ps(x_i, s_i)$ of the workspace can be reached by n -number of actions (upper indexes) and besides: $\exists f, f \in F: x_i = f_i(x_{i-1}, s_{i-1})$, $\exists \psi, \psi \in \Psi: x_i = \psi_i(x_{i-1}, s_{i-1})$, with F and Ψ —general sets of RA and WS transitions.

Therefore, $pa(x_i, s_i) = tr \| f_i + \psi_i \|$ and presents trace of norm.

The problem-solving process is the sequence of m -alternatives to reach the goals of the RA:

$$pg^0(pr, ps) \leftarrow pg^0(pr_0, ps_0, pa_0) \wedge pg^1(pr_1, ps_1, pa_1) \wedge \dots \\ \wedge pg^{n-1}(pr_{n-1}, ps_{n-1}, pa_{n-1}) = \bigwedge_{i=0}^{n-1} pg_i^0(pr_i, ps_i, pa_i) \\ pg^m(pr, ps) \leftarrow pg^m(pr_0, ps_0, pa_0) \wedge pg^m(pr_1, ps_1, pa_1) \wedge \dots \\ \wedge pg^{m-1}(pr_{m-1}, ps_{m-1}, pa_{m-1}) = \bigwedge_{i=0}^{m-1} pg_i^m(pr_i, ps_i, pa_i). \quad (2)$$

As a result, the global (final) goal is defined as follows:

$$pg^{total}(pr, ps) \leftarrow \bigvee_{j=0}^{m-1} \bigwedge_{i=0}^{n-1} pg_i^j(pr_i, ps_i, pa_i). \quad (3)$$

Every RA starts planning from the development of the initial plan of actions in the WS. It includes the next transitions:

$$pr(x_1, s_1) \leftarrow pa_0^0(pr(x_0, s_0) \vee ps(x_0, s_0)) \\ pr(x_2, s_2) \leftarrow pa_1^0(pr(x_1, s_1) \vee ps(x_1, s_1)) \\ pr(x_n = Y, s_n) \leftarrow pa_{n-1}^0(pr(x_{n-1}, s_{n-1}) \vee ps(x_{n-1}, s_{n-1})). \quad (4)$$

Such transitions can be correct for a static WS, but become practically wrong if the WS is dynamic, with an impossibility to reach the desired state:

$$pr(x_i, s_i) \neq pa_i^0(pr(x_{i-1}, s_{i-1}) \vee ps(x_{i-1}, s_{i-1})). \quad (5)$$

In this case, the strategy must be modified (signed with *):

$$pr(x_i, s_i) \leftarrow pa_i^*(pr(x_{i-1}, s_{i-1}) \vee ps(x_{i-1}, s_{i-1})), \\ pr(x_{i+1}, s_{i+1}) \leftarrow pa_{i+1}^*(pr(x_i, s_i) \vee ps(x_i, s_i)), \quad (6)$$

with the general result for m^* (modified) strategy:

$$pg^{m*}(pr, ps) \leftarrow pg^{m*}(pr_0, ps_0, pa_0) \wedge pg^{m*}(pr_1, ps_1, pa_1) \wedge \dots \\ \wedge pg^{m-1}(pr_{m-1}, ps_{m-1}, pa_{m-1}) = \bigwedge_{i=0}^{m-1} pg_i^{m*}(pr_i, ps_i, pa_i). \quad (7)$$

4. Model of Robotic System Planning on Base of Predicates

Let's define RA states as set $X = \{X^0, X^1, \dots, X^{n-1}\}$. In the process of the execution of the decision, the automated control system of the RA provides transformation of the initial state $state(x_0^0, x_1^0, x_2^0, \dots, x_{n-1}^0)$ into a specific goal state $state(x_0^m, x_1^m, x_2^m, \dots, x_{n-1}^m)$, with upper indexes for states, and lower indexes for different objects of the RA.

If the system (the RA and the WS around it) at the initial time is a set of arguments x_0^0, \dots, x_{n-1}^0 and is characterized by a state $state(x_0^0, x_1^0, x_2^0, \dots, x_{n-1}^0)$, then considering the discrete process of planning strategies, which consists of individual actions $action_0, \dots, action_k$, we can indicate that the transition from one discrete state to another is a certain relationship between objects:

$$state(x_0^1, x_1^1, \dots, x_{n-1}^1) \leftarrow action_0(state(x_0^0, x_1^0, \dots, x_{n-1}^0)). \quad (8)$$

Here, *state* is a relationship (predicate) that characterizes the state of the system as a whole, and *action (state)* means the action of transition from one state to another.

All activities related to the transitions from one state of the system to another (by executing a list of solutions) are sets of predicates:

$$\begin{aligned} \text{state}(X^1) &\leftarrow \text{action}_0(\text{state}(X^0)), \\ \text{state}(X^2) &\leftarrow \text{action}_1(\text{state}(X^1)), \\ \text{state}(X^{n-1}=Y) &\leftarrow \text{action}_{n-2}(\text{state}(X^{n-2})). \end{aligned} \quad (9)$$

Thus, the goal of a strategy planning system is to find the appropriate number of *action_i* that would satisfy the *state_i* conditions of the system. The choice of the *action_i* action to convert the *state(Xⁱ)* to state *state(Xⁱ⁺¹)* is made to ensure compatibility of the *action* arguments and the corresponding state *Xⁱ⁺¹*, which, in practice, will mean the possibility of implementing the state (local goal) *Xⁱ⁺¹* by the performing of *action*:

$$X^{i+1} \leftarrow \text{action}(X^i),$$

with *Xⁱ⁺¹* as a possible result of action *action* for the condition of state *Xⁱ*.

The predicate scheme is adaptive if the components of the antecedent (right part of the predicate expression) and the result of the scheme (consequent) change depending on changes in the state of the robotic system (RS) and the workspace (WS):

$$\text{state}(Y) \leftarrow \text{action}(S_0), \text{action}(S_1), \dots, \text{action}(S_{n-1}), \quad (10)$$

Here, *Y*—final goal of RA; *S₀, S₁, ... S_{n-1}*—the sequential states of the control system.

However, in case of the dynamics of the WS (world of robot), the state of the robot can also be transformed (such changes are not obligatory, but probable), and there are possible situations when, for some, state *state(Xⁱ⁻¹)* action *action_{i-1}(state(Xⁱ⁻¹))* will not transform the system to state *state(Xⁱ)*, thus:

$$\text{state}(X^i) \neq \text{action}_{i-1}(\text{state}(X^{i-1})). \quad (11)$$

For this case, the initial solution is in the determination of predicate *action*, which satisfies the condition. However, the actual number of real actions is limited (unlike the number of states), and the solution can be found in search of the predicate vector *action*, which satisfies the goal of system.

Therefore, if there is set *X* of the world's objects, and *X⁰* is a set of their initial states, then to supply goal state *X*, it is needed to compose a plan consisting of sequences of actions, expressed by predicate *action*, and with states of the system—by predicate *state*:

$$\begin{aligned} &\text{state}(X^0), \\ \text{state}(X^1) &\leftarrow \text{action}_0(\text{state}(X^0)), \\ \text{state}(Y) &\leftarrow \text{action}_{n-1}(\text{state}(X^{n-1})). \end{aligned} \quad (12)$$

If, at some step *i*, the state *Xⁱ* is unobtainable, that *state(Xⁱ) ≠ action_{i-1}(state(Xⁱ⁻¹))*, then the adaptive strategies planning system must generate the new order of action predicates *action*, that will satisfy the changes of the WS:

$$\text{state}(X^{i1}) \leftarrow \text{action}_{n-1}^1(\text{state}(X^{i-1})), \text{state}(Y) \leftarrow \text{action}_{n-1}^{m-1}(\text{state}(X^{n-1})). \quad (13)$$

A similar situation is when the decision-making system (DMS) has information that the goal of system is changed. It means that the goal state *state(Y)* will be changed to some *state(Yⁱ)*. Here, two variants are possible:

(a) the information on the change of the goal comes at the moment when the system is in the state i — $state(x^i)$, and it is possible to generate plan \vec{action} to transit from x^i to state y^i ;

(b) the information on the change of the goal comes at the moment when the system is in the state i — $state(x^i)$, but the generation of plan \vec{action} is possible only from state $state(x^{i-k})$, where $k \leq i$, so to generate the plan, the system must return to previous states, possibly up to state $state(x^0)$.

Again, it will need the generation of a new order of predicate actions.

According to definition, the plan of decision will consist of sets $\{action_0, action_1, \dots, action_n\}$, and the entire plan (of all the plans), developed during the problem-solving. The adapted decision plan will be the expression:

$$plan^{adaptive}(Y) \leftarrow action_0(state(X^0), action_1(state(X^1), \dots, action_{n-1}(state(X^{n-1})))) \quad (14)$$

Such a plan is in the final decision of the adaptive DMS.

The developed plan will be changed up to the execution of the last subgoal of planned order, and the plan's adaptation can be considered as its essential specification.

Also, note the need to evaluate the actions proposed by the DMS in a sequence of predicates.

As is known, a predicate has verity value. In classics, it is a mapping of n arguments to verity value. Though the fuzzy sets theory directs to the possible introduction of a fuzzy predicate term [10,11], the classic predicate has only two values: *true* and *false*. At this point, the DMS transition from one state $state(X_{i-1})$ to state $state(X_i)$ also has values of *true* or *false*, so the system transfers to a new state or does not. Probably, it is difficult to predict the state of the whole system even in ideal cases, and more to give the simple system evaluation in binary values of *true* or *false*. Rather, the verity or falsity will describe the particular system parameters. As to predicated theory, the robot's world can be described as a relation set between the world's objects, for instance, *is_a*—membership to the object's type, *is_at*—one object positioning near the other, *stands*—object's being in some state, etc.

5. Planning for Assembling System and Its Adaptation

The flexible integrated assembling system (FIAS) contains assembling automatic (soldering) machines, assembling-transport robots, storehouses, defining set $q_i \in Eq, i=0 \dots n-1$. The aim of FIAS is the execution of assembling technological process (modules) of radio-electronic devices, in particular, for the printed circuit board M .

Here $M = \langle B, Ch, T, R, C, L, \dots \rangle$, where B —the printed circuit board, Ch —microchips, T —semiconductor devices, R —resistors, C —capacitors, L —inductances.

The configuration of the device is determined by its construction design M^G , which defines the purpose location of elements at the printed circuit board. Actually, the module (board) is a rectangular matrix, filled by elements of set M (shown at Figure 1).

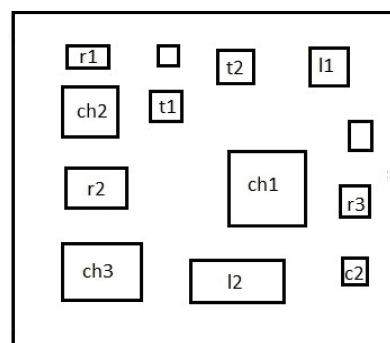


Figure 1. Workspace for manipulation task.

Initially, M_0 is zero matrix. FIAS generates decisions $d_k \in D, k=0, \dots, l-1$, which are implemented by actions (technological transitions): $a_k \in A, k=0, \dots, l-1$. Decision \vec{D} for the order of assembling operations execution is a sequence of operations $a_i \in A, i=0, \dots, l-1$, which are in the settings to board B of some elements from sets $Ch, T, R, C, L \dots$, for example:

$$\vec{D} = \{Ch_0, Ch_1, T_0, T_1, R_0, Ch_3, C_0, L_0, \dots\}$$

To reach the goal state M^G , there are transformations $M_i = f_i(Eq, D_i, M_{i-1})$:

$$M_0 \rightarrow M_1 \rightarrow M_2 \rightarrow \dots \rightarrow M^G$$

The filling of M is defined by the order of assembling operations. Such an order is set by design project M^G , technological rules Tr , and the abilities of technological equipment E . Therefore, $\vec{D} = g(M^G, Tr, E)$. The purpose of the search is to find such a sequence of transitions f_1, \dots, f_n , that provides transformation from initial state M_0 to goal M^G .

The strategies planning process is an act of constant comparison of the system's goal to the current state and current possibilities. The strategies planning process according to the discontinuity of operations also must be discrete, be correspondent to the goal's achievement, and implement the particular technological operations.

In general, the strategies planning process is a mapping of such a view:

$$F : D \times X \rightarrow Y, \quad (15)$$

that is, strategies planning process means an application of decision set $D = \{D_0, D_1, \dots, D_n\}$ to the set X_0, \dots, X_{n-1} , that formally is considered as a Cartesian product of sets $D \times X \rightarrow Y$, where Y is the set which defines ACS at the moment of goal achievement.

The transition of the robotic system (RS) from its initial state to goal is a sequence of the state's transformations, and has the view:

$$\begin{bmatrix} x_0^0 \\ x_1^0 \\ \dots \\ x_{n0}^0 \end{bmatrix} \Rightarrow \begin{bmatrix} x_0^1 \\ x_1^1 \\ \dots \\ x_{n1}^1 \end{bmatrix} \Rightarrow \begin{bmatrix} x_0^2 \\ x_1^2 \\ \dots \\ x_{n2}^2 \end{bmatrix} \Rightarrow \dots \Rightarrow \begin{bmatrix} x_0^n \\ x_1^n \\ \dots \\ x_{nn}^n \end{bmatrix} \equiv \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{nn} \end{bmatrix} \quad (16)$$

It corresponds to a real situation, when, in the process of generation and execution of the solution, there is an evolution of RS states.

However, the mentioned sequence of changes describes not only the problem-solving process, but also the dynamics of the system's changes in time. On strategies planning, the system's state changes in active mode, so, at every step, the strategies planning may change the characteristics of the RS. To consider the sequence of actions on strategies planning, there is need to define the function (vector) of problem-solving $\vec{D} = \{D_0, D_1, \dots, D_{n-1}\}$.

Therefore, the application of decision D_i for every step of ACS functioning leads to a transformation of vector column $X_i^j \rightarrow X_{i+1}^j$ of RS states.

$$\begin{bmatrix} x_0^0 \\ x_1^0 \\ \dots \\ x_{n0}^0 \end{bmatrix} * D_0 \Rightarrow \begin{bmatrix} x_0^1 \\ x_1^1 \\ \dots \\ x_{n1}^1 \end{bmatrix} * D_1 \Rightarrow \dots \Rightarrow \begin{bmatrix} x_0^{n-1} \\ x_1^{n-1} \\ \dots \\ x_{nn-1}^{n-1} \end{bmatrix} * D_{n-1} \Rightarrow \begin{bmatrix} x_0^n \\ x_1^n \\ \dots \\ x_{nn}^n \end{bmatrix} \equiv \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{nn} \end{bmatrix} \quad (17)$$

Note, that the case of adaptive strategies planning needs to take into account the effect of "third side", for example, of external world objects or rivals, which affects (positively or negatively) the strategies planning process. From one side, the effect of the external workspace may be direct, and, to take into account its existence and effect on the strategies

planning process, we need to introduce the additional factor S of external WS states, containing the objects' set $S^i = \{s_0^i, s_1^i, \dots, s_m^i\}$, with index i for discrete states of external WS:

$$\begin{bmatrix} x_0^0 \\ x_1^0 \\ \dots \\ x_{n0}^0 \end{bmatrix} * \begin{bmatrix} s_0^1 \\ s_1^1 \\ \dots \\ s_{m0}^1 \end{bmatrix} * D_0 \Rightarrow \dots \begin{bmatrix} x_0^{n-1} \\ x_1^{n-1} \\ \dots \\ x_{nn-1}^{n-1} \end{bmatrix} * \begin{bmatrix} s_0^{n-1} \\ s_1^{n-1} \\ \dots \\ s_{mm}^{n-1} \end{bmatrix} * D_{n-1} \Rightarrow \begin{bmatrix} x_0^n \\ x_1^n \\ \dots \\ x_{nn}^n \end{bmatrix} \equiv \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{nn} \end{bmatrix} \quad (18)$$

The other way is in the introduction of functional dependence for particular acts of strategies planning of workspace's states:

$$F : D(S) \times X \rightarrow Y, \quad (19)$$

and, correspondingly:

$$\begin{bmatrix} x_0^0 \\ x_1^0 \\ \dots \\ x_{n0}^0 \end{bmatrix} * D_0 \left(\begin{bmatrix} s_0^1 \\ s_1^1 \\ \dots \\ s_{m0}^1 \end{bmatrix} \right) \Rightarrow \dots \begin{bmatrix} x_0^{n-1} \\ x_1^{n-1} \\ \dots \\ x_{nn-1}^{n-1} \end{bmatrix} * D_{n-1} \left(\begin{bmatrix} s_0^{n-1} \\ s_1^{n-1} \\ \dots \\ s_{mm}^{n-1} \end{bmatrix} \right) \Rightarrow \begin{bmatrix} x_0^n \\ x_1^n \\ \dots \\ x_{nn}^n \end{bmatrix} \equiv \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_{nn} \end{bmatrix} \quad (20)$$

Therefore, D_i , as the strategies planning act, depends on the state of external workspace objects.

The difference of both the ways is not very expressive, but the explanation can be inequal. In the first case, the strategies planning system directly interacts with WS, and such an interaction leads to changes on RTS states, and, therefore, the act of strategies planning relates to the system's state, affected by the WS. For the second case, the planning act depends on the WS state, and must take into account its effect during the definition of the decision's procedures (strategies), and the decision executor transforms the state of RTS being WS-dependent.

Therefore, the ACS goal at the stage of strategies planning for the given task is in determination of ordered set (vector) $\vec{D} \subset D$, as a set of problem-solving acts, implementing the transition of the robot's ACS from initial state X_0 to the goal Y as to expression $F : D \times X \rightarrow Y$.

The importance of adaptive problem-solving arises in the case of essential changes on the conditions of decision implementation. For cases of the robot's static workspace, the goal Y , as a state of RTS, is reached by the application of the possible action's set $\vec{D} = \{D_0, D_1, \dots, D_{n-1}\}$, which transfers the systems from initial state X_0 to the goal $X_{n-1} = Y$. The set of selected actions \vec{D} is considered as a decision plan.

For a static WS, an initial decision plan $\vec{D} = \{D_0^0, D_1^0, \dots, D_{n-1}^0\}$ is developed, where the particular decision acts (strategies) are directly connected, and the application of local problem-solving act D_i to the current state X_i will transfer the system to the state X_{i+1} , which is, correspondingly, the goal for the decision act D_i . In its turn, the state X_{i+1} is initial for the new state D_{i+1} , which will transfer the system from state X_{i+1} to X_{i+2} , etc.

In the case of a WS dynamic state, the problem-solving system application of decision acts D_i can transfer the system to state X_{i+1} , which can be insufficient to implement action D_{i+1} , and will acquire the additional decision acts D'_{i+1} , D'_{i+1} , etc. Therefore, the changes of WS will lead to an indetermination of possible problem-solving tools.

6. Practical Implementations

Proposed mathematical models are used as a basis for development of decision-making systems for mobile and manipulation robots.

Initially implemented with Prolog language, the decision-making system (DMS), based on principles of proposed models, was added by graphical simulation procedures, and translated to C++/MFC. It had view of a classic STRIPS-like system, which describes the robotic workspace (such as for a transportation robot), consisting of rooms with boxes inside, and connected by opening/closing doors. The user's interface allows the selection

of the agent of action (robot or boxes), the type of action (go/push (for boxes); open/close (for doors)), and, in this way, to set the task (goal) for the system. According to the formulated goal (if formulated correctly), the DMS finds the action scheme which satisfies the goal, automatically sets subgoals, and reaches them. Structurally, the DMS consists of a decision-maker procedure and action scheme procedures descriptions.

The decision-making procedure includes the stages:

- selection of the action scheme, compatible to the goal (subgoal);
- satisfaction of pre-conditions for the selected action scheme, possibly with recursive execution of subgoals;
- execution of post-conditions (for lists of deleted and added facts).

Correspondingly, every action scheme includes the parts of the action result, and of precondition and post-condition facts lists. Implementation of the DMS is shown in Figure 2.

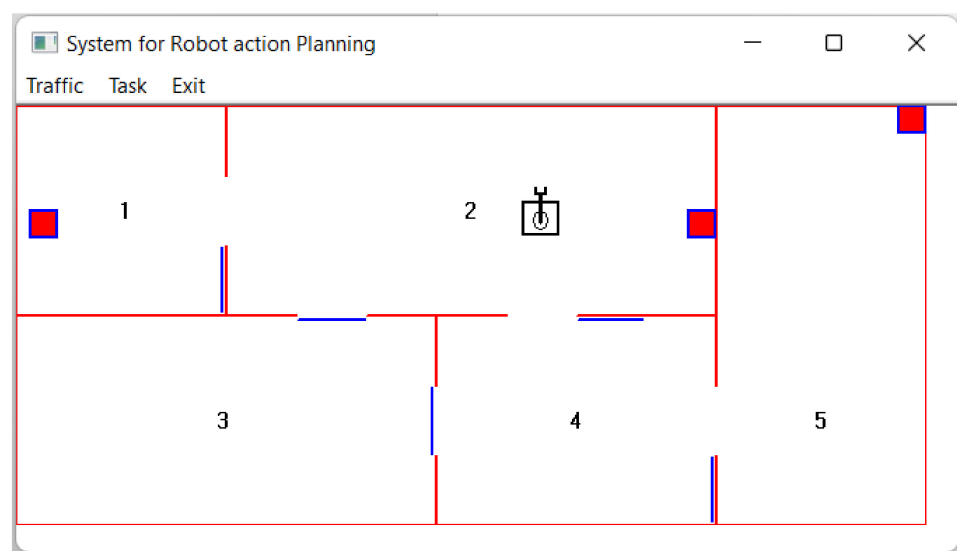


Figure 2. Decision-making system implementation.

This DMS deals with elements of sets, mentioned in Sections 3 and 4 of this article. Here, X includes the positions and states of the robot (such as $is_at(robot, door45, now)$; or $is_with(robot, box1, now)$), S contains states of WS (such as $stands(door12, closed, now)$; or $is_in(box1, room1, now)$), whereas D consists of a list of actions: “go_to”; “open_door”; “take_box”; “push_box”; etc.

Later implementations of the DMS, based on the predicate’s models, were included to projects with the NXT MindStorms mobile robot with a visual-guided control system, and, currently, with the Festo Robotino (version 2). These projects consider the predicate-based DMS as the upper level of decision making to plan actions within distributed workspaces, or for manipulations (if robots are equipped with manipulators), whereas lower decision-making levels are mostly based on path-optimized methods to plan movements within the discrete workspace of a particular work cell (e.g., as shown in Figure 3 for the Festo Robotino Robot).

The Festo Robotino is controlled by a two-level RPC (remote procedure call) protocol, which allows a program running on one computer to access the functions (procedures) of a program running on another computer.

To test the developed decision-making models, there is created a program to control the movements of the Festo Robotino. In this program, the task of the robot is set in the form of the final (target) robot position in 2D-space, for example, $P(x, y)$. The possibility to reach the target position is analyzed, taking into account the information about the occupied and free areas of the robotic workspace provided by the visual monitoring system, implemented

using a set of web-cameras. The check of the robot's position is provided by global cameras mounted above the workspace, and a local camera on board with the robot.



Figure 3. Example of discrete workspace for the Festo Robotino.

In a command mode, the developed program provides a number of modes, including determination of the robot's current position, a stop command, a setting of a target point, a setting of vector of long-distance movement, and opening and closing the robot's gripper.

In strategy planning mode, the user can set tasks for the robot. In particular, the next options can be specified: action agent–robot; current action, for example, “take object” or “take”; subject (or subjects) of action, such as “instrument”.

DMS in this program is provided in the manner described in Section 3 of the article. The generation of the robot's route is supported by analyses of the objects and obstacles allocation in the robotic workspace with computer vision methods. After the decision is made, it is implemented by the control system of the Robotino robot.

The developed software allows the consideration of the robot as a multi-agent system, which includes: motion agent (robot motion control subsystem); vision agent (computer vision system robot); sensory agent (implemented by the sensor system of the mobile robot and information processing means); agent navigation (associated with the robot's motion control system and computer vision system); strategy planning agent.

7. Discussion

Therefore, the proposed article describes the formalization of strategies planning for the problem-solving component of a robotic control system. Such formalization includes the setting of a theory basis, the definition of predicates to describe the workspace, and the states, actions, and goals for the robotic system. Here, goals are reached as conjunctions of actions, with recursive definitions as sub-goals. The selection of the sequence of actions to reach the desired goal is defined as strategy, selected from several alternative plans. Such a definition makes it possible to implement a decision-making system in the form of an automatic problem-solver, efficient for closed workspaces or areas with advanced monitoring. This set of strategies can be considered as a knowledge base for particular robots or groups of robots, acting as manufacturing agents.

From the formal description of problem solving, the article shifts to the implementation of Prolog-similar predicate notation (Section 4), while also supporting the idea of a decision-making engine based on logics. The dynamics of the robotic workspace can lead to the re-formalization of strategies, which is also discussed as a requirement for plan modification. The proposed formalization of decision-making can be applied for trans-

portation tasks of mobile robots, and for manipulations (Section 5 contains an example of the assembling system).

8. Conclusions

Action planning for robotic systems can be introduced into transportation and manipulation problems, and applied for the solution of manufacturing tasks [18–21], and also for social and collaboration tasks of robots [23,25,26,30]. For transportation problems in static workspaces, a solution can be effectively reached by computation methods. But, if the system has knowledge how to solve the problem, AI-methods become more effective, especially with the application of neural networks and genetic algorithms. More complex transportation problems appear for 3D systems of the warehouse type with vertical shelves, or for container terminals for seaports. However, with a different scale, the latter tasks look very similar to the 3D problems of robotic manipulators, especially if they make actions in a narrow space of obstacles [18,39]. Here, the knowledge of the system becomes more critical, and helps to solve the problems.

However, for conditions of dynamic space, the situation is more complicated. The motion of other objects (equipment, robots, vehicles, humans) can easily stop the execution of the best calculated plan (or strategy—in the terms of the proposed paper). In this case, the system must be adaptive to solve its problems, or come back to previous steps of the solution, to previous key points, or even to starting points [11,12]. In this case, robots, like humans, must be more logically intelligent (with experience in a greater number of operator's schemes), while combining computations, AI-methods, and group methods [13–16]. Possibly, in this way, we can find the combined solution for robots which aim to be intelligent.

In future works, the authors will try to expand the proposed model with the consideration of uncertain factors of robotic workspaces, with more detailed descriptions of robotic manipulations, and by modelling the group work of robots.

Author Contributions: Conceptualization, O.T., P.M. and O.S.; methodology, O.T., P.M. and O.S.; software, O.T.; validation, O.T., O.S. and P.M.; formal analysis, O.T. and O.S.; investigation, O.S.; resources, P.M.; data curation, O.T. and O.S.; writing—original draft preparation, O.T.; writing—review and editing, P.M. and O.S.; visualization, O.T.; supervision, P.M.; project administration, P.M.; funding acquisition, P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mercorelli, P.; Voss, T.; Strassberger, D.; Sergiyenko, O.; Lindner, L. A model predictive control in robotino and its implementation using ROS system. In Proceedings of the International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC), Toulouse, France, 2–4 November 2016; pp. 1–6. [\[CrossRef\]](#)
2. Mercorelli, P.; Voss, T.; Strassberger, D.; Sergiyenko, O.; Lindner, L. Optimal trajectory generation using MPC in robotino and its implementation with ROS system. In Proceedings of the 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), Edinburgh, Scotland, 19–21 June 2017; pp. 1642–1647. [\[CrossRef\]](#)
3. Tsymbal, O.; Bronnikov, A.; Mercorelli, P. Decision-making models for Robotic Warehouse. In Proceedings of the 2020 International Symposium on Power Electronics, Electrical Drives, Automation and Motion, Sorrento, Italy, 24–26 June 2020; pp. 546–551.
4. Mikhailov, E.; Remenyuk, B. Optimize the placement warehouse transport system. *J. Electrotech. Comput. Syst.* **2015**, *18*, 60–64.
5. Kerak, P. Novel trends in the intelligent manufacturing systems. In Proceedings of the 8th International Baltic Conference Industrial Engineering, Tallinn, Estonia, 19–21 April 2012.
6. Red'ko, V. Interaction between learning and evolution in population of autonomous agents. *Computing* **2013**, *12*, 42–47. [\[CrossRef\]](#)
7. Eiter, T.; Wolfgang, F.; Leone, N.; Pfeifer, G. A Logic Programming Approach to Knowledge-State Planning: Semantics and Complexity. *ACM Trans. Comput. Log.* **2004**, *5*, 206–263. [\[CrossRef\]](#)

8. Tsymbal, A.; Bronnikov, A.; Yerokhin, A. Adaptive Decision-making for Robotic adaptive tasks. In Proceedings of the IEEE 8th International Conference on Advanced Optoelectronics and Lasers (CAOL), Sozopol, Bulgaria, 6–8 September 2019; pp. 594–597. [\[CrossRef\]](#)
9. Nevliudov, I.; Tsymbal, O.; Bronnikov, A. Intelligent means in the system of managing a manufacturing agent. *Innov. Technol. Sci. Solut. Ind.* **2018**, *1*, 33–47. [\[CrossRef\]](#)
10. Nevliudov, O.; Tsymbal, A.; Andrushevich, V.; Gopejenko. *Intelligent Decision-Making Support for Flexible Integrated Manufacturing*; ISMA: Riga, Latvia, 2020; 390p.
11. Bronnikov, A.; Nevliudov, I.; Tsymbal, O. Flexible manufacturing tendencies and improvements with visual sensoring. *Eskiseh. Tech. Univ. J. Sci. Technol. Appl. Sci. Eng.* **2019**, *20*, 77–83.
12. Vacic, V.; Sobh, T. Vehicle routing problem with time windows. *Computing* **2004**, *3*, 72–80.
13. Sergiyenko, O.; Flores-Fuentes, W.; Mercorelli, P. (Eds.) *Machine Vision and Navigation*; Springer: Berlin/Heidelberg, Germany, 2019; 851p.
14. Sergiyenko, O.Y.; Ivanov, M.V.; Tyrsa, V.V.; Kartashov, V.M.; Rivas-López, M.; Hernández-Balbuena, D.; Flores-Fuentes, W.; Rodríguez-Quinonez, J.C.; Hipólito, J.I.N.; Hernandez, W.; et al. Data transferring model determination in robotic group. *Robot. Auton. Syst.* **2016**, *83*, 251–260. [\[CrossRef\]](#)
15. Sergiyenko, O.Y.; Tyrsa, V.V. 3D Optical Machine Vision Sensors With Intelligent Data Management for Robotic Swarm Navigation Improvement. *IEEE Sens. J.* **2021**, *21*, 11262–11274. [\[CrossRef\]](#)
16. Ivanov, M.; Sergiyenko, O.; Tyrsa, V.; Lindner, L.; Flores-Fuentes, W.; Rodríguez-Quinonez, J.C.; Hernandez, W.; Mercorelli, P. Influence of data clouds fusion from 3D real-time vision system on robotic group dead reckoning in unknown terrain. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 368–385. [\[CrossRef\]](#)
17. Palmieri, N.; Yang, X.; De Rango, F.; Santamaria, A.F. Self-adaptive decision-making mechanisms to balance the execution of multiple tasks for a multi-robots team. *Neurocomputing* **2018**, *306*, 17–36. [\[CrossRef\]](#)
18. Kangru, T.; Riives, J.; Otto, T.K.; Pohlak, M.; Mahmood, K. Intelligent Decision Making Approach for Performance Evaluation of a Robot-Based Manufacturing Cell. In Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition, Pittsburgh, PA, USA, 9–15 November 2018; Volume 2: Advanced Manufacturing.
19. Hubmann, C.; Becker, M.; Althoff, D.; Lenz, D.; Stiller, C. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1671–1678. [\[CrossRef\]](#)
20. Guérin, J.; Thiery, S.; Nyir, E.; Gibaru, O. Unsupervised Robotic Sorting: Towards Autonomous Decision Making Robots. *Int. J. Artif. Intell. Appl.* **2018**, *9*, 81–98. [\[CrossRef\]](#)
21. Schwarting, W.; Alonso-Mora, J.; Rus, D. Planning and Decision-Making for Autonomous Vehicles. *Ann. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 187–210. [\[CrossRef\]](#)
22. Popescu, G.; Valášková, K.; Majerova, J. Real-Time Sensor Networks, Advanced Robotics, and Product Decision-Making Information Systems in Data-driven Sustainable Smart Manufacturing. *Econ. Manag. Financ. Mark.* **2020**, *15*, 29–38.
23. Pérula-Martínez, R.; Castro-González, A.; Malfaz, M.; Alonso-Martín, F.; Salichs, M. Bioinspired decision-making for a socially interactive robot. *Cogn. Syst. Res.* **2019**, *54*, 287–301. [\[CrossRef\]](#)
24. Wang, B.; Rau, P.L.P. Influence of Embodiment and Substrate of Social Robots on Users' Decision-Making and Attitude. *Int. J. Soc. Robot.* **2019**, *11*, 411–421. [\[CrossRef\]](#)
25. Chen, M.; Nikolaidis, S.; Soh, H.; Hsu, D.; Srinivasa, S. Trust-Aware Decision Making for Human-Robot Collaboration: Model Learning and Planning. *J. Hum.-Robot Interact.* **2020**, *9*, 1–23. [\[CrossRef\]](#)
26. Ebert, J.; Gauci, M.; Nagpal, R. Multi-Feature Collective Decision Making in Robot Swarms. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18), Stockholm, Sweden, 10–15 July 2018; pp. 1711–1719.
27. WanLee, S.; Seymour, B. Decision-making in brains and robots—The case for an interdisciplinary approach. *Curr. Opin. Behav. Sci.* **2019**, *26*, 137–145.
28. Patle, B.K.; Pandey, A.; Jagadeesh, A.; Parhi, D.R. Path planning in uncertain environment by using firefly algorithm. *Def. Technol.* **2018**, *14*, 691–701. [\[CrossRef\]](#)
29. Joo, S.-H.; Manzoor, S.; Rocha, Y.G.; Bae, S.-H.; Lee, K.-H.; Kuc, T.-Y.; Kim, M. Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling. *Appl. Sci.* **2020**, *10*, 3219. [\[CrossRef\]](#)
30. Unhelkar, V.; Li, S.; Shah, J. Decision-Making for Bidirectional Communication in Sequential Human-Robot Collaborative Tasks. In Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, Cambridge, UK, 23–26 March 2020; pp. 329–341.
31. Shi, H.; Lin, Z.; Zhang, S.; Li, H.; Hwang, F. An adaptive decision-making method with fuzzy Bayesian reinforcement learning for robot soccer. *Inf. Sci.* **2018**, *436–437*, 268–281. [\[CrossRef\]](#)
32. Sun, C.; Kingry, N.; Dai, R. A Unified Formulation and Nonconvex Optimization Method for Mixed-Type Decision-Making of Robotic Systems. *IEEE Trans. Robot.* **2021**, *37*, 831–846. [\[CrossRef\]](#)
33. Zafar, M.N.; Mohanta, J.C. Methodology for Path Planning and Optimization of Mobile Robots: A Review. *Procedia Comput. Sci.* **2018**, *133*, 141–152. [\[CrossRef\]](#)

-
34. Zagradjanin, N.; Pamucar, D.; Jovanovic, K. Cloud-Based Multi-Robot Path Planning in Complex and Crowded Environment with Multi-Criteria Decision Making Using Full Consistency Method. *Symmetry* **2019**, *11*, 1241. [\[CrossRef\]](#)
 35. Wojtak, W.; Ferreira, F.; Vicente, P.; Louro, L.; Bicho, E.; Erlhagen, W. A neural integrator model for planning and value-based decision making of a robotics assistant. *Neural Comput. Appl.* **2021**, *33*, 3737–3756. [\[CrossRef\]](#)
 36. Li, L.; Ota, K.; Dong, M. Humanlike Driving: Empirical Decision-Making System for Autonomous Vehicles. *IEEE Trans. Veh. Technol.* **2018**, *67*, 6814–6823. [\[CrossRef\]](#)
 37. Upadhyay, J.; Rawat, A.; Deb, D.; Muresan, V.; Unguresan, M.-L. An RSSI-Based Localization, Path Planning and Computer Vision-Based Decision Making Robotic System. *Electronics* **2020**, *9*, 1326. [\[CrossRef\]](#)
 38. Tsarouchi, P.; Spiliotopoulos, J.; Michalos, G.; Koukas, S.; Athanasatos, A.; Makris, S.; Chryssolouris, G. A Decision Making Framework for Human Robot Collaborative Workplace Generation. *Procedia CIRP* **2016**, *44*, 228–232. [\[CrossRef\]](#)
 39. Agostini, A.; Torras, C.; Wörgötter, F. Efficient interactive decision-making framework for robotic applications. *Artif. Intell.* **2017**, *247*, 187–212. [\[CrossRef\]](#)