applied sciences

MDPI

Article

Formation Control of Multiple Autonomous Mobile Robots Using Turkish Natural Language Processing

Kadir Aram [1], Gokhan Erdemir [2,*] and Burhanettin Can [1]

[1] Computer Engineering Department, Fatih Sultan Mehmet Vakıf University, 34445 Istanbul, Türkiye; karam@fsm.edu.tr (K.A.); bcan@fsm.edu.tr (B.C.)
[2] Department of Engineering Management and Technology, University of Tennessee at Chattanooga, Chattanooga, TN 37405, USA
* Correspondence: gokhan-erdemir@utc.edu

Abstract: People use natural language to express their thoughts and wishes. As robots reside in various human environments, such as homes, offices, and hospitals, the need for human–robot communication is increasing. One of the best ways to achieve this communication is the use of natural languages. Natural language processing (NLP) is the most important approach enabling robots to understand natural languages and improve human–robot interaction. Also, due to this need, the amount of research on NLP has increased considerably in recent years. In this study, commands were given to a multiple-mobile-robot system using the Turkish natural language, and the robots were required to fulfill these orders. Turkish is classified as an agglutinative language. In agglutinative languages, words combine different morphemes, each carrying a specific meaning, to create complex words. Turkish exhibits this characteristic by adding various suffixes to a root or base form to convey grammatical relationships, tense, aspect, mood, and other semantic nuances. Since the Turkish language has an agglutinative structure, it is very difficult to decode its sentence structure in a way that robots can understand. Parsing of a given command, path planning, path tracking, and formation control were carried out. In the path-planning phase, the A* algorithm was used to find the optimal path, and a PID controller was used to follow the generated path with minimum error. A leader–follower approach was used to control multiple robots. A platoon formation was chosen as the multi-robot formation. The proposed method was validated on a known map containing obstacles, demonstrating the system's ability to navigate the robots to the desired locations while maintaining the specified formation. This study used Turtlebot3 robots within the Gazebo simulation environment, providing a controlled and replicable setting for comprehensive experimentation. The results affirm the feasibility and effectiveness of employing NLP techniques for the formation control of multiple mobile robots, offering a robust and effective method for further research and development on human–robot interaction.

Keywords: Turkish; natural language processing; robot path planning; platoon; path tracking; robot formation; leader–follower

check for updates

Citation: Aram, K.; Erdemir, G.; Can, B. Formation Control of Multiple Autonomous Mobile Robots Using Turkish Natural Language Processing. *Appl. Sci.* **2024**, *14*, 3722. https://doi.org/10.3390/app14093722

Academic Editors: Douglas O'Shaughnessy and Krzysztof Koszela

Received: 14 February 2024
Revised: 11 April 2024
Accepted: 16 April 2024
Published: 27 April 2024

1. Introduction

Robots continue to be used in every aspect of life, including housework, rehabilitation, medicine, educational technologies, office work, the military, various service areas, and production lines, according to technological development. The proliferation of these areas enables robots and humans to exist intertwined [1]. As areas of robot usage grow, the need for their communication with humans increases. As this need increases, communication becomes more complex. For this reason, the number of studies in the field of human–robot interaction has increased. Methods are being developed to provide interaction between humans and robots. Examples of these methods include human hand and arm behaviors [2] and interaction with natural languages [3]. The increasing use of natural language expressions has led to increased research on natural language processing (NLP). The primary function of natural language processing is to design and implement computer systems for analyzing, interpreting, and understanding a natural language [4]. With studies on NLP, various libraries that can be used with many languages have been developed, such as NLTK [5], CoreNLP [6], TextBlob [7], Gensim [8], and SpaCy [9]. These libraries provide a programming interface, especially for researchers. Studies on natural language processing in Turkish are constantly increasing. One of the first studies was Zemberek [10]. Zemberek is an open-source application that contains many units [10]. Zemberek is a natural language processing (NLP) tool developed specifically for Turkish [10]. It provides various functionalities such as morphological analysis, disambiguation, stemming, and spell-checking [10,11]. Zemberek is an open-source project that aims to assist developers and researchers in processing Turkish text effectively [10,11] via services such as normalization, morphological analysis, entity name recognition, sentence break detection, correction, disambiguation, and language recognition [10,11]. The ITU Natural Language Processing software chain provides many features, such as a normalization module, sentence separator, vowel restoration, morphological analysis, syllable corrector, entity name identification, and dependency analysis by providing a web interface and a programming interface [12]. Another important function of Zemberek is stemming, which involves reducing words to their base or root form. This is particularly useful in tasks such as information retrieval and text mining in which variations of words need to be treated as the same entity. Furthermore, Zemberek includes a spell-checking module that can identify and suggest corrections for misspelled words in Turkish text. This feature enhances the overall quality of text processing and improves the accuracy of NLP applications.

NLP provides control of communication between a user and a system. An example of this use is the control of an autonomous mobile robot [13]. For robots to execute given commands, the commands must undergo some processing. In the literature, many studies on the realization of commands exist, especially in English. The main aim is to understand commands with different structures. For this purpose, approaches such as the reward-defined method [14], using image and depth information with natural language processing [15], understanding abstract expressions [16], responding to changes in an environment [17], and asking questions if the command is not understood [18] have been presented. Another example is the Controlled Robot Language (CRL) design, which was presented for efficient human–robot collaboration and team communication [19]. In [19], the CRL framework maps linguistic commands to logical semantic expressions designed explicitly for automated robot planning using the IEEE CORA ontology for grammar, and a CRL checker detects linguistic patterns for sentences outside the grammar coverage.

The main problems in mobile robotic studies are mapping, location estimation, exploration, and navigation. Path planning is one of the most important problems in robot systems. Robots need to choose the optimal path in an environment containing obstacles. Algorithms such as A*, D*, and RRT are used for global path planning [20]. Another important issue in this field is the path-following problem, which concerns the ability to drive a mobile robot as autonomously as possible on a predefined reference path [21]. The path-following problem can be solved by different control methods, such as a PID controller [22,23], model predictive control [24–26], and the fuzzy logic approach [27]. In multi-robot systems, these processes become even more complex. It is naturally necessary for each robot in a multi-robot system to navigate independently while cooperating with other robots for efficient performance, which depends on each robot's conditions, such as its structure, location, and speed [28]. One frequently preferred trajectory-tracking approach in multi-robot systems is the leader–follower approach [29]. In this approach, a leader robot follows a specified path, and a follower robot follows the leader robot at a specified distance and angle [29].

In this study, the formation control of autonomous mobile robots is achieved utilizing the Turkish natural language. Turkish commands received are subjected to parsing employing Turkish natural language processing techniques. Subsequently, the mobile robots execute the assigned tasks based on these parsed commands. Various scenarios associated with these tasks are formulated for both obstacle-laden and obstacle-free environments. It is very difficult to parse Turkish because of its agglutinative structure, not only for robots but also for people learning the language. To give an example, in Turkish, the single word "Görüşmeyeceklermiş" translates into English as "I heard that they are not going to be able to see each other". This example shows how difficult it is to analyze language. The Turkish language has an agglutinative structure, and it is very difficult to decode its sentence structure in a way that robots can understand. Parsing of the given command, path planning, path tracking, and formation control were carried out. According to our review of the literature, this study is the first study in the literature to use Turkish NLP for multi-robot formation control. The Zemberek library used in this study is used for the first time in robot control. This study is a pioneering study in the application of Turkish natural language processing within the domain of multiple-mobile-robot systems. This study tested natural commands from a user using Turkish NLP on a multi-mobile-robot system. After parsing the commands received from the user, the robots were enabled to move from a starting point to a target point while maintaining their formation. The A* algorithm was used for path planning, and a PID controller was used for path tracking. The leader–follower approach and platoon-type formations were used. In this study, ROS (Robotic Operating System), Noetic version, and Gazebo were used to create a simulation environment, and the Turtlebot3 Burger model was chosen as the mobile robot.

2. Natural Language Processing (NLP)

Natural language is an important tool for people to communicate with each other. Natural language is used in many methods that people use in daily life during communication. Although at first glance, it seems simple to communicate using natural languages in daily life, it is necessary to go through certain stages to learn these languages. Approaching the problem from this point of view, the comprehension of natural languages in a computer environment involves many problems. Natural language processing studies that use natural languages for different functions in the computer environment have emerged.

There are different definitions of NLP. For example, NLP is a subfield of computer science that uses computational techniques to learn, understand, and produce human language content [30]. NLP involves designing and implementing models, systems, and algorithms to solve practical problems in understanding human languages [31]. NLP is an interdisciplinary phenomenon using concepts and techniques from many fields. It has found its place in many different fields. In particular, it is necessary to conduct research and collaborate, centering this focus on NPL research on the following topics: phonetics, morphology, syntax, and semantics [32]. Some of the applications of NLP are given [32,33]. They can be classified as correcting spelling errors, translating text and speech, summarizing text, answering questions, executing commands, and grading exams [33]. NLP stages are considered under five main sections. These are normalization, morphological analysis, syntactical parsing-dependency parsing, semantics, and discourse [32,33].

Turkish exhibits distinctive characteristics compared to languages like English, German, and Spanish, which have been extensively examined in the scholarly literature. Particularly notable is its agglutinative nature, which enables the formation of numerous word forms through suffixation. Consequently, spell-checking in Turkish poses significant challenges [34,35]. Despite relatively lagging progress in active Natural Language Processing (NLP) endeavors focused on Turkish, substantial advancements have been made in recent years. Mukayese stands out as a noteworthy effort, providing a comprehensive benchmarking study for various NLP tasks pertaining to the Turkish language [36]. In [37], an elaborate finite automaton model encompassing Turkish grammar rules alongside developing tools for stemming, morphological labeling, and verb negation was devised in Turkish. TurkishDelightNLP represents a neural NLP framework tailored specifically for Turkish, offering a wide array of functionalities, including morphological tagging, dependency parsing, semantic tagging, named entity recognition, and part-of-speech tagging [38]. In a separate investigation, a machine learning-driven hybrid sentiment analysis model was introduced to enhance sentiment analysis performance within Turkish question–answer systems [39].

3. Autonomous Mobile Robots

Autonomous ground robots are robots that can move autonomously from one place to another. These robots have the ability to move freely within a predefined workspace to fulfill given tasks and achieve desired goals [40]. Navigation is a challenging problem for autonomous mobile robots. The robot goes through different phases to achieve successful navigation, such as sensing, localization, cognition, and motion control. The robot interprets information from its sensors in the sensing phase and extracts meaningful data [41,42]. In the localization phase, the robot estimates its position in its environment using information from external sensors. In the cognition phase, the robot plans the steps needed to reach the goal. The motion control phase enables the robot to achieve the desired trajectory by changing its motor outputs [41].

3.1. Path Planning

Path planning can be defined as the determination of the path that mobile robots will follow to reach a target point from a starting point. This process involves the calculation of alternative paths to reach the target and determining which points to pass through one by one. The distance taken must be short, the time to reach the target point must be optimized, and no factors must make the path difficult [43]. Different approaches based on different distance definitions are used to calculate the path between the robot and the target [40,41]. A hybrid system for cooperative driving systems, combining discrete events and continuous vehicle dynamics, was presented [44]. A control algorithm combined the artificial potential field (APF) approach with model predictive control (MPC) to achieve synchronous path planning and motion control [44].

In this study, the A* algorithm is used for path planning. The A* algorithm is a heuristic method that computes the path between projected starting and destination positions at the lowest cost [45]. It has been widely used in the field of mobile robots due to its least-cost trajectory design [45]. A* is an algorithm designed on a weighted graph. It is considered a search algorithm. It is a search algorithm that aims to find a path from a given initial node of a graph to the goal node at the lowest cost (taking the shortest path and the shortest amount of time). To find the path with the lowest cost, it builds a path tree starting from the initial node. These paths are advanced by extending one edge of the graph at a time until the algorithm's performance criteria are met [45]. The algorithm uses heuristics to optimize its search. A function can be defined as one that can estimate, to some extent, the cost from one node to another node. A disadvantage is that all nodes created are kept in memory, which increases the computational load. The cost calculation of each node visited to reach the destination is expressed as shown in Equation (1) [45].

$$f(n) = g(n) + h(n) \tag{1}$$

where n is the current node, $g(n)$ is the cost from the current node to the initial node-equation, $h(n)$ is a heuristic function, and $f(n)$ is the cost from the current node to the destination node. The heuristic function specifies the heuristic cost from a node to a given destination.

3.2. Autonomous Mobile Robot Control

Various control methods are used to make autonomous mobile robots (AMRs) reach a specified target or to move in a desired direction. These methods vary according to the type of robot used and the environment. The robot's path planning process (navigation) is challenging and requires many stages. For successful navigation, the robot must complete sensing, localization, cognition, and motion control stages. In the sensing phase, the robot interprets the data received from its sensors and extracts meaningful data. In the localization phase, it estimates its position in the environment used. In the cognition phase, the steps necessary for the robot to reach the target point are planned. In the motion control phase, the robot is moved in the desired trajectory by changing its motor outputs [40–42]. The geometric approach for the AMR's kinematics and the target point information are shown in Figure 1 [46,47].

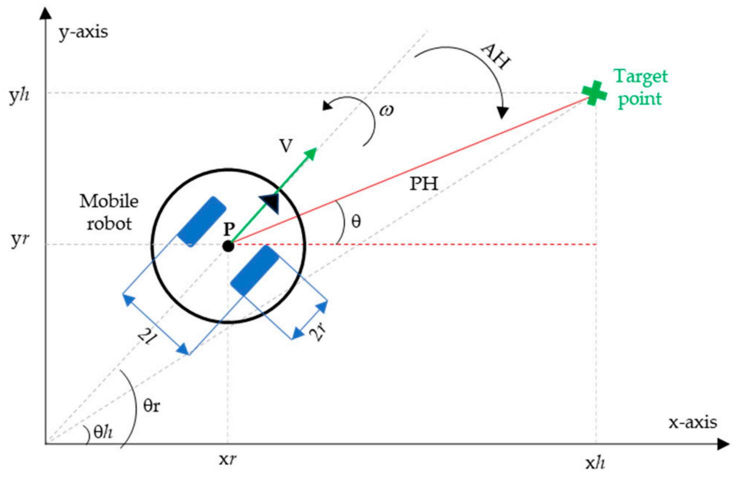


Figure 1. Geometric approximation of kinematics: AMR kinematics and target point.

A point on the robot's chassis is determined as a reference point to specify the position of the robot. This point is denoted by P. (X_R, Y_R) is the robot coordinate system placed on the robot at the point P; ($x_ı$, $y_ı$) is the base/universal/reference coordinate system. θ_r is the angle between the x_r-axis and the $x_ı$ coordinate axes due to rotation around the z-axis. The position of the robot in the (X_I, Y_I) base coordinate system can be represented as a three-element vector as follows:

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2}$$

where ξ_I is the position vector, x is the x component of the position, y is the y component of the position, and θ is the angular position. The speed of the wheel is given in Equation (3) [47]:

$$\dot{x}_R = \frac{r(\dot{\varphi}_1 + \dot{\varphi}_2)}{2} \tag{3}$$

where r is the radius of the wheel, l is the distance of each wheel from point P, φ is the rotation of the wheel, and $r\dot{\varphi}_i\ i = (1,2)$ is the speed of each wheel. The angular velocity of each wheel with respect to P ($\dot{\theta}_i$) and the total angular velocity ($\dot{\theta}$) are calculated as follows in Equations (4) and (5) [47].

$$\dot{\theta}_i = \frac{r\dot{\varphi}_i}{2l} \tag{4}$$

$$\dot{\theta} = \frac{r}{2l}(\dot{\varphi}_1 - \dot{\varphi}_2) \tag{5}$$

The motion of the robot can be described as shown in Equation (6):

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \dot{\varphi}_1, \dot{\varphi}_2) \tag{6}$$

The robot's motion is obtained by rotating the robot coordinate system around the z-axis with respect to the base coordinate system. In this case, the rotation matrix around the z-axis is given in Equation (7) [47].

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

By utilizing the transformation matrix between the coordinate systems of the robot, the motion of the robot can be written as follows [47] in Equations (8)–(11).

$$\xi_R = R(\theta)\xi_I \tag{8}$$

$$\xi_I = R(\theta)^{-1}\xi_R \tag{9}$$

$$R(\theta)^{-1} = R(\theta)^T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$\xi_I = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{r}{2} \begin{bmatrix} \dot{\varphi}_1 + \dot{\varphi}_2 \\ 0 \\ \dfrac{\dot{\varphi}_1 + \dot{\varphi}_2}{l} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{11}$$

Figure 1 shows the current position of the robot and the position of the target point. The values for the target position are calculated using the units shown in Figure 1 [47]. AH is the angle between the robot's orientation and the target, and PH is the distance between the robot and the target. These values can be obtained using Equations (12)–(16).

$$PH = \sqrt{D_x^2 + D_y^2} \tag{12}$$

$$AH = \theta_r - \theta \tag{13}$$

$$D_x = x_h - x_r \tag{14}$$

$$D_y = y_h - y_r \tag{15}$$

$$\theta = \arctan\left(D_y, D_x\right) \tag{16}$$

3.3. Multiple Mobile Robots

Multi-robot systems are used for some tasks that a single robot cannot perform or which may be more costly. Instead of one powerful robot with a complex structure, multiple robots with more superficial structures can be used. Examples of multi-robot applications include exploration, search and rescue, and transportation [48–50]. There are two main ways in which multi-robot systems make decisions. These are centralized systems and distributed systems. In centralized systems, the movements of the entire robot group are managed by a single center. All information is collected in a central system or robot; the most appropriate decision is made there. In distributed systems, each robot makes its own decision. It can act independently of the other robots. Apart from these, market-based approaches have also been developed. In this approach, robots can perform a task on their own, or they can perform a given task as a robot community [51].

Multiple robots need to be coordinated while performing a task. While forming the robot system, different classifications can be made according to the control strategy. Examples are behavioral, leader–follower, and virtual structure approaches [52]. Since the behavior-based control strategy integrates several goal-oriented behaviors simultaneously, multiple robots can simultaneously navigate to waypoints, avoid hazards, and maintain formation [53]. This control strategy allows robotic vehicles to concentrate and act on the inputs received by their sensors. Thus, all robots in the formation respond to the information they receive from their environment and ensure the formation is fully realized [54]. The virtual structure approach is used for the formation of structures that have the appearance of a fixed geometric shape. A general control strategy is developed to force a collection of robots to behave like particles embedded in a solid structure. In this approach, there is no need to select a specific leader for the formation [55]. Multi-robot systems using the leader–follower control method have at least one leader, with the rest of the agents acting as followers. The control design is such that the followers monitor the leader's position and follow the leader's prescribed trajectory. Two types of supervision are required here. One maintains the specified distance between the leader and the followers, and the other keeps the relative angle between them [56].

The platoon formation is one of the most frequently used formation types in the literature [57]. In the platoon driving approach, the mobile robot system consists of more than one robot moving in a line like a single object. In this structure, each robot usually only receives information from the previous one [58]. The leader follows a set trajectory, and the followers must maintain a desired relative distance and orientation between them. Longitudinal controllers focus on keeping the relative distance between vehicles at a predetermined value [59]. Ref. [60] introduces a distributed guiding-vector-field (DGVF) algorithm for robots to form spontaneous-ordering platoons along a predefined path in an n-dimensional Euclidean space. In [60], the algorithm eliminates singular points and guides robots toward a self-intersecting path. Ref. [61] proposes a four-layer framework for cooperative mechanisms within and across strings and designs longitudinal and lateral controllers based on vehicle roles. Many approaches to this type of control have been proposed [57,62,63]. Another study used a platooning approach involving air and ground vehicles [64].

This study preferred a distributed system according to the decision-making approach. The leader–follower approach was used as the control method. Detachment was used as the formation type.

4. Materials and Methods

4.1. Proposed NLP-Based Command-Creation Model

This study covers two main areas. These are the field of robotics and the field of NLP. Therefore, the study is conducted on two levels. The NLP system detects what the user says or writes, translates it into an appropriate format, and transfers it to the robot system. The robotic part is the structure that implements these commands on a robot and realizes them in the simulation. The stages required to realize these processes are divided into four parts. A block diagram of these stages is shown in Figure 2.

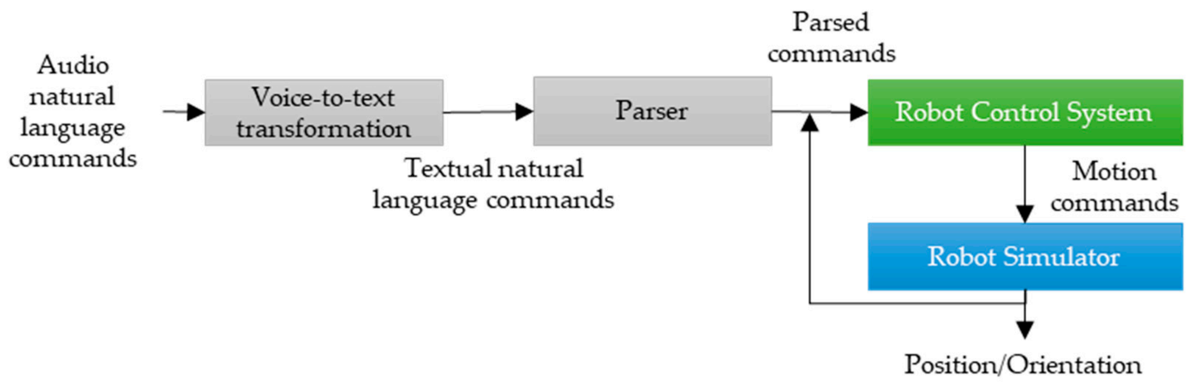

Audio natural language commands
Voice-to-text transformation
Parser
Textual natural language commands
Parsed commands
Robot Control System
Motion commands
Robot Simulator
Position/Orientation

Figure 2. A block diagram of the proposed approach.

First, a morphological analysis is performed on the commands, which are received as text from the user or as audio and converted into text. Here, the morphology module in the Zemberek application is used [10]. The types of words in the given sentence, such as adjectives, nouns, and verbs, are obtained via the morphological analysis. For example, if we analyze the sentence "Yeşil bardağı kutuya koy (English: "Put the green glass in the box")," the following information will be obtained (hereafter, English translations of the Turkish command sentences will be given in parentheses after the statement for a better understanding of the commands):

yeşil ("green") → adjective
bardak ("glass") → noun
kutu ("box") → noun
koy ("put") → verb

As can be seen from the result, we start with a morphological analysis, but the output we obtain at this stage is not enough. In sentence structure, the component that performs the action and is affected by the action must be obtained. To understand the task in a task sentence, it is necessary to infer the purpose for which the words are used in the sentence rather than their types. For this reason, a syntactic inference must be made. The elements of the task must be identified by correctly separating the sentence into its components.

4.1.1. Sentence Analysis

Natural language commands to be sent to the robot are analyzed by the parser. After analyzing the command sentence, the complement and predicate of the sentence are obtained. With these data, the function of the robot is determined. Since Turkish is an agglutinative language, each word has a complex structure in terms of its form; therefore, the words we derive from a word can have very different meanings.

For example, let us consider the word "kitap" in this context:

Kitap ("Book") → noun: A collection of printed or written sheets of paper.
Kitapçı ("Bookseller") → noun: A person who sells books.
Kitaplık ("Bookcase") → noun: A compartment where books are placed.
Kitaplı ("Bookish") → adjective: A person who has books.
Kitapsız ("Bookless") → adjective: A person without books.

As can be seen, the root of the word is a noun, but it can be used to create many different meanings in a sentence.

In this study, the "Turkish Sentence Analyzer (TSA)" was used for sentence dependency analysis [11,12]. The TSA is an open-source Java application that performs sentence parsing for Turkish sentences. The book "Turkish Syntax" was used to create Turkish syntax rules [65]. Context-free Grammar (CFG), which is a method for describing the structure and syntax of natural languages, was used to test the suitability of the text structure given in the application [66].

Turkish sentences may be created in various forms, for example, "sentence → verb", "sentence → object + verb", "sentence → subject + complement + verb", "sentence → subject + object + verb", etc. As can be seen from these examples, a syntactic inference must be made, and it is critical to break down sentences into their elements correctly. Figure 3 shows the parse tree of the sentence structure of "robot ileriye git ("robot go forward")" generated from the defined grammar.

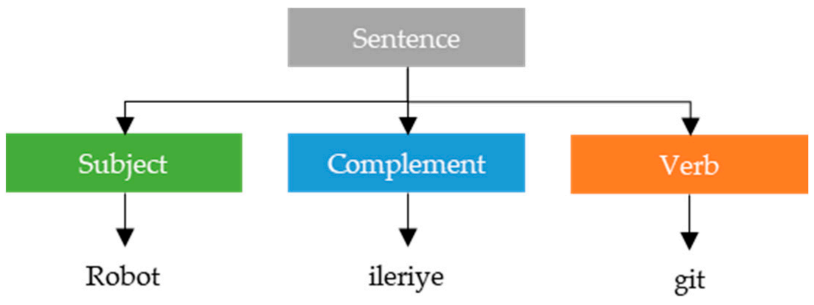


Figure 3. The parse tree of the sentence structure of "robot ileriye git".

CFG is mainly a grammar used to describe a language. The description of natural language cannot cover all language because semantic expressions cannot be defined in this grammar. As can be seen in the example shown in Figure 3, this grammar is created from Turkish syntactic structures. If a sentence given with this grammar is recognized as input, the sentence is accepted as valid.

TSA uses Zemberek in the morphological analysis phase while analyzing the given sentence [10]. When performing the morphological analysis, more than one result can be obtained. We can examine this using an example of the Turkish word "boyun", which has several meanings. Possible solutions are presented in Table 1. The analyzer basically works as follows: the given sentence is separated on a word-by-word basis, and these words are separated into nouns, adjectives, verbs, etc., with the help of Zemberek.

Table 1. Morphological analysis of the Turkish word "boyun".

Solution	Turkish Root	Turkish Equivalent	English Translation	Word Type	Meaning
1	Boyun	Boyun	Neck	Noun	an area on the body
2	Boy	Uzunluk	Length	Noun	the size of an object
3	Boy	Kabile	Tribe/Clan	Noun	a group of persons

Detailed examples of the identification of word types in different sentences using Zemberek are shown in detail in [67].

Finding a Suitable Solution

First, only sentences with verbs were used in this study to find a suitable solution. Part-of-speech tagging (via a POS tagger), which uses statistical data, was added to the system to find the appropriate solution. Here, the frequencies of the generated solutions were determined. Each similar solution was identified as an element of a class. The number of classes was unrestricted due to the many solutions found. Then, the frequencies of the specified class elements were calculated. This calculation is shown in Equation (17).

$$p_i = \frac{f_i}{n}\ , \quad i = 1,\ 2\ \ldots .\ k \quad ; \quad \textstyle\sum_{i=1}^{k} p_i = 1 \tag{17}$$

In the equation, p_i is the frequency rate of each class, f_i is the frequency of each class, n is the total number of data points, and k is the number of classes. The class element with the highest ratio among the obtained solutions is considered the solution.

Creating a Flow Sequence

After the essential analysis and parsing, a step was carried out to extract the flow order of sentences that contain more than one task and are parsed with specific words. What is meant by flow order here is which task should be carried out first. For example, if we take the sentence "Kapıyı geçtikten sonra yerdeki kutuyu al. ("After passing the door, pick up the box on the floor.")", the first task is to "kapıyı geçme ("pass the door")", and the second task is to pick up the "yerdeki kutuyu ("box on the floor")". Here, rules are created based on time-indicating words such as "önce ("before")" and "sonra ("after")" in the sentence. A sample demonstration in which all these operations are performed is as follows.

Below is the analysis made by the application for the expression "Robot ileriye git ("Robot go forward")":

"robot/noun"; "ileriye/noun"; "git/verb";

"subject/robot"; "complement/ileri ("forward")"; "predicate/git ("go")".

With the predicate obtained as a result of the analysis, the operation to be performed and the direction information are obtained with the complement of the sentence. In this process, the conjunctions that can determine the orders of operation of the given sentences were identified, and the operations in the given expression were ordered according to these conjunctions. As an initial study, the order of the operations was determined by checking the conjunction "sonra ("then")" in the command statement entered into the system.

Another example of the expression "Robot ileriye git sonra sola dön ("Robot go forward then turn left")" is analyzed as follows:

First action:

"subject/robot"; "complement/ileri ("forward")"; "predicate/git ("go")".

Second action:

"complement/sol ("left")"; "predicate/dön ("turn")".

The word "sonra ("after")" in the expression shown in the example above indicates time and sequence. When the application sees the word after, it understands that the actions should be performed sequentially and sends the statements to the robot control system in this order.

The third improvement in sentence analysis is the addition of numerical units of measurement. The following example illustrates the new additions to the program. "Robot, iki metre ileriye git sonra kırk beş derece sola dön. ("Robot, move forward two meters and then turn forty-five degrees to the left.")". Here, the numbers are decomposed through a dependency analysis. Our parsing methodology for creating commands for such sentences with multiple actions is described in more detail in the next section.

4.1.2. Creating Commands

The solutions obtained after parsing must be adapted to the robot system. For this reason, a new data structure was created which is illustrated in Figure 4. In this data structure, the parsed commands are kept in a form suitable for the functions of the robot system.

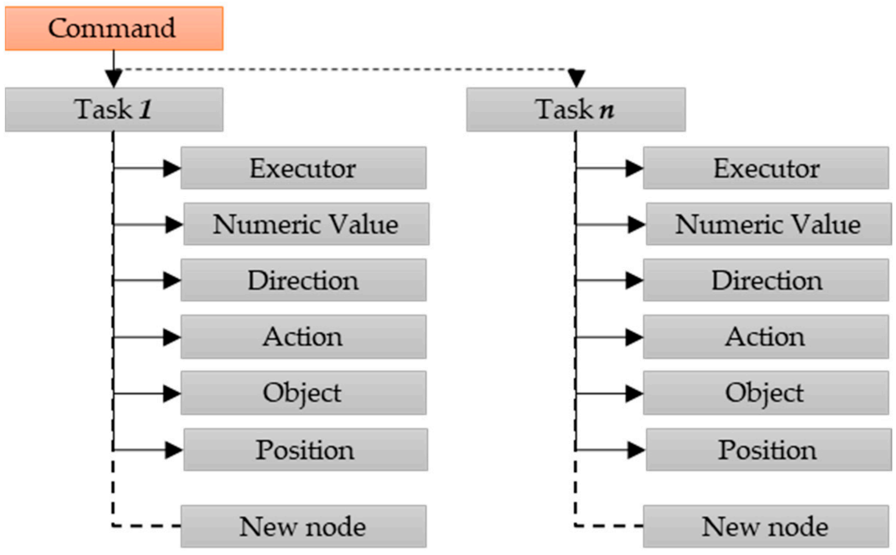


Figure 4. Scheme of new data structure.

It contains the mobile robot that will act, the action to be performed, direction information, trajectory information, obstacle information, and the front and back positions of the obstacle. These commands are converted into a data format and sent to the robot system. For example, the data structure created after parsing for the command "Robot ileriye git sonra 45 derece sağa dön ("Robot go forward and then turn 45 degrees to the right")" is shown in Figure 5.

In Figure 5, the command consists of two actions. Here, the robot will perform the given task. In single-robot environments, the command is applied directly to the robot in the system. In multi-robot environments, the identity of the robot must also be specified. This command keeps the direction information separate because it also takes a number value. First, "Task 1" commands are sent, and then "Task 2" commands are sent to the robot system.

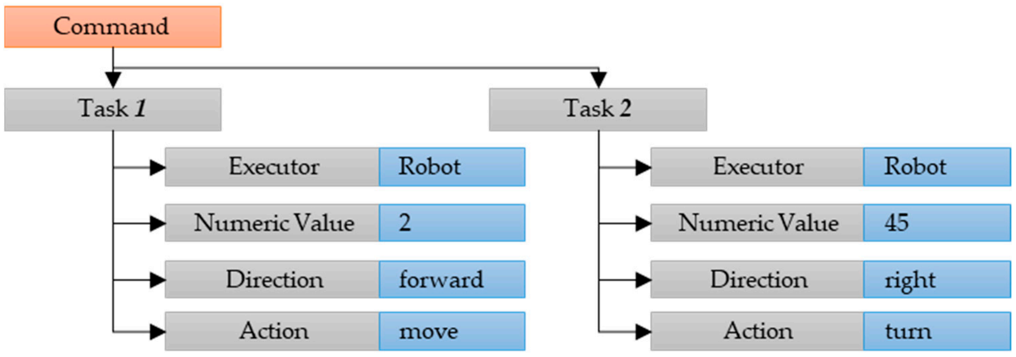


Figure 5. The created data structure scheme after parsing for the command containing two actions.

Figure 6 shows the data structure format for the command "Lider beyaz engelin arkasına git sonra yeşil engelin önüne git ("Leader go behind the white obstacle and then go in front of the green obstacle")". Here, the leader robot in the multi-robot system is the one that will perform the task. In the command, an obstacle is given as an obstacle, and the location is behind and in front of the obstacle. The data structure keeps these parts as the object type and position. In this study, since the robots are tested in a known and mapped environment, the features of the objects, such as their color, are available on the map in the simulation environment.

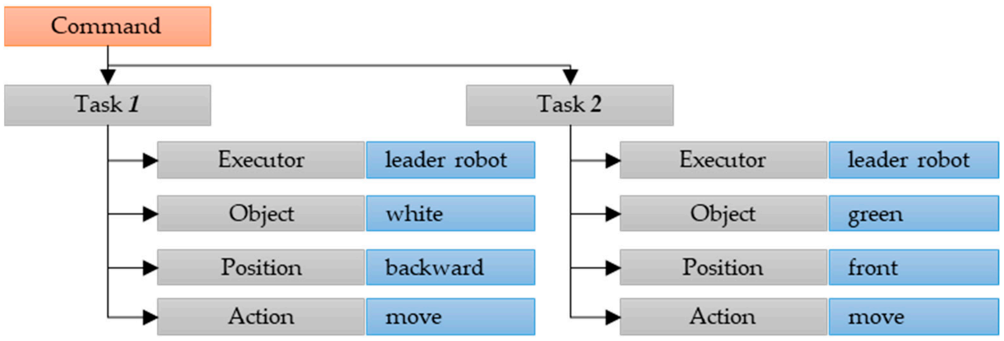


Figure 6. The data structure scheme created after parsing for the command "Lider beyaz engelin arkasına git sonra yeşil engelin önüne git ("Leader go behind the white obstacle and then go in front of the green obstacle")".

4.2. Simulation Environment

In this study, Gazebo was used due to its ability to connect directly to ROS through special packages; it is a 3D interface that provides the robots, sensors, plugins, and environment models required for robotic simulations. It exhibits high performance and has various plugins. The Gazebo simulator can be connected directly to ROS through specialized packages. These packages provide the necessary interfaces to simulate a Gazebo robot using ROS messages, services, and dynamic reconfiguration [68]. For ROS to work correctly with a robot, it needs a description of the robot's kinematics. In this way, trajectories, navigation, and more can be planned and executed. To describe a robot, the robot's properties are specified in URDF (Universal Robot Description Format) files [68,69]. In this work, all experiments were carried out in the Gazebo environment on the map shown in Figure 7.

The map in the simulation shows four circular obstacles colored green, red, blue and white, which were also described in the commands in the previous sections.

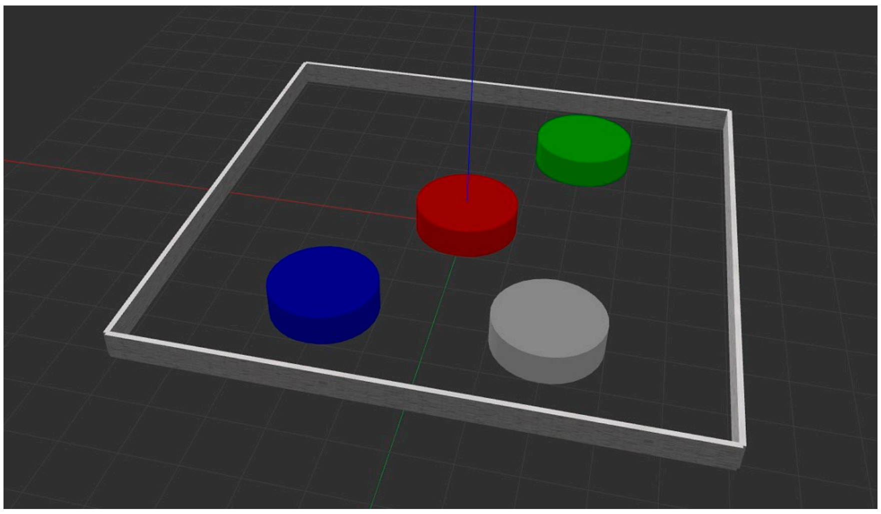


Figure 7. The simulation environment was created in Gazebo.

4.3. Mobile Robot Platform

In this study, the Turtlebot3 Burger mobile robot was used [70]. TurtleBot3 is a small, programmable, ROS-based mobile robot for use in education, research, hobby robotics, and product prototyping [70]. Most of the work implementing ROS features such as simulations, drivers, SLAM, and navigation is performed using Turtlebot3 packages. Detailed technical specifications of the Turtlebot3 Burger model can be found in [70]. The primary sensor used for navigation is a 360-degree LIDAR sensor. Using this sensor, obstacles in the environment were detected. Secondly, the camera was used to detect the features of the obstacles, such as their color.

4.4. Robot Control

In this study, a controller was used for more than one process. The processes are as follows: path planning, establishing the leader–follower system, and formation generation. The structure of the controller used in the system is shown in Figure 8.

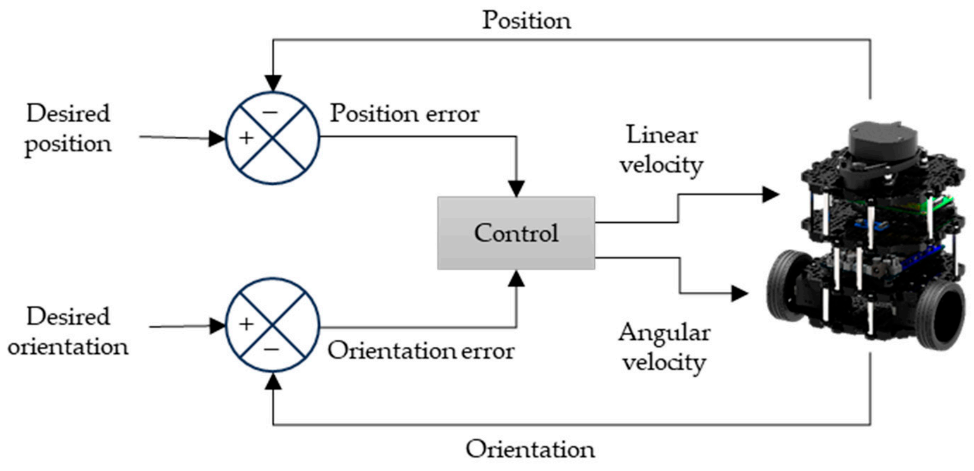


Figure 8. Block diagram of mobile robot control.

The system takes the desired position and orientation as inputs. The error is obtained by taking the difference between these inputs as a reference and the actual position and orientation of the robot. The error is processed in the controller, and a linear velocity and angular velocity are generated for the robot. This process continues until the error is zero or reaches the reference range. A PID controller is used as the controller in the system.

4.5. Trajectory Planning

In this study, the A* algorithm was used to find the shortest path between the start point and the end point. The algorithm takes the instantaneous actual position of the robot and the coordinates of the calculated target point as parameters. It determines the optimal path using these parameters. The function returns information about the node that the robot should visit. The Euclidean distance was used as the heuristic function's cost in the algorithm.

4.6. Determining Target Point

Verbal expressions are received from the user, and the robot performs these commands after an NLP analysis. However, these verbal expressions must be transformed so the robot system can understand them. For example, when the command "Robot kırmızı engelin arkasına git ("Robot goes behind the red obstacle")" is given, the position of the desired target must be calculated. For this, the numerical value of the back target position of the verbally given obstacle is calculated, as shown in Figure 9.

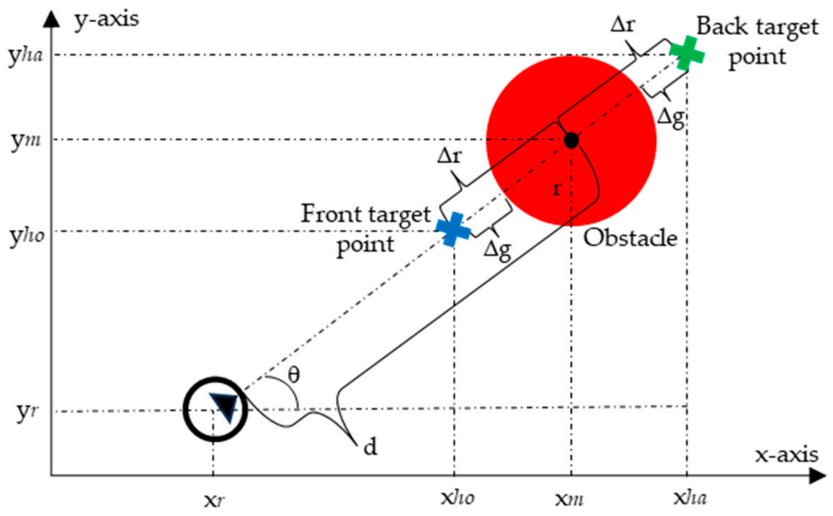


Figure 9. A block diagram of the calculation of the robot's target destination.

In Figure 9, θ is the angle between the robot and the origin of the obstacle, d is the distance between the robot and the origin of the obstacle, r is the radius of the obstacle, Δg is the desired distance between the target point and the obstacle, Δ is: the distance between the target point and the origin of the obstacle, x_r, y_r is the current position of the robot, x_m, y_m is the position of the origin of the obstacle, X_{ha}, Y_{ha} is the back target position of the obstacle, and X_{ho}, Y_{ho} is the front position of the obstacle. Also, R_o is the distance between the robot and the front target points, and R_a is the distance between the robot and the back target point. The equations for these parameters are shown in Equations (18)–(26).

$$\theta = \arctan\left(\frac{y_m - y_r}{x_m - x_r}\right) \tag{18}$$

$$d = \sqrt{(\,x_m - x_r)^2 + \left(y_m - y_r\right)^2} \tag{19}$$

$$\Delta r = r + \Delta g \tag{20}$$

$$R_a = d + \Delta r \tag{21}$$

$$X_{ha} = R_a \cos\theta \tag{22}$$

$$Y_{ha} = R_a \sin\theta \tag{23}$$

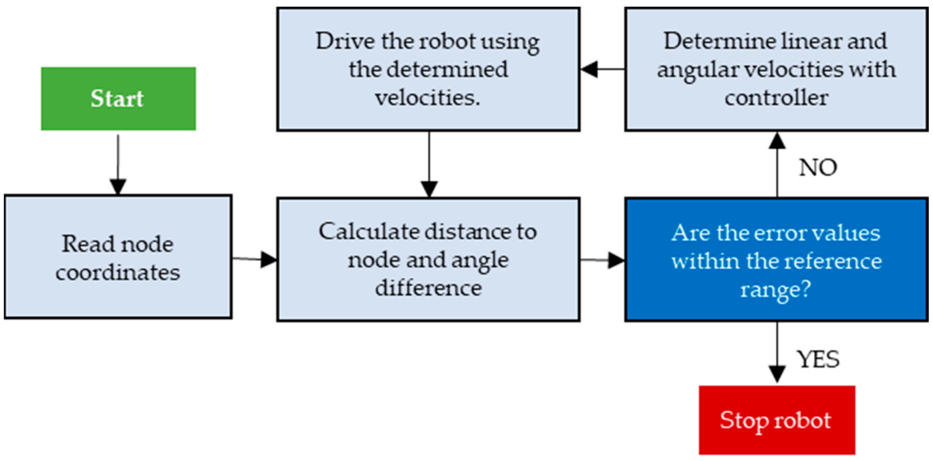
$$R_o = d - \Delta r \tag{24}$$

$$X_{ho} = R_o \cos\theta \tag{25}$$

$$Y_{ho} = R_o \sin\theta \tag{26}$$

4.7. Path Tracking

The optimal path between the robot's origin and its destination point is found by the A* algorithm. The algorithm calculates the path according to the size of the area used the distance to the target point, and node values are returned. These node coordinates are sent to the controller system as reference values, and the robot reaches the target. Figure 10 shows a block diagram of the process.



Start
Read node coordinates
Calculate distance to node and angle difference
Are the error values within the reference range?
NO
Determine linear and angular velocities with controller
Drive the robot using the determined velocities.
YES
Stop robot

Figure 10. Block diagram of path tracking.

As shown in Figure 10, the system first reads the coordinates of the next target node. The distance between the robot and the target is calculated according to Equation (27), and the angle is calculated according to Equation (28). If these calculated values are within the reference range, the robot stops and waits. If the error values are outside the reference range, the controller uses these values to determine the angular velocity and linear velocity of the robot. The robot is driven at these determined velocities. This process continues until the error values enter the reference range.

$$d = \sqrt{(x_d - x_r)^2 + (y_d - y_r)^2} \tag{27}$$

$$\theta = \arctan\left(\frac{y_d - y_r}{x_d - x_r}\right) \tag{28}$$

where x_d, y_d are the coordinates of the node, x_r, y_r are the coordinates of the robot, d is the distance between the robot and the node, and θ is the angle between the robot and the node.

4.8. Leader Following

The aim is to maintain a certain distance and angle between the leader and follower robots. The followers need to keep the desired alignment. Figure 11 shows the structure of following the leader. As can be seen in the figure, the distance between the leader and the follower is denoted by l. The formation aims to keep this distance in the reference distance range and the orientation between the two robots in the reference orientation range.

The formation works as follows: The leader and follower robots communicate using the publish–subscribe method. The leader robot broadcasts its position and orientation, and the follower robot receives this information. The controller receives the current position and orientation of the leader as inputs. This information is used as the target coordinates of the follower robot. The desired distance between the target coordinates and the desired distance are taken as references to determine the amount of error, and the controller determines the linear and angular velocity of the follower robot according to this error. If the follower robot enters the specified reference range, the speed information is reset to zero, and the robot stops.

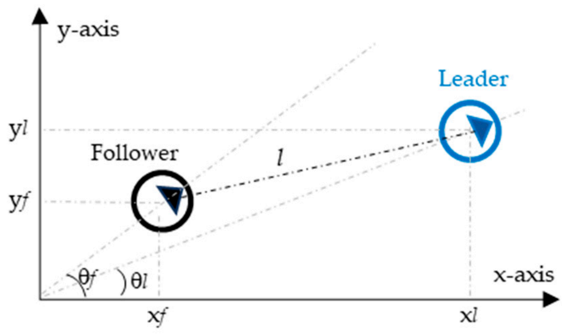


Figure 11. Leader–follower formation.

5. Experimental Studies

This section presents experimental studies on the formation control of multiple mobile robots using NLP and different scenarios envisioned for formation control. The Turkish NLP approach asks a group of mobile robots to follow a prescribed or desired trajectory under other conditions. In the simulation environment, obstacles were assumed to exist, and the mobile robots were asked to move. Considering these obstacles, the robots were given commands in Turkish and asked to do what was necessary. The general view of the environment with obstacles in which the robots move is shown in Figure 7. After calculating the coordinates of the target point, the A* algorithm is used to calculate the optimal/economic path.

The terms "In front of the target" and "behind the target" in the Turkish commands given to the robot are shown in Figure 9, and the formulation is generated. Some preliminary assumptions were made, and experimental studies were carried out on the formation control of the mobile robot system using Turkish NLP. Instead of very complex Turkish sentences, sentence structures containing one or more commands were used. The functional status of the leader and follower robots in formation control was initially determined as leader, follower 1, follower 2, . . . follower n. Multi-robot systems using the leader–follower control method have at least one leader, with the rest of the agents as followers. The control design is such that the followers monitor the leader's position and follow the leader's predicted trajectory. The leader and followers do not fail, and the followers follow the leader by communicating. There is no dynamic obstacle to the leader and followers. Two types of supervision are required here. One maintains the specified distance between the leader and the follower, and the other keeps the relative angle between them. A PID controller was used as the control method. PID coefficients were obtained via the trial-and-error method as follows: Kp = 1; Ki = 0.5; Kd = 0.5. These coefficients were used for both leaders and followers.

The leader–follower approach was used for formation control, and the platoon approach was used as the formation type. In our study, a platoon formation was adopted as the formation type that expresses the positioning of the robots. In the platoon driving approach, each robot receives information only from the previous one, and all mobile robots move in a line as if they were a single object. According to the decision-making approach, a distributed system is preferred.

The leader follows a set trajectory. The followers have to maintain a desired relative distance and orientation between them. In the environment used, the positions of the obstacles are fixed. The radius of the obstacle is taken as r = 1 m, and the distance to approach the obstacle is taken as Δg = 1 m at target positions, such as in front of and behind the obstacle. In this case, the robot can approach the center of the target up to 2 m. This is valid for the target location. The robot can be closer to other obstacles as it moves forward. The distance between the leader robot and the follower is variable. The robot's maximum speed was set to 0.22 m/s, and its maximum rotation speed was 2.8 rad/s. Due to page limitations, only two test results are presented. These are in the form of a Turkish sentence containing one direct command and two interlinked statements in a sentence.

5.1. Case 1: "Lider Kırmızı Engelin Arkasına Git ("Go behind the Red Obstacle")" Command

"Lider kırmızı engelin arkasına git ("Go behind the red obstacle")" was given in Turkish to the leader of a traveling robot group of three robots. The command was first decomposed into its elements using a NLP:

"subject/lider ("leader")";

"complement/kırmızı engelin arkası ("behind the red obstacle")";

"predicate/git ("go")".

After parsing, the command structures obtained were applied to the robot system, and the necessary actions were requested. While the leader follows the prescribed trajectory, the other follower robots follow the leader in a platoon formation. This application was carried out considering two different initial conditions to examine whether the initial conditions affected the robots' behavior. The initial positions of the robots are given in Table 2.

Table 2. Initial coordinates of robots for Case 1.

Robot Name	X	Y
Leader	2	0
Follower 1	3	0
Follower 2	4	0

According to the first initial conditions of the robots, the leader and followers depend on the command given to the leader. The robots realized the predicted trajectory, as shown in Figure 12. As seen in Figure 12, the leader robot planned its path according to the command given to it and went to the target location. When the position and orientation errors of the leader robot are analyzed, it can be observed that the error increases at node transitions. Still, when the target node is reached, the errors approach zero. Initially, there is a distance of one meter between the robots. While the leader robot makes sharper turns at some points according to the given trajectory, since the followers follow their leader, they cannot keep up with these turns properly and follow the leader with more deviations than in other places.

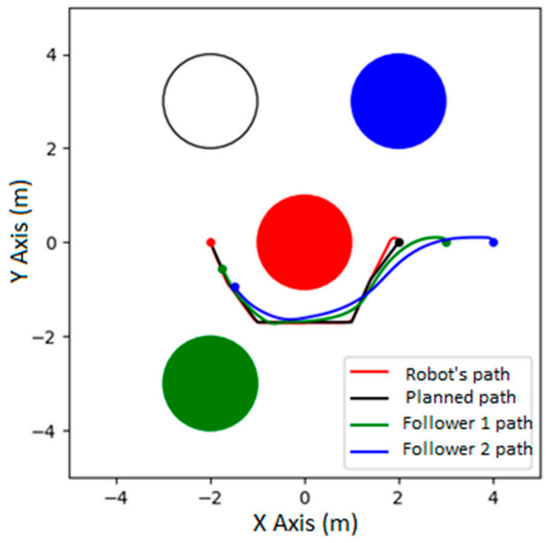


Figure 12. Position–path graph for Case 1.

5.2. Case 2: "Lider Beyaz Engelin Arkasına Git Sonra Yeşil Engelin Önüne Git ("Leader Go behind the White Obstacle and then go in Front of the Green Obstacle")" Command

There is a two-step process here. The robots have to pass behind the white obstacle and move from there to the front of the green obstacle. The target point is in front of the green obstacle.

"Lider beyaz engelin arkasına git sonra yeşil engelin önüne git ("Leader go behind the white obstacle and then go in front of the green obstacle")" was given in Turkish to the leader of the traveling robot group of three robots. The command was first decomposed into its elements using NLP:

First action:
"subject/lider ("leader")";
"complement/beyaz engelin arkası ("behind the white obstacle")";
"predicate/git ("go")".
Second action:
"subject/lider ("leader")";
"complement/yeşil engelin arkası ("in front of the green obstacle")";
"predicate/git ("go")".

The subject of the command is the leader robot. Here, a conjunction is used to indicate sequence and time. The location information is behind the white obstacle and in front of the green obstacle. The predicate of the sentence is the action of going. The data structure used for this process is given in Figure 4. This decomposition was then applied to the robot system, and they were required to carry out the task accordingly. In this experiment, the starting positions of the robots were taken, as shown in Table 2. While the leader followed the prescribed trajectory, the other follower robots followed the leader in a platoon logic. Figure 13 shows the position–path graph of this experiment.

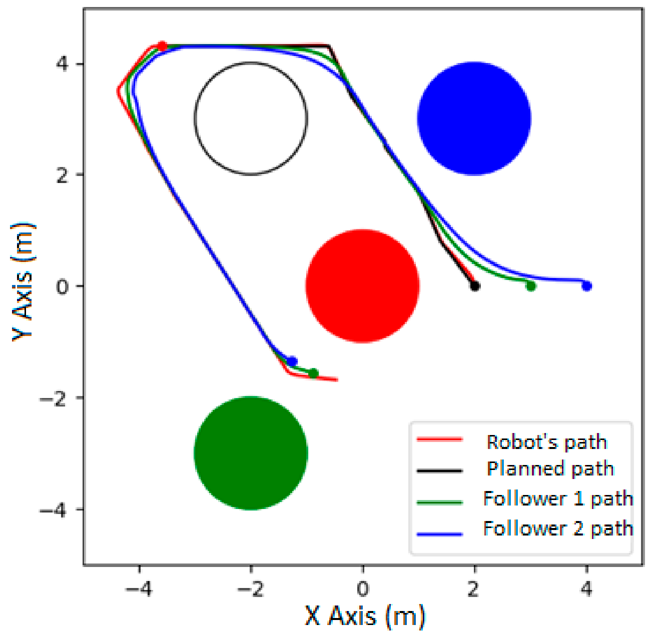


Figure 13. Position–path graph for Case 2.

Due to differences in the initial conditions, followers 1 and 2 tried to follow the leader using a shorter route without passing through the leader's initial position. A similar situation occurs when the leader made a sharp turn. The position and orientation of the paths taken by the leader and the followers concerning the predicted initial positions can be clearly seen in the position–path graph.

6. Discussion

This study focuses on using Turkish Natural Language Processing for the formation control of mobile robot systems, particularly focusing on the leader–follower approach. Through experiments conducted in the Gazebo simulation environment with Turtlebot 3 Burger robots, this study demonstrates the effectiveness of interpreting and executing Turkish commands for navigation and formation control. In this study, by preprocessing natural Turkish commands and employing path-planning algorithms, the robots successfully navigate to desired locations and maintain their formation in multi-robot scenarios. The agglutinative structure of the Turkish language presents challenges in the context of NLP for mobile robot control. The grammar structure of Turkish may be defined by its extensive use of suffixes, which are appended to root words, especially to verbs, to convey complex meanings and grammatical functions. This agglutinative structure has highly variable and context-dependent command structures, making it challenging to parse and interpret commands for robot control accurately. Moreover, Turkish exhibits rich morphological features, such as case markings and verb conjugations, which further complicate language-understanding processes. As a result, developing robust and reliable approaches for preprocessing Turkish commands and extracting meaningful information for NLP applications poses significant difficulties. Addressing these challenges necessitates a comprehensive understanding of Turkish grammar and morphology to increase robot–human interaction and develop innovative NLP approaches tailored to agglutinative languages' intricacies. By integrating language-processing capabilities into multiple-robot formation control, we propose an approach for more intuitive human–robot interaction, potentially expanding the accessibility and usability of robotic technology in Turkish language applications. Furthermore, this study establishes a foundation for future research in Turkish NLP applications and robot control aimed at enhancing the robustness and versatility of natural language-based control systems for mobile robots.

7. Conclusions

In this study, formation control of a mobile robot system was achieved using Turkish Natural Language Processing. The emphasis was on leader–follower formation control based on natural language. A Gazebo simulation was used as the experimental environment, and a Turtlebot 3 Burger was used as the mobile robot. Our main motivation in this paper was not to develop a new Turkish morphological analyzer or a new swarm formation algorithm. As mentioned before, our primary motivation was to perform robot formation control using a language like Turkish, which was challenging to analyze in this study. Another key point of our work is that it can be embedded in multi-robot systems. In this study, Turkish commands could be received both verbally and in writing. These commands could be in front of an obstacle, behind an obstacle, or as a combination of two different expressions due to the test environment. However, various combinations can be used in the proposed method. Creating an intermediate data structure situated between natural language commands and robot system control makes these commands meaningful for the robot. The commands were first separated into their components and then transformed into the necessary structures according to their components. The target location was extracted from the location information obtained, and the coordinates of the target location were determined by making the necessary calculations. A suitable path for the robots to follow to the target coordinate was planned using the A* algorithm. This path was then tracked using a designed PID controller. Experiments were first conducted with a single robot. After a single robot successfully navigated to the desired location, experimental studies with multiple robots were performed and are presented in this paper. The leader–follower approach was used in multiple robots. A platoon formation was used as the robot formation. It was observed that the robots reached the target point by maintaining the desired distance between each other. In this study, it was ensured that the mobile robot system was controlled using the Turkish natural language. According to our literature review, this study is at the forefront of controlling mobile robot systems using the Turkish natural language.

Author Contributions: Conceptualization, G.E.; methodology, K.A., G.E. and B.C.; software, K.A.; validation, K.A., G.E. and B.C.; formal analysis, K.A., G.E. and B.C.; investigation, K.A., G.E. and B.C.; resources, G.E.; data curation, K.A., G.E. and B.C.; writing—original draft preparation, K.A., G.E. and B.C.; writing—review and editing, K.A., G.E. and B.C.; visualization, K.A.; supervision, G.E. and B.C.; project administration, G.E. and B.C.; funding acquisition, G.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article.

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

References

1. De Greeff, J.; Belpaeme, T.; Bongard, J. Why Robots Should Be Social: Enhancing Machine Learning through Social Human-Robot Interaction. *PLoS ONE* **2015**, *10*, e0138061. [CrossRef] [PubMed]
2. Yang, H.D.; Park, A.Y.; Lee, S.W. Gesture Spotting and Recognition for Human-Robot Interaction. *IEEE Trans. Robot.* **2007**, *23*, 256–270. [CrossRef]
3. Dang, J.; Liu, L. Implicit Theories of the Human Mind Predict Competitive and Cooperative Responses to AI Robots. *Comput. Hum. Human. Behav.* **2022**, *134*, 107300. [CrossRef]
4. Cherpas, C. Natural Language Processing, Pragmatics, and Verbal Behavior. *Anal. Verbal Behav.* **1992**, *10*, 135–147. [CrossRef] [PubMed]
5. Loper, E.; Bird, S. NLTK: Natural Language ToolKit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, Philadelphia, PA, USA, 7 July 2002; Volume 1.
6. Manning, C.D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.J.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, MD, USA, 22–27 June 2014.
7. Lorla, S. TextBlob Documentation, Release 0.15, 2. 2018. Available online: https://textblob.readthedocs.io/ (accessed on 1 February 2020).
8. Řehůřek, R.; Sojka, P. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valetta, Malta, 22 May 2010.
9. Honnibal, M.; Montani, I. spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and İncremental Parsing. 2017. Available online: https://www.spacy.io/ (accessed on 1 February 2020).
10. Akın, A.A.; Akın, M.D. Zemberek, an Open Source NLP Framework for Turkic Languages. *Structure* **2007**, *10*, 1–5.
11. Çöltekin, Ç.; Doğruöz, A.S.; Çetinoğlu, Ö. Resources for Turkish Natural Language Processing: A Critical Survey. *Lang. Resour. Eval.* **2023**, *57*, 449–488. [CrossRef] [PubMed]
12. Eryiğit, G. ITU Turkish NLP Web Service. In Proceedings of the EACL 2014—Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics, Gothenburg, Sweden, 26–30 April 2014.
13. Aram, K.; Erdemir, G.; Can, B. Açık Kaynak Kod Türkçe Doğal Dil İşleme Kütüphanelerinin Robotik Uygulamalarda Kullanımı. *İstanbul Sabahattin Zaim Üniversitesi Fen Bilim. Enstitüsü Derg.* **2021**, *3*, 133–137. [CrossRef]
14. MacGlashan, J.; Babeş-Vroman, M.; DesJardins, M.; Littman, M.L.; Muresan, S.; Squire, S.; Tellex, S.; Arumugam, D.; Yang, L. Grounding English Commands to Reward Functions. In Proceedings of the Robotics: Science and Systems, Rome, Italy, 13–17 July 2015; Volume 11.
15. Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. *Artif. Intell. Rev.* **2020**, *53*, 5455–5516. [CrossRef]
16. Arumugam, D.; Karamcheti, S.; Gopalan, N.; Wong, L.L.S.; Tellex, S. Accurately and Efficiently Interpreting Human-Robot Instructions of Varying Granularities. In Proceedings of the Robotics: Science and Systems, Cambridge, MA, USA, 12–16 July 2017; Volume 13.
17. Kollar, T.; Tellex, S.; Roy, D.; Roy, N. Grounding Verbs of Motion in Natural Language Commands to Robots. In *Experimental Robotics*; Khatib, O., Kumar, V., Sukhatme, G., Eds.; Springer Tracts in Advanced Robotics; Springer: Berlin/Heidelberg, Germany, 2014; Volume 79. [CrossRef]
18. Tellex, S.; Thaker, P.; Deits, R.; Simeonov, D.; Kollar, T.; Roy, N. Toward Information Theoretic Human-Robot Dialog. In Proceedings of the Robotics: Science and Systems, Berlin, Germany, 24–28 June 2013; Volume 8.
19. Tran, D.; Yan, F.; Yihun, Y.; Tan, J.; He, H. A Framework of Controlled Robot Language for Reliable Human-Robot Collaboration. In Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Singapore, 10–13 November 2021; Volume 13086 LNAI.

20. Ahmadzadeh, H.; Masehian, E. Modular Robotic Systems: Methods and Algorithms for Abstraction, Planning, Control, and Synchronization. *Artif. Intell.* **2015**, *223*, 27–64. [CrossRef]
21. Normey-Rico, J.E.; Alcalá, I.; Gómez-Ortega, J.; Camacho, E.F. Mobile Robot Path Tracking Using a Robust PID Controller. *Control Eng. Pract.* **2001**, *9*, 1209–1214. [CrossRef]
22. Wang, S.; Yin, X.; Li, P.; Zhang, M.; Wang, X. Trajectory Tracking Control for Mobile Robots Using Reinforcement Learning and PID. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2020**, *44*, 1059–1068. [CrossRef]
23. Chang, H.; Jin, T. Adaptive Tracking Controller Based on the PID for Mobile Robot Path Tracking. In Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Busan, Republic of Korea, 25–28 September 2013; Volume 8102 LNAI.
24. Fnadi, M.; Plumet, F.; Benamar, F. Model Predictive Control Based Dynamic Path Tracking of a Four-Wheel Steering Mobile Robot. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Macau, China, 3–8 November 2019.
25. Bai, G.; Liu, L.; Meng, Y.; Liu, S.; Liu, L.; Luo, W. Real-Time Path Tracking of Mobile Robot Based on Nonlinear Model Predictive Control. *Nongye Jixie Xuebao/Trans. Chin. Soc. Agric. Mach.* **2020**, *51*, 47–52.
26. Bai, G.; Meng, Y.; Liu, L.; Luo, W.; Gu, Q.; Liu, L. Review and Comparison of Path Tracking Based on Model Predictive Control. *Electronics* **2019**, *8*, 1077. [CrossRef]
27. Antonelli, G.; Chiaverini, S.; Fusco, G. A Fuzzy-Logic-Based Approach for Mobile Robot Path Tracking. *IEEE Trans. Fuzzy Syst.* **2007**, *15*, 211–221. [CrossRef]
28. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
29. Besseghieur, K.L.; Trębiński, R.; Kaczmarek, W.; Panasiuk, J. From Trajectory Tracking Control to Leader–Follower Formation Control. *Cybern. Syst.* **2020**, *51*, 339–356. [CrossRef]
30. Hirschberg, J.; Manning, C.D. Advances in Natural Language Processing. *Science* **2015**, *349*, 261–266. [CrossRef] [PubMed]
31. Lauriola, I.; Lavelli, A.; Aiolli, F. An Introduction to Deep Learning in Natural Language Processing: Models, Techniques, and Tools. *Neurocomputing* **2022**, *470*, 443–456. [CrossRef]
32. Şeker, S.E. Doğal Dil İşleme (Natural Language Processing). *YBS Ansiklopedi* **2015**, *2*, 14–31.
33. Kang, Y.; Cai, Z.; Tan, C.W.; Huang, Q.; Liu, H. Natural Language Processing (NLP) in Management Research: A Literature Review. *J. Manag. Anal.* **2020**, *7*, 139–172. [CrossRef]
34. Oflazer, K. Turkish and Its Challenges for Language Processing. *Lang Resour. Eval.* **2014**, *48*, 639–653. [CrossRef]
35. Oflazer, K.; Saraçlar, M. Turkish and Its Challenges for Language and Speech Processing. In *Turkish Natural Language Processing*; Springer International Publishing: Cham, Switzerland, 2018; pp. 1–19.
36. Safaya, A.; Kurtuluş, E.; Göktoğan, A.; Yuret, D. MUKAYESE: Turkish NLP Strikes Back. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022.
37. Yucebas, S.; Tintin, R. Govdeturk: A Novel Turkish Natural Language Processing Tool for Stemming, Morphological Labelling and Verb Negation. *Int. Arab. J. Inf. Technol.* **2021**, *18*, 148–157. [CrossRef]
38. Aleçakır, H.; Bölücü, N.; Can, B. TurkishDelightNLP: A Neural Turkish NLP Toolkit. In Proceedings of the NAACL 2022—2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session, Online, 10–15 July 2022.
39. Tohma, K.; Okur, H.I.; Kutlu, Y.; Sertbas, A. Sentiment Analysis in Turkish Question Answering Systems: An Application of Human-Robot Interaction. *IEEE Access* **2023**, *11*, 66522–66534. [CrossRef]
40. Tzafestas, S.G. Mobile Robot Control and Navigation: A Global Overview. *J. Intell. Robot. Syst. Theory Appl.* **2018**, *91*, 35–58. [CrossRef]
41. Panigrahi, P.K.; Bisoy, S.K. Localization Strategies for Autonomous Mobile Robots: A Review. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 6019–6039. [CrossRef]
42. Filliat, D.; Meyer, J.A. Map-Based Navigation in Mobile Robots: I. A Review of Localization Strategies. *Cogn. Syst. Res.* **2003**, *4*, 243–282. [CrossRef]
43. Laumond, J.P.; Jacobs, P.E.; Taïx, M.; Murray, R.M. A Motion Planner for Nonholonomic Mobile Robots. *IEEE Trans. Robot. Autom.* **1994**, *10*, 577–593. [CrossRef]
44. Huang, Z.; Chu, D.; Wu, C.; He, Y. Path Planning and Cooperative Control for Automated Vehicle Platoon Using Hybrid Automata. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 959–974. [CrossRef]
45. Duchon, F.; Babinec, A.; Kajan, M.; Beno, P.; Florek, M.; Fico, T.; Jurišica, L. Path Planning with Modified A Star Algorithm for a Mobile Robot. *Procedia Eng.* **2014**, *96*, 59–69. [CrossRef]
46. Sanchez-Sanchez, A.G.; Hernandez-Martinez, E.G.; González-Sierra, J.; Ramírez-Neria, M.; Flores-Godoy, J.J.; Ferreira-Vazquez, E.D.; Fernandez-Anaya, G. Leader-Follower Power-Based Formation Control Applied to Differential-Drive Mobile Robots. *J. Intell. Robot. Syst. Theory Appl.* **2023**, *107*, 6. [CrossRef]
47. Siegwart, R.; Nourbakhsh, I.R.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2011; Volume 23.
48. Burgard, W.; Moors, M.; Fox, D.; Simmons, R.; Thrun, S. Collaborative Multi-Robot Exploration. In Proceedings of the Proceedings—IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 1.

49. Sugar, T.; Kumar, V. Control and Coordination of Multiple Mobile Robots in Manipulation and Material Handling Tasks. In *Experimental Robotics VI*; Springer: London, UK, 2008.
50. Fierro, R.; Das, A.; Spletzer, J.; Esposito, J.; Kumar, V.; Ostrowski, J.P.; Pappas, G.; Taylor, C.J.; Hur, Y.; Alur, R.; et al. A Framework and Architecture for Multi-Robot Coordination. *Int. J. Robot. Res.* **2002**, *21*, 977–995. [CrossRef]
51. Chen, X.; Zhang, P.; Du, G.; Li, F. A Distributed Method for Dynamic Multi-Robot Task Allocation Problems with Critical Time Constraints. *Rob. Auton. Syst.* **2019**, *118*, 31–46. [CrossRef]
52. Litimein, H.; Huang, Z.Y.; Hamza, A. A Survey on Techniques in the Circular Formation of Multi-Agent Systems. *Electronics* **2021**, *10*, 2959. [CrossRef]
53. Balch, T.; Arkin, R.C. Behavior-Based Formation Control for Multirobot Teams. *IEEE Trans. Robot. Autom.* **1998**, *14*, 926–939. [CrossRef]
54. Li, D.; Ge, S.S.; He, W.; Ma, G.; Xie, L. Multilayer Formation Control of Multi-Agent Systems. *Automatica* **2019**, *109*, 108558. [CrossRef]
55. Lewis, M.A.; Tan, K.H. High Precision Formation Control of Mobile Robots Using Virtual Structures. *Auton. Robot.* **1997**, *4*, 387–403. [CrossRef]
56. Chen, Y.Q.; Wang, Z. Formation Control: A Review and a New Consideration. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Edmonton, AB, Canada, 2–6 August 2005.
57. Swaroop, D.; Karl Hedrick, J.; Choi, S.B. Direct Adaptive Longitudinal Control of Vehicle Platoons. *IEEE Trans. Veh. Technol.* **2001**, *50*, 150–161. [CrossRef]
58. Velasco-Villa, M.; Cruz-Morales, R.D.; Rodriguez-Angeles, A.; Domínguez-Ortega, C.A. Observer-Based Time-Variant Spacing Policy for a Platoon of Non-Holonomic Mobile Robots. *Sensors* **2021**, *21*, 3824. [CrossRef] [PubMed]
59. Delimpaltadakis, I.M.; Bechlioulis, C.P.; Kyriakopoulos, K.J. Decentralized Platooning with Obstacle Avoidance for Car-like Vehicles with Limited Sensing. *IEEE Robot. Autom. Lett.* **2018**, *3*, 835–840. [CrossRef]
60. Hu, B.B.; Zhang, H.T.; Yao, W.; Ding, J.; Cao, M. Spontaneous-Ordering Platoon Control for Multirobot Path Navigation Using Guiding Vector Fields. *IEEE Trans. Robot.* **2023**, *39*, 2654–2668. [CrossRef]
61. Li, Y.; Tang, C.; Li, K.; He, X.; Peeta, S.; Wang, Y. Consensus-Based Cooperative Control for Multi-Platoon under the Connected Vehicles Environment. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 2220–2229. [CrossRef]
62. Caruntu, C.F.; Copot, C.; Lazar, C.; Keyser, R.D. Decentralized Predictive Formation Control for Mobile Robots without Communication. In Proceedings of the IEEE International Conference on Control and Automation, ICCA, Edinburgh, UK, 16–19 July 2019.
63. Yan, M.; Ma, W.; Zuo, L.; Yang, P. Distributed Model Predictive Control for Platooning of Heterogeneous Vehicles with Multiple Constraints and Communication Delays. *J. Adv. Transp.* **2020**, *2020*, 4657584. [CrossRef]
64. Venzano, E.; Pousseur, H.; Victorino, A.C.; Garcia, P.C. Motion Control for Aerial and Ground Vehicle Autonomous Platooning. In Proceedings of the IEEE 17th International Conference on Advanced Motion Control (AMC), Padova, Italy, 18–20 February 2022.
65. Karahan, L. *Türkçede Söz Dizimi*; Akçağ Yayınları: Ankara, Turkey, 2000.
66. Jurafsky, D.; Martin, J.H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd ed.; Pearson, Prentice Hall: Upper Saddle River, NJ, USA, 2014; Chapter 18.
67. Aram, K. Türkçe Doğal Dil İşleme İle Çoklu Gezgin Robot Sistemlerinin Formasyon Kontrolü. Ph.D. Dissertation, Istanbul Sabahattin Zaim University, Istanbul, Turkey, 2023.
68. Tufekci, Z.; Erdemir, G. Experimental Comparison of Global Planners for Trajectory Planning of Mobile Robots in an Unknown Environment with Dynamic Obstacles. In Proceedings of the HORA 2023—2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Istanbul, Turkiye, 8–10 June 2023.
69. Chikurtev, D. Mobile Robot Simulation and Navigation in ROS and Gazebo. In Proceedings of the 2020 International Conference Automatics and Informatics, ICAI 2020, Varna, Bulgaria, 1–3 October 2020.
70. TurtleBot 3. Available online: https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ (accessed on 19 November 2023).