CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR THESIS

Filip Eckstein

Formation control in environment with dynamic obstacles

Department of Cybernetics

Thesis supervisor: **Dr. Martin Saska**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Filip Eckstein

Study programme: Cybernetics a Robotics

Specialisation: Robotics

Title of Bachelor Project: Formation Control in Environment with Dynamic Obstacles

Guidelines:

The aim of this thesis is to extend algorithms developed for formations of mobile robots in [3].
Work plan:
- To study and implement an algorithm for description of dynamic obstacles [1,2,3].
- To integrate dynamic obstacle avoidance methods to the system of formation control [3].
- To design and implement at least two models of movement of dynamic obstacles.
- To integrate these models into the rules of the formation control.
- To verify the complete system by a simulation of movement of formations and by a real experiment with the SyRoTek system [4].

Bibliography/Sources:

[1] Stipanovic, D. M.; Hokayem, P. F.; Spong, M. W. and Siljak, D.D.: Cooperative avoidance control for multi-agent systems. Journal of Dynamic Systems, Measurement, and Control, 129:699-707, 2007.

[2] Dimarogonas, D. V.; Loizou, S. G.; Kyriakopoulos, K. J. and Zavlanos, M. M.: A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents. Automatica, 42(2):229-243, 2006.

[3] Saska, M.: Trajectory planning and optimal control for formations of autonomous robots. Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 3. Würzburg: Universität Würzburg. 2010. ISSN: 1868-7466.

[4] Kulich, M.; Košnar, K.; Chudoba, J.; Faigl, J.; Přeučil, L.: On a Mobile Robotics E-learning System. In Proceedings of the Twentieth European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetics Studies, 2010, p. 597-602. ISBN 978-3-85206-178-8.

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the winter semester of academic year 2012/2013

L.S.

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 9, 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Filip Eckstein

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Řízení formace mobilních robotů v prostředí s dynamickými překážkami

Pokyny pro vypracování:

Cílem práce je rozšířit systém pro řízení formace mobilních robotů popsaný v [3] o model pohybu dynamických překážek a výslednou aplikaci přizpůsobit pro použití na robotické platformě SyRoTek.
Plán prací:
- nastudovat, implementovat a odladit algoritmus pro popis dynamických překážek [1,2,3],
- integrovat metody pro vyhýbání se dynamickým překážkám do systému pro řízení formací autonomních robotů [3],
- navrhnout a implementovat minimálně dva modely pohybu dynamické překážky,
- začlenění těchto modelů do pravidel pro řízení formace,
- verifikace výsledného systému simulací pohybu formace reálným experimentem se systémem SyRoTek [4].

Seznam odborné literatury:

[1] Stipanovic, D. M.; Hokayem, P. F.; Spong, M. W. and Siljak, D.D.: Cooperative avoidance control for multi-agent systems. Journal of Dynamic Systems, Measurement, and Control, 129:699-707, 2007.
[2] Dimarogonas, D. V.; Loizou, S. G.; Kyriakopoulos, K. J. and Zavlanos, M. M.: A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents. Automatica, 42(2):229-243, 2006.
[3] Saska, M.: Trajectory planning and optimal control for formations of autonomous robots. Schriftenreihe Würzburger Forschungsberichte in Robotik und Telematik, Band 3. Würzburg: Universität Würzburg. 2010. ISSN: 1868-7466.
[4] Kulich, M.; Košnar, K.; Chudoba, J.; Faigl, J.; Přeučil, L.: On a Mobile Robotics E-learning System. In Proceedings of the Twentieth European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetics Studies, 2010, p. 597-602. ISBN 978-3-85206-178-8.

Vedoucí bakalářské práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce zimního semestru 2012/2013

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

Statement

I hereby declare that I have completed this thesis independently and that I have used only the sources (literature, software, etc.) listed in the enclosed bibliography.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on..... .....

Acknowledgment

Foremost, I would like to thank to Ing. Martin Saska Dr. rer. nat. for his excellent supervision and relevant discussions. I would also like to thank to Ing. Jan Chudoba for a great technical support during the SyRoTek experiments and to Mgr. Dan Erben MRes for critical evaluation of this thesis. Great gratitude also goes to my family for calm and patient environment they provided me during the study.

Abstract

Trajectory planning of nonholonomic formations of mobile robots is a challenging problem in mobile robotics. It enables robotic formations to autonomously move in static and also dynamic environment. The aim of this thesis is to study formation control in environment with dynamic obstacles and integrate dynamic obstacle avoidance algorithm into rules of MPC (Model-predictive control) using framework that is being developed at Department of Cybernetics, CTU in Prague. The final verification of the complete system has been simulated by a movement of formation in a real experiment with the SyRoTek system.

Abstrakt

Plánování trajektorií formací mobilních robotů patří v mobilní robotice k podnětným problémům představujících důležitou součást budoucího rozvoje robotických aplikací. Robotickým formacím umožňuje autonomní pohyb nejen ve statickém, ale i v dynamickém prostředí. Cílem této bakalářské práce je podrobné prostudování teorie řízení formací v dynamickém prostřední a integrace vhodných pravidel umožňujících robotickým formacím vyhýbat se dynamickým překážkám. Práce se zakládá na rozšíření projektu vyvíjeným katedrou Kybernetiky ČVUT v Praze. Ověření funkčnosti výsledného systému proběhne na reálné robotické platformě SyRoTek.

Contents

List of Figures

List of Tables

List of scenarios

1 Introduction

The main goal of this thesis is an integration, simulation and final verification of the extension of the model predictive control (MPC)[1] focused algorithms to avoid obstacles in dynamic environment in experiments with car-like[2] robots according to certain models.

These algorithms will be integrated into rules of the formation control (Section 2.2). The integration of new models of dynamic obstacle avoidance methods consists in design and implementation of an ability to predict a movement of dynamic obstacles that improves the final solution of the robot solver (Section 2.3). The quality of the predetermined model of dynamic obstacle represents a significant influence on the feasibility of the final solution. This work using a model predictive control based theory which is being developed by the Department of Cybernetics, CTU in Prague.

1.1 Motivation

1.1.1 Leader-follower approach

This work is based on the leader-follower approach. In practice, this means that one or more robots are designed to be leaders, and their states are distributed within the rest of formation. The followers are designed to maintain positions relative to the state of the leaders. The utilization of this approach is motivated by heterogeneity of the robotic formation, because it allows to well equip (with additional sensors, computing power etc.) only robots that are determined to be leaders. The rest of robots from the formation (followers) can be simpler and the only technology they must carry are drive and wireless devices to communication with the leader.

1.1.2 Practical use

The purpose of this thesis is to simulate and test on a real robotic platform new implemented rules of model predictive control for trajectory planning. In practice, this approach can be used for many applications nowadays or in the future, because the need for flexible autonomous robots for working in dynamic workspace is still rising. One of the most revealing example in the future can be snow removal from airports (as was showed in [1]) as being one of the target applications of this approach. The network of runways and airport corridors is a fine example of a vast space dynamic environment, where adaptable formation of autonomous robots can be used. Main advantage of using robotic formations lie within an energetic effectivity caused by planning of an optimal cleaning trajectory that can much more easily predetermine a behavior of dynamic obstacles[3] and convert them to

[1] Model predictive control theory will be explained in Section 2.2.
[2] Car-like robot will be described in more detail in Section 2.2.2
[3] In this case cars, planes, etc.

a mathematical model. Optimal trajectory planning in combination with predictive models of dynamic objects can lower the amount of consumed fuel and save time.

Figure 1: Real indoor experiment of robot plow formation developed by Intelligent and Mobile Robotics Group, CTU in Prague.

1.1.3 Contributions

The main contribution of this project lies within the extension of the MPC framework being developed at the Department of Cybernetics, CTU in Prague. Testing the overall MPC system and its simulation on the real multi-robotic platform[4] brought a significant progress in development of real time planning algorithms. A demonstration of the real time online computation of the MPC system on the SyRoTek platform was executed as the first experiment of this kind. Up to now, only the offline version of this approach was tested as is shown in the Figure 1. Implementation and comparison of two predictive models also verified the functionality and usefulness in the real application domains, where the environment consists of dynamic obstacles, where model of movement can be easily predetermined.

1.2 State of the art

Navigation based functions and motion planning of car-like robots formations draws a great deal of interest in mobile robotics. A several methodologies and formulations of controlling multi-agent system to reach the target position in dynamic environment with minimum time and the highest possible robustness in the sense of avoiding collisions whether with obstacles or with other members of the formation has been formed. Among the main

[4]Experiment with the SyRoTek system will be explained in more detail in Section 4

approaches which we will devote this chapter belongs the Receding Horizon Control (presented by Saska, Mejía, Stipanović and Schilling in 2009 [2] and Dunbar and Murray in 2004 [3]) and an example of decentralized concept of Collision Avoidance Scheme for Multiple Independent Non-point Agents (presented by Dimarogonas, Loizou, Kyriakopoulos and Zavlanos in 2005 [4].
In this thesis, we will present a predictive collision avoidance extension of model predictive control approach, but we will also introduce a different point of view in the sense of state space motion planning algorithms with solving the collision scenarios.

The main difference between centralized and decentralized (as was described in [4] concept lies in the knowledge of target region, which is supposed to be reached by other members of the formation. Decentralized method considers a multi-agent formation, where each agent has knowledge only of its own desired destination but not of the others [4], while centralized concept is supposed to (leader-follower approach[5]) share the actual state of the robot leader within the rest of the formation to follow it. It means that the goal region is the same for all the members of the formation. To demonstrate the advantages of decentralized approach, a basic motivation of decentralized method was introduced in [4] and comes from practical application domain, where implemented navigation functions provide increased robustness in the sense of agent failures. This advantage makes the method appropriate for conflict resolutions in air traffic management. According to Dimarogonas, Loizou, Kyriakopoulos and Zavlanos, reduced computational complexity makes the decentralized approach more appealing to centralized ones. Due to simulations of Collision Avoidance Scheme for Multiple Independent Non-point Agents presented in [4], each agent is forced to participate in the conflict resolution procedure even if already in its final destination region. But the decentralized method of navigation of a formations (consisted of car-like robots) is not the only matter of highly critical application such as air traffic control. Jaeger and Nebel in the Dynamic Decentralized Area Partitioning for Cooperating Cleaning Robots [6] used this approach for cooperative cleaning of a large room without global synchronization or global communication network.

Another area of interest, where the robot motion planning can bring a significant improvement can be for example the mapping of hazardous areas (as was described in [7]), where an appropriate sensor equipment can track sources of environmental or biological radiation using Particle Swarm Optimization algorithms (PSO). There are also other interesting applications and methods such as trajectory planning through neural networks and genetic algorithm as is shown in A neural network approach to complete coverage path planning from 2004 [8].

Motion and trajectory planning in the multi-robotic system can be interpreted by several methods and as is shown in the last subsection, it is important to choose an appropriate approach for the most feasible solution of desired mission of our multi-agent robotic system. For purpose of this thesis, we have chosen a model predictive control for driving multi-

[5]The leader-follower approach was also used by Fierro, Das, Kumar and Ostrowski in the Hybrid Control of Formations of Robots from 2001 [5].

robotic formations.

1.3 Model details

Concept of model predictive approach can be interpreted as a predictive information of obstacle's next movement. In practice we can imagine this problem as two cars at an intersection. Driver in the first car observes other car that is moving at the crossroads according to a certain model. For instance, second car has constant velocity and zero curvature, so the direction does not fluctuate. Based on this approximate model, driver of the first car can predetermine where the second car on the crossroads will be in next seconds and is able to avoid a collision.

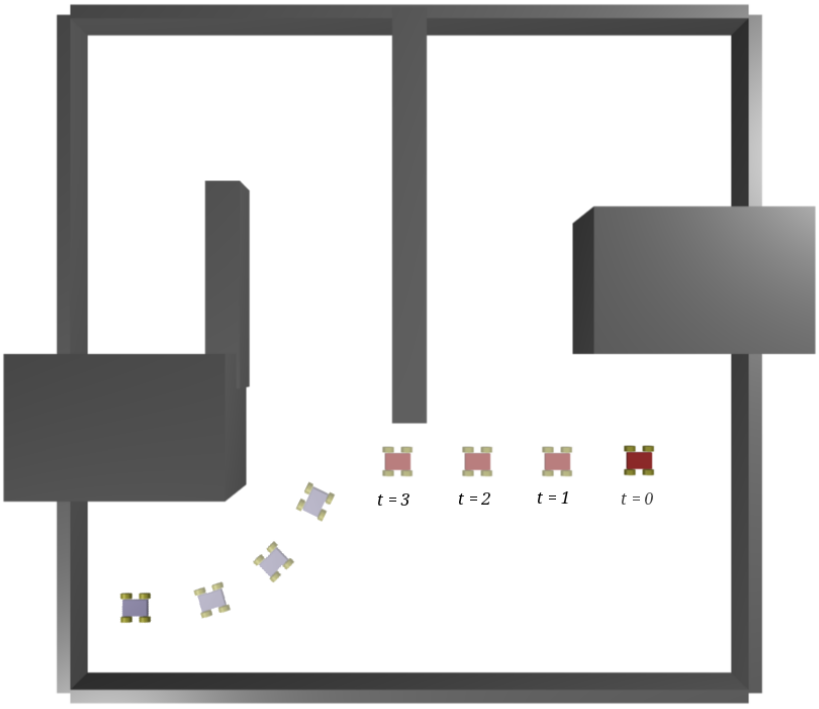


Figure 2: Demonstration of predictive model approach

In this thesis we will be using two types of models of dynamic obstacles: linear and quadratic. These two models are defined by the change of position over time (velocity) and curvature. The properties of the linear model rest in constant velocity and zero curvature while quadratic model is defined by velocity and non-zero curvature. If velocity and curvature are not defined, we will consider the model as static.

1.4 Car-like robot

We can describe a car-like robot concept as a vehicle with four wheels, where the front pair of wheels is able to change angle. In combination with forward velocity, a robot can change a heading like a four-wheeled car. Kinematic model of car-like robot will be described in more detail in Section (2.2.2).
In hardware experiments with the SyRoTek system, substitution of four-wheeled car-like robot for a robot with two actuated wheels will be necessary. Because of this, the change of heading will be converted from angle in radians to the input angular velocity. Experiments with the SyRoTek system will be discussed in Section (4).

2 Implementation

Control and navigation algorithms of formations of car-like robots are not the primary subject of this thesis but it represents the key part by providing a core framework for this project. Brief description of this approach will be described in section (2.2). Detailed explanation of MPC implementation can be found in [1]. The approximate chart is shown in Figure (3).

2.1 Initialization trajectory

Model predictive control as utilized in this thesis works on basis of pushing and optimizing known control points of input predetermined trajectory. This trajectory, which is defined by user or state space searching computer algorithm (RRT[6], visibility graph, Voronoi diagram etc.), need not to be necessarily optimal but it should provide the most feasible solution possible. Imperfection of input trajectory in sense of colliding with walls or other obstacles can cause high amount of local minima in the cost function[7]. Then the process can easily get stuck and robot will not find feasible solution.

2.1.1 Visibility graph

For needs of this project (in sense of initialization trajectory as an input of the optimization), the visibility graph algorithm was chosen. In definition, in robot motion planning, a visibility graph is a graph typically consisted of nodes and edges in the Euclidean plane. Locations of all nodes are defined by the position of edge points of the inflated obstacles in the environment. Only edges with visible connection to each of two connected nodes are valid. Visible connection means that the edge does not intersect with any obstacle in the environment. Computed example of visibility graph for robot motion planning is shown in Figure (4) (blue line).

The shortest path in visibility graph provides implementation of Dijkstra algorithm, which should ensure a prerequisite that the path will be feasible (red line in Figure (4)). The main advantage of choosing this method to initialize the input trajectory for the solver is a short computation time and less control inputs, which results in faster solving of the final trajectory by model predictive control. For purposes of the experiment with real robots on the SyRoTek system, visibility graph appeared sufficient.

[6] Rapidly-exploring Random Tree

[7] The cost function will be explained in Section (2.2.5).

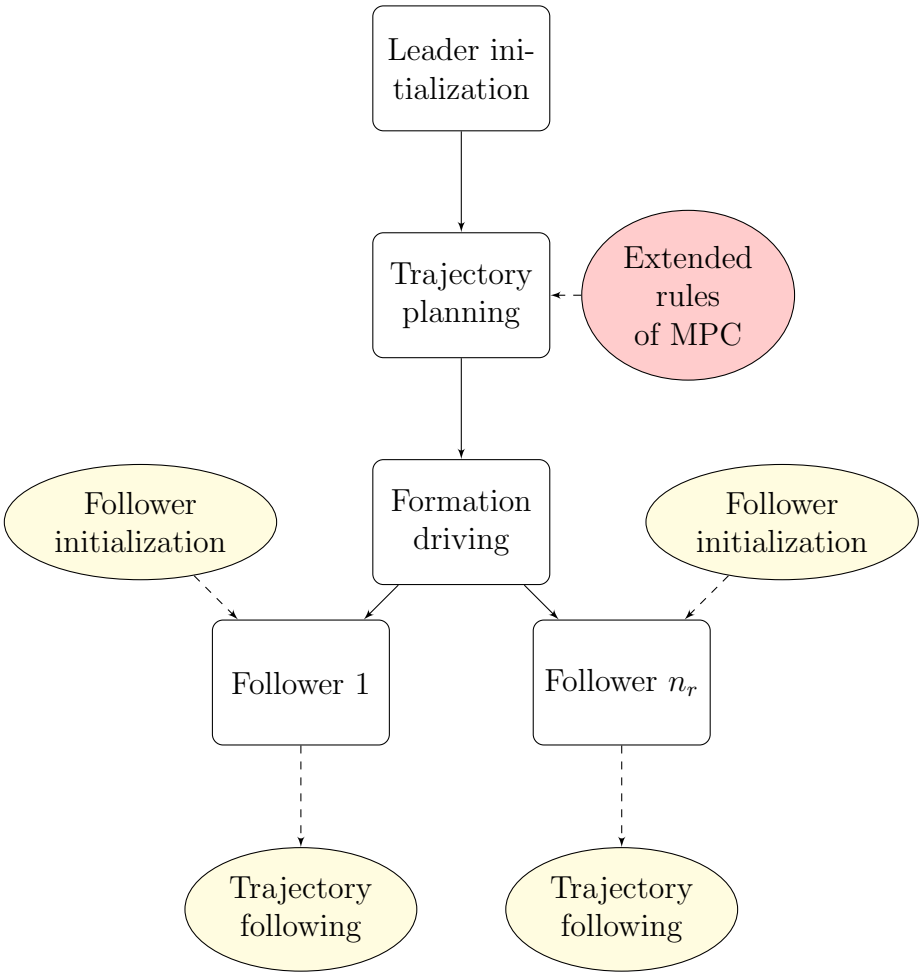


Figure 3: The approximate chart of application modules of model predictive control. The rules extension, which is the main purpose of this thesis, is marked by red bubble.

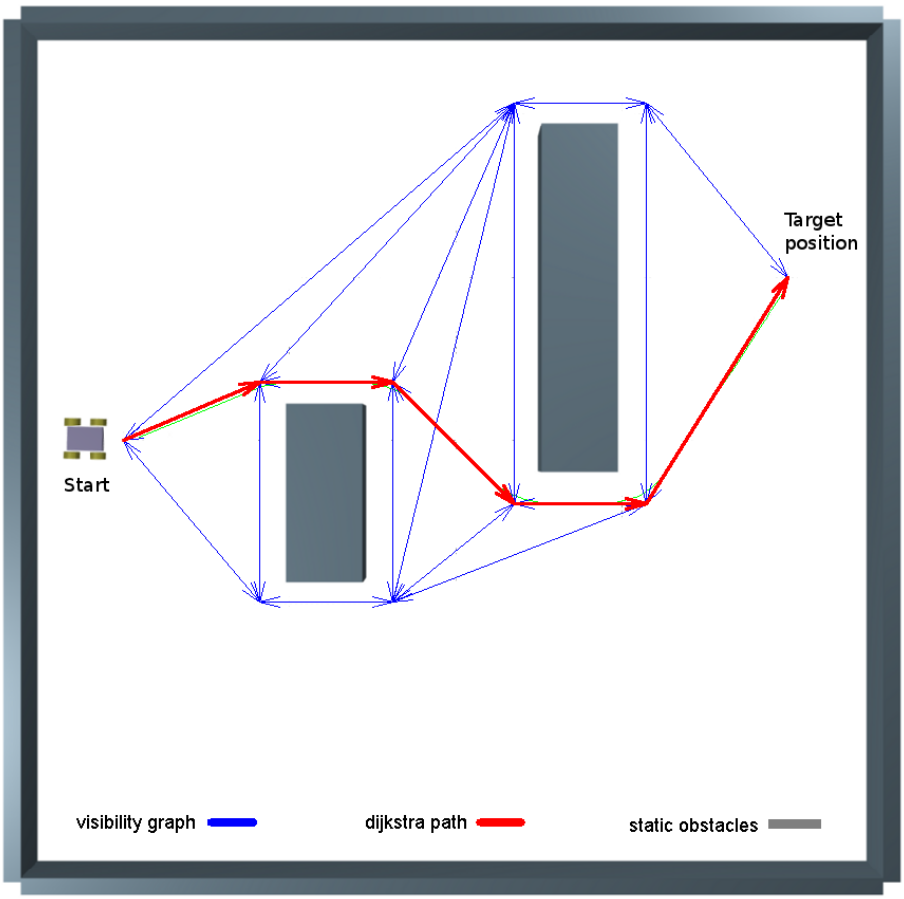


Figure 4: Computer simulation of visibility graph with dijkstra shortest path searching algorithm.

2.2 Model predictive control

As was mentioned in the beginning of Section 2, the theory of model predictive control is not the primary subject of this thesis, but it is need to be summarized the basics of this approach as were described in detail in [1] and [2].

2.2.1 Robot's configuration space

Let $\psi_L(t) = \{x_L(t), y_L(t), \theta_L(t)\} \in C$ denote the configuration of a virtual leader robot $R_L(\psi_L(t))$ and $\psi_i(t) = \{x_i(t), y_i(t), \theta_i(t)\} \in C$, for $i \in \{1, ..., n_r\}$, denote the configuration for each of the n_r follower robots $R_i(\psi_i(t))$ at time t where C is the configuration space[1].

2.2.2 Kinematics and model

In this thesis, we will consider a car-like type of robot as was mentioned in Section (1.4). The actual position of any robot is defined in the Cartesian coordinates $(x_i(t), y_i(t))$ and by heading $\theta_i(t)$. The kinematics for any robots is described by the following kinematic model:

$$\frac{d}{dt}x_j = v_j(t)cos\theta_j(t) \tag{1}$$

$$\frac{d}{dt}y_j = v_j(t)sin\theta_j(t) \tag{2}$$

$$\frac{d}{dt}\theta_j = K_j(t)v_j(t), \tag{3}$$

where curvature K_j is defined by this equation:

$$K_j = \frac{\tan\omega_j}{L} \tag{4}$$

In this case, L is the distance between front and rear wheels of a car-like robot. The ω_j is then angle of front pair of wheels.
If we integrate the model in (1) over interval $< t_o, t_{N+M} >$ with constant control inputs in each time interval $< t_k, t_{k+1} >$, we can obtain model for transition points at which control inputs change:

$$x_j(k+1) = \begin{cases} x_j(k) + \frac{1}{K_j(k+1)}[sin\theta_j(k) + K_j(k+1)v_j(k+1)\Delta t(k+1)) - \\ -sin(\theta_j(k))], \text{if } K_j(k+1) \neq 0; \\ x_j(k) + v_j(k+1)cos(\theta_j(k))\Delta t(k+1), \text{if } K_j(k+1) = 0 \end{cases} \tag{5}$$

$$y_j(k+1) = \begin{cases} y_j(k) - \frac{1}{K_j(k+1)}[cos\theta_j(k) + K_j(k+1)v_j(k+1)\Delta t(k+1)) - \\ -sin(\theta_j(k))], \text{if } K_j(k+1) \neq 0; \\ y_j(k) + v_j(k+1)sin(\theta_j(k))\Delta t(k+1), \text{if } K_j(k+1) = 0 \end{cases} \tag{6}$$

$$\theta(k+1) = \theta_j(k) + K_j(k+1)v_j(k+1)\Delta t(k+1), \tag{7}$$

where $x_j(k)$, $y_j(k)$ are the rectangular coordinates and $\theta_j(k)$ is the heading angle. This model notation is useful for describing long trajectories exactly using a minimal amount of information. More detailed explanation can be found in [1].

2.2.3 Constraints

In MPC theory, the Cartesian coordinate system is used to determine the actual position of any robot as was mentioned in the previous paragraph. But the main problem lies in determining the change of position over time, where the Cartesian system is inappropriate because in the real experiments we must respect hardware propositions of the robot. The wheels of a car-like robot are limited by maximal and minimal curvature as well as the engine has constrained power to actuate the wheels - this parameter we will take in consideration by defining velocity. These constraints can be expressed by the following inequalities:

$$v_{min,j} \leq v_j(k) \leq v_{max,j} \tag{8}$$

$$|K_j(k)| \leq K_{max,j}, \tag{9}$$

where $v_{max,j}$ is the maximal forward velocity of the car-like robot, $v_{min,j}$ is the maximal backward velocity and $K_{max,j}$ is the maximal possible curvature.

2.2.4 Shape of a formation

The major problem during the computation of a new control point of any formation lies in the constrained curvatures as was explained in Section (2.2.3). When formation is turning, every robot has to execute a movement with different value of curvature. If input curvature to outer follower of a turning formation will be too small, limited range of inner followers will not be enough and collision becomes imminent.
Another very important fact when driving formation of car-like robots is caused by the

inability to the change angle of heading of the robot on the spot. This problem was solved by implementation of curvilinear coordinates instead of the Cartesian, which cannot be used. To convert the state of followers in curvilinear coordinates to the state of rectangular coordinates, the following equation can be applied:

$$x_i(t) = x_L(t_{p_i(t)}) - q_i(t_{p_i(t)})sin(\theta_L(t_{p_i(t)})) \tag{10}$$
$$x_i(t) = x_L(t_{p_i(t)}) - q_i(t_{p_i(t)})sin(\theta_L(t_{p_i(t)})) \tag{11}$$
$$\theta_i(t) = \theta_L(t_{p_i(t)}) \tag{12}$$

2.2.5 The cost function

The basic equation in the model predictive control theory is the minimization of the cost function $J_L(.)$,which is given by (13).

$$J_L(\Omega_L) = \sum_{k=N+1}^{N+M} \Delta t(k) + \alpha \sum_{j=1}^{n_0} \left(min \left\{ 0, \frac{dist_j(\Omega_L, \Theta_{Obs}) - r_{s,L}}{dist_j(\Omega_L, \Theta_{Obs}) - r_{a,L}} \right\} \right)^2, \tag{13}$$

where the first part of the formula expresses endeavor to reach a desired target as soon as possible and the second part accounts for expressing the influence of the environment. This influence has an impact on the final solution and we are able to control it's cost by adjusting the α constant.

2.3 Implemented models

2.3.1 Rules extension for dynamic obstacles

The cost function in Section (13) is defined correctly for the leader only, but for the whole formation we need to have an extended variant, which is composed of three components with their influence adjusted by constants α and β in (14). The first component represents deviations of the desired position in the same way as in the previous equation. The second summation term is also the same as is mentioned in (13) and is responsible for detection of static or lately detected dynamic obstacles. The main difference is in the third component with the β constant, which has been added to ensure avoiding collision between other members of the formation by considering them as other dynamic obstacles in the environment.

The main contribution of this thesis was the extension of model predictive control framework by dynamic obstacle avoidance. These rules were implemented into the second component of equation (14), where the position of obstacles dynamically changes with respect to time. It means that the solver can predict the next move of an obstacle in each iteration according to the input model.

$$J_i(\Omega_i) = \sum_{k=1}^{N} \parallel (\bar{p}_{d,i} - \bar{p}_i(k)) \parallel^2 + \alpha \sum_{j=1}^{n_0} \left(min \left\{ 0, \frac{dist_j(\Omega_i, \Theta_{Obs}(t)) - r_s}{dist_j(\Omega_i, \Theta_{Obs}(t)) - r_a} \right\} \right)^2 + \beta \sum_{j \in \bar{n}_n} \left(min \left\{ 0, \frac{d_{i,j}(\Omega_i, \Omega_j^O) - r_{s,i}}{d_{i,j}(\Omega_i, \Omega_j^O) - r_{a,i}} \right\} \right)^2, \tag{14}$$

where r_s is a circular detection boundary radius, r_a is a circular avoidance boundary radius for a single robot, Ω is a position of a robot and Θ is a position of an obstacle.

2.3.2 Linear model interpretation

In this subsection, we will introduce mathematical interpretation of the linear model avoidance predictive method. The concept of obstacle movement is very similar to the model of motion which was defined for robot leaders in Section 2.2. The main difference lies within input velocity, which remains constant during the whole computation. In the case of linear model, we do not take curvature K in consideration. Actual position of virtual dynamic obstacle is defined by summation of partial changes over time by equations shown in (15). Initial position of dynamic obstacle is given by coordinates $(x_{lm_0}, y_{lm_0}, \theta_{lm_0})$, where the θ_{lm_0} remains constant. Actual position of a virtual marauder is given by $(x_{lm}(k), y_{lm}(k))$, where k is the local transition point.

$$x_{lm}(k_{n+1}) = x_{lm_0} + \sum_{k=1}^{n} \Delta x_{lm}(k_n) \tag{15}$$

$$y_{lm}(k_{n+1}) = y_{lm_0} + \sum_{k=1}^{n} \Delta y_{lm}(k_n). \tag{16}$$

where the actual Euclidean coordinates of position of robot marauder (dynamic obstacle) moving with straight heading and constant velocity is defined by the following equations:

$$\Delta x_{lm}(k) = v_{lm} \sin(\theta_{lm}) \Delta t \tag{17}$$

$$\Delta y_{lm}(k) = v_{lm} \sin(\theta_{lm}) \Delta t \tag{18}$$

$$\theta_{lm}(k) = \theta_{lm_0}, \tag{19}$$

2.3.3 Quadratic model interpretation

The same concept as was described in Section 2.3.2 we can use in the case of interpretation of quadratic model but the input curvature K_{qm} must be taken in account. Actual state of virtual dynamic obstacle with initialization coordinates $(x_{qm_0}, y_{qm_0}, \theta_{qm_0})$ is defined by the following equations:

$$x_{qm}(k_{n+1}) = x_{qm_0} + \sum_{k=1}^{n} \Delta x_{qm}(k_n) \tag{20}$$

$$y_{qm}(k_{n+1}) = y_{qm_0} + \sum_{k=1}^{n} \Delta y_{qm}(k_n) \tag{21}$$

$$\theta_{qm}(k_{n+1}) = \theta_{qm_0} + \sum_{k=1}^{n} \Delta \theta_{qm}(k_n), \tag{22}$$

where the Euclidean coordinates of position of robot marauder (dynamic obstacle) moving with straight heading and constant velocity is defined by the following equations:

$$\Delta x_{qm}(k) = \frac{\sin(\theta_{qm}) + K_{qm} v_{qm} \Delta t - \sin(\theta_{qm})}{K_{qm}} \tag{23}$$

$$\Delta y_{qm}(k) = -\frac{\cos(\theta_{qm}) + K_{qm} v_{qm} \Delta t - \sin(\theta_{qm})}{K_{qm}} \tag{24}$$

$$\Delta \theta_{qm}(k) = K_{qm} v_{qm} \Delta t, \tag{25}$$

2.3.4 Implementation details

Inside the fitness function (14), the measure of an actual distance between marauder and robot leader is being executed in each iteration of a solver through the Euclidean metric, which is shown in Equation (26):

$$dist(\Omega_L, \Theta_{Mau}) = \sqrt{\sum_{n=1}^{3} (\Omega_{L_i} - \Theta_{Mau_i})}, \tag{26}$$

where n represents a system of coordinates (x, y, z) of a robot, Ω_L defines an actual position of leader and Θ_{Mau} is the position of a virtual marauder. A brief approximation of extended algorithm of model predictive control approach is shown in the Scenario 1:

→ description **getVisibilityGraph**;
– Calling of visibility graph algorithm, which is responsible for creating initializing control inputs from the actual state of the world.
→ description **callSolver**;
– CallSolver is the core function of the whole algorithm, which on the basis of control inputs, constraints and the final value of fitness function evaluates the quality of solution. Also, this function can iteratively measure the Euclidean distance of marauder and the rest of the formation as was described in equation (26).

```
Algorithm;
world = loadMap();
control = getVisibilityGraph(world);
leader → init;
followers → init;
marauder → init;
while dist2goal > 0 do
    callSolver(state, end, N, control, constants);
    leader → move(solution → velocity, solution → curvature, solution → time);
    insertState(state, leaderPath);
    for i = 0; i < followers; i + 1 do
        formation_i → callSolver(state, end, N, control, constants);
        insertState(state, followerPath_i);
    marauder → move(predictiveModel);
    // on the real robotic platform (SyRoTek), here would be a feedback
    controller, which would be taken in account an actual position of a
    marauder;
    updateWorld(world);
    control = getVisibilityGraph(world);
Robot reached goal position;
```

Scenario 1: The algorithm for trajectory planning using MPC approach with new rules for avoiding dynamic obstacles implemented.

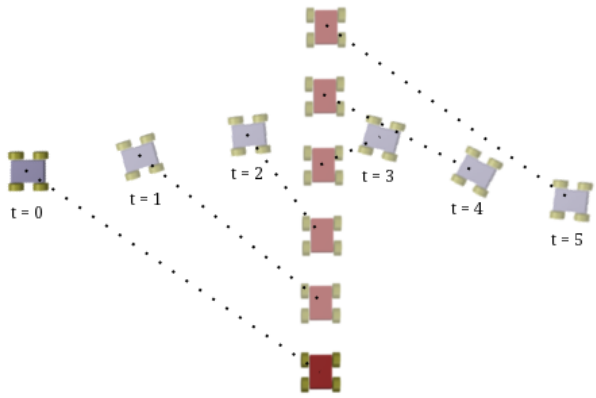


Figure 5: The demonstration of measuring Euclidean distance between marauder and leader.

2.4 Complicated maneuvers

The implemented method of avoiding dynamic obstacles works on a predictive base, but situations in which formation gets stuck can appear. This can be caused by incorrectly tuned predictive model or inaccurate movement of the real robot as a result of bad hardware propositions.

In this case, solution is valid in the sense of the input model, but the detected position[8] of the dynamic obstacle does not correspond. This sets out the problem where dynamic obstacle is moving while the robot leader is executing a new trajectory so the found solution is no longer acceptable and it has to be resolved according to new data. The quality of solving this situation can be enhanced by model where the robot is continuously reversing along the previous trajectory until it is in safe predefined minimal distance from the obstacle. Boundary for this precaution can be set by the constant that is defining the minimal safe distance to the obstacle. The robot then must repeat the process in a loop until the final solution is feasible. This algorithm can guarantee that the future solution will be feasible (assuming the obstacle remains at constant velocity and direction).

This process and final correctness of solution may be very sensitive to tuning up the constants such as sensor sensitivity or distance that is considered as critical in the sense of hardware and physical robot propositions. If the critical distance is tuned up badly, new solution may not be feasible or even solvable. A flowchart of the precaution algorithm can be found in Figure (6).

[8]From the distance sensor or global camera positioning system.

2.4.1 Flow chart

In practice, the algorithm detects the dynamic obstacle is receding in the sensor radius of the robot leader. Robot will continue by small steps forward as long as the dynamic object is detectable on sensor. At the moment when the output data from sensor are clear, robot leader will follow it's feasible trajectory as long as it is necessary to reach the end position which is defined in the target radius.

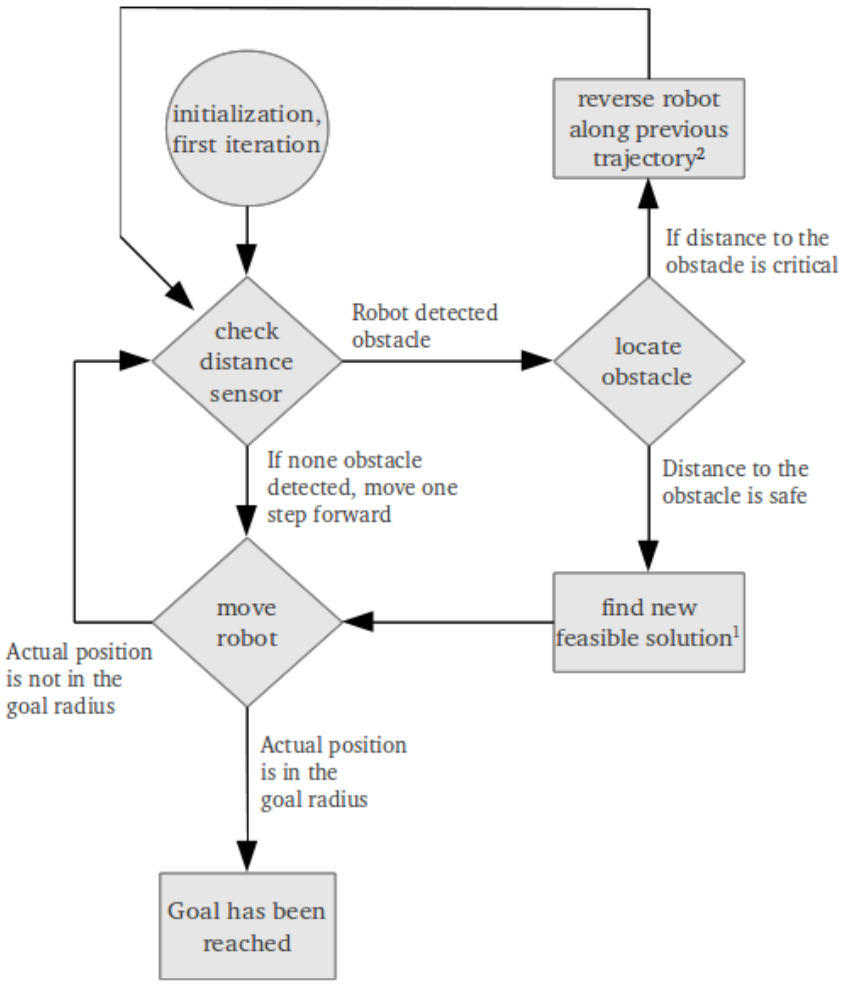


Figure 6: Flowchart of the algorithm which is responsible for avoiding collisions in non-predicated situations.

3 Simulations and results

The final results of this project are summarized in this chapter which consists of detailed report of the experimental mission simulation executed by formation with car-like robot leader using predictive control algorithm for avoiding the dynamic obstacles.

3.1 Computer simulations

Following types of obstacle model movement (3.1.1 and 3.1.2) were implemented into the model-predictive control framework, which is being developed at the Department of Cybernetics, CTU in Prague. As was mentioned in the Section (1.3), two ways of model prediction were integrated. Results of experiments of both implemented models can be found in Section (3.3). In the simulations, two models of dynamic obstacles were tested according to following parameters:

- Linear model - constant velocity, curvature $k = 0$
- Quadratic model - constant velocity, curvature $k > 0$

All simulations and graphics in this section are generated in Gnuplot[9].

3.1.1 Linear model

In this subsection, we are presenting an example of solved solution of solver with extended rules of predictive control in the environment with dynamic obstacle. Following figures demonstrates numeric results for the concrete conditions with respect to the actual time.

In the Figure 7 we can observe the beginning of the simulated mission, where the three blue vehicles represent robotic formation, that are supposed to achieve target ring by feasible trajectory. Red vehicle represents marauder (or broken) robot, that is moving straight down through the optimal trajectory, which is marked by full black line. The second image in the Figure 7 demonstrated critical part of a solution, where the driven formation slowed down[10] and turned to the left to avoid collision with the red car.

The second Figure 7 demonstrates the final part of the solution, where the red car is receding and the formation is starting to balance trajectory to be optimal.

[9]Command-line driven graphic utility for Linux.
[10]More static-time control points on a shorter distance

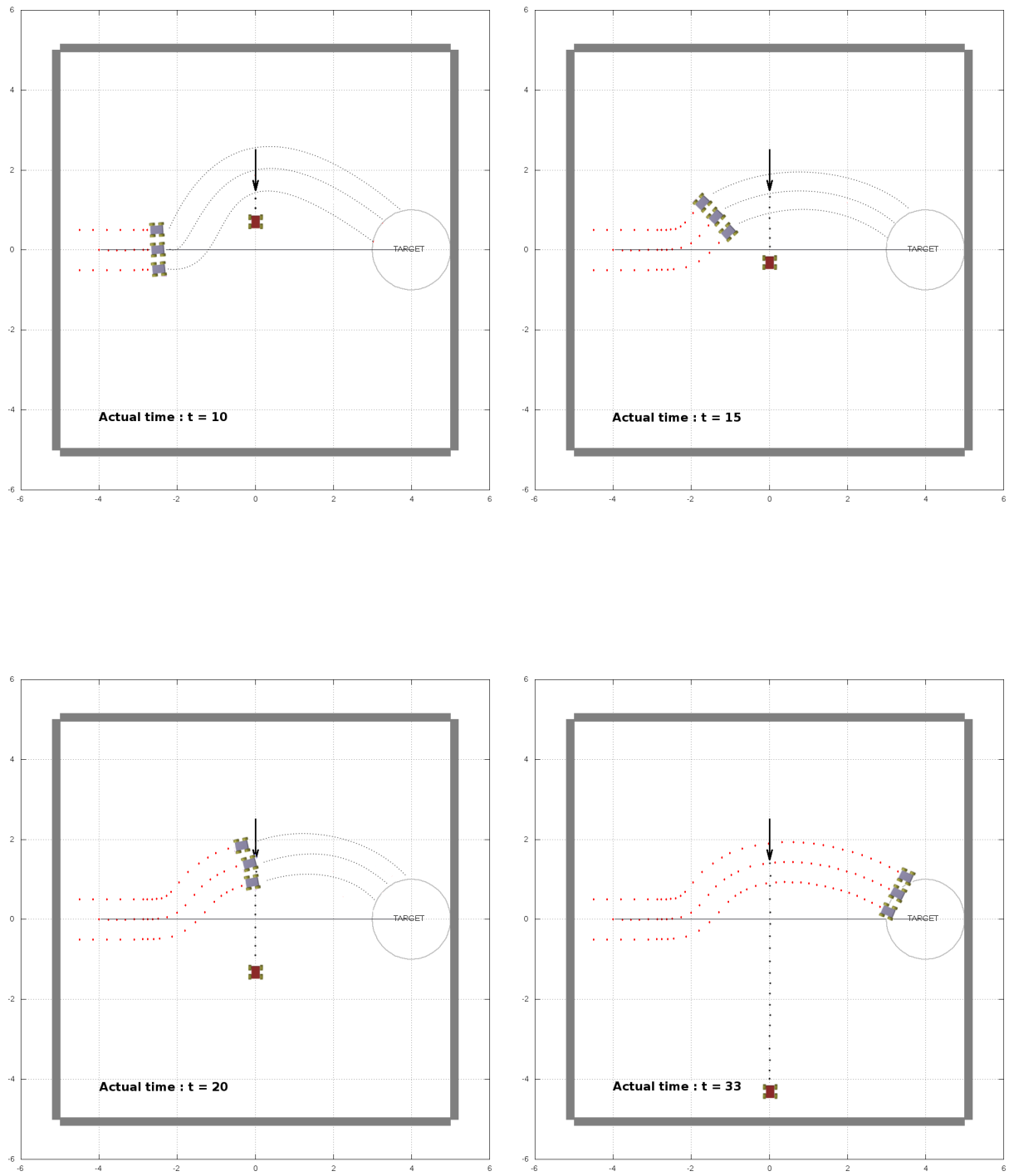


Figure 7: Computer demonstration of a feasible solution solved by linear model of extended rules of model predictive control.

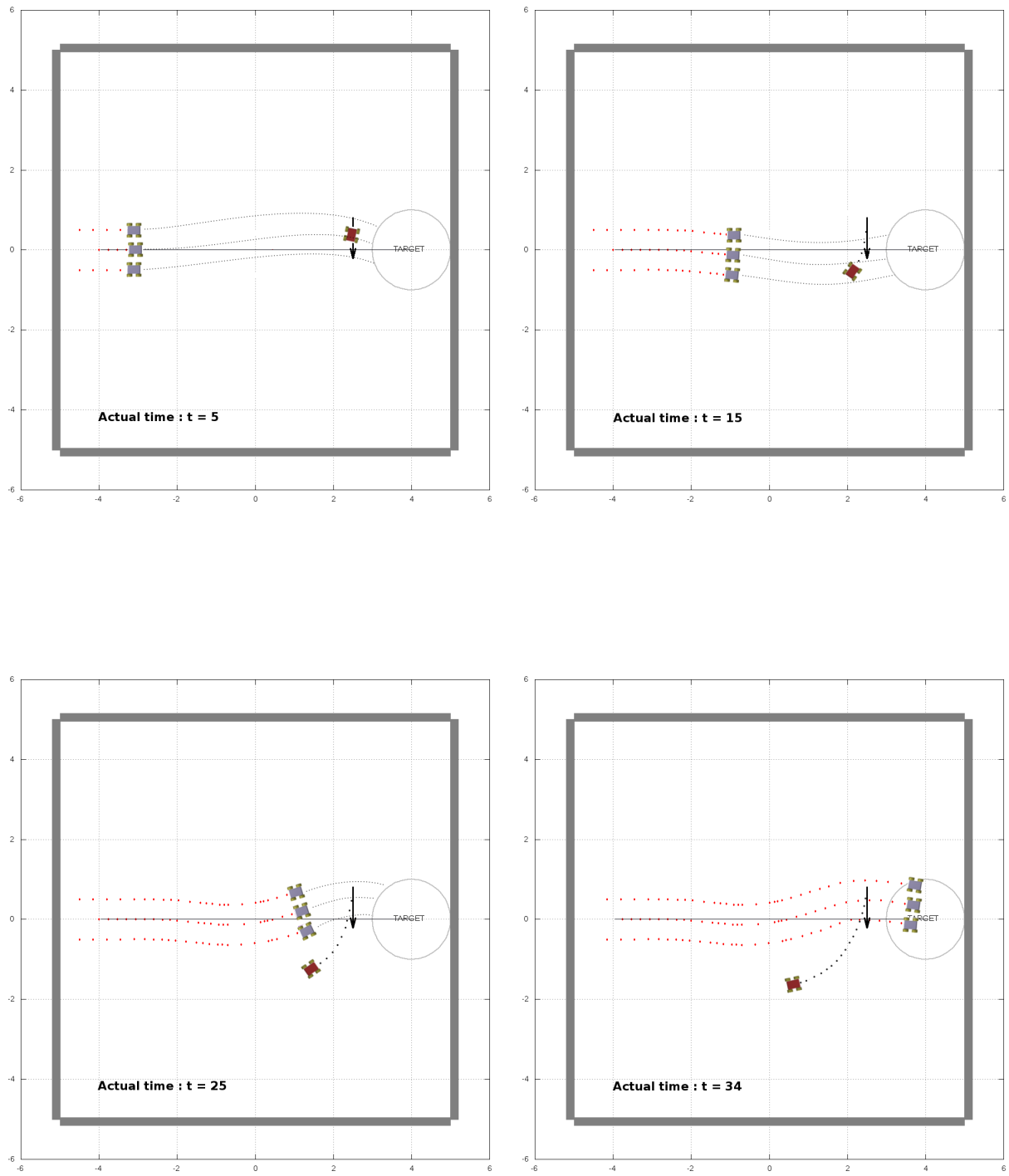


Figure 8: Computer demonstration of a feasible solution solved by quadratic model of extended rules of model predictive control.

3.1.2 Quadratic model

Example of movement of the formation using the algorithm with quadratic model of the obstacle employed for obstacle avoidance is presented in the figure 8. This time, red car is following a circular path.

Quadratic model is characterized by constant velocity and curvature. Resulting movement of this setting can be seen in the following figures, where the red car is moving on a circle trajectory. New rules for model predictive control should provide a safe and feasible solution in sense of predictive avoiding as is demonstrated in the picture by curved trajectory of a blue robotic formation in the figure 8. Higher density of control points indicates an effort to slow down before turning and to increase a distance between the car-like robot formation and the the red robot.

3.2 Quality of solution

The quality of the final solution can be compared among two implemented models by two different criteria: final times which will show how quickly the formation can achieve target region and the minimal distance to collision from formation that is critical for feasibility of the final solution. In dynamic environment, bad tuned predictive model (or none model) can easily get into collision state with a moving obstacle. In the Figure 9, the comparison of final trajectories of two solvers with different predictive model implemented for the same initialization conditions can be seen.

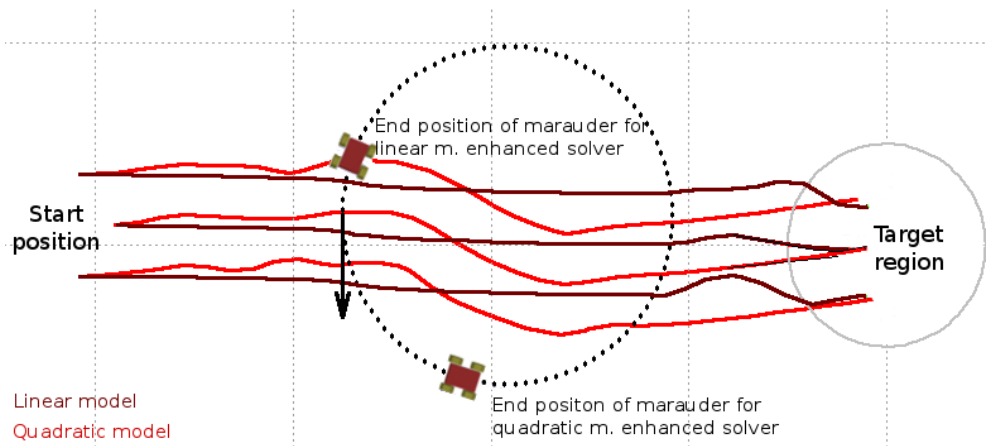


Figure 9: Comparison between linear and quadratic solver for the same initialization conditions.

This computer simulation (as is shown in the Figure 9 was executed for dynamic obstacle (robot marauder) that is moving according to the quadratic model. In case of the employed linear model, robot formation does not account for curvature of a robot and after each

iteration solver considers the heading of marauder as static. This can cause the situation as the one shown in the Figure 10, where the formation is in a direct collision with robot marauder while solver with an accurate quadratic model implemented found a feasible trajectory. Actual position of robot marauder (as is shown in 9) show that the solver with linear predictive model implemented ended in shorter time than with quadratic prediction but solved infeasible trajectory 10.

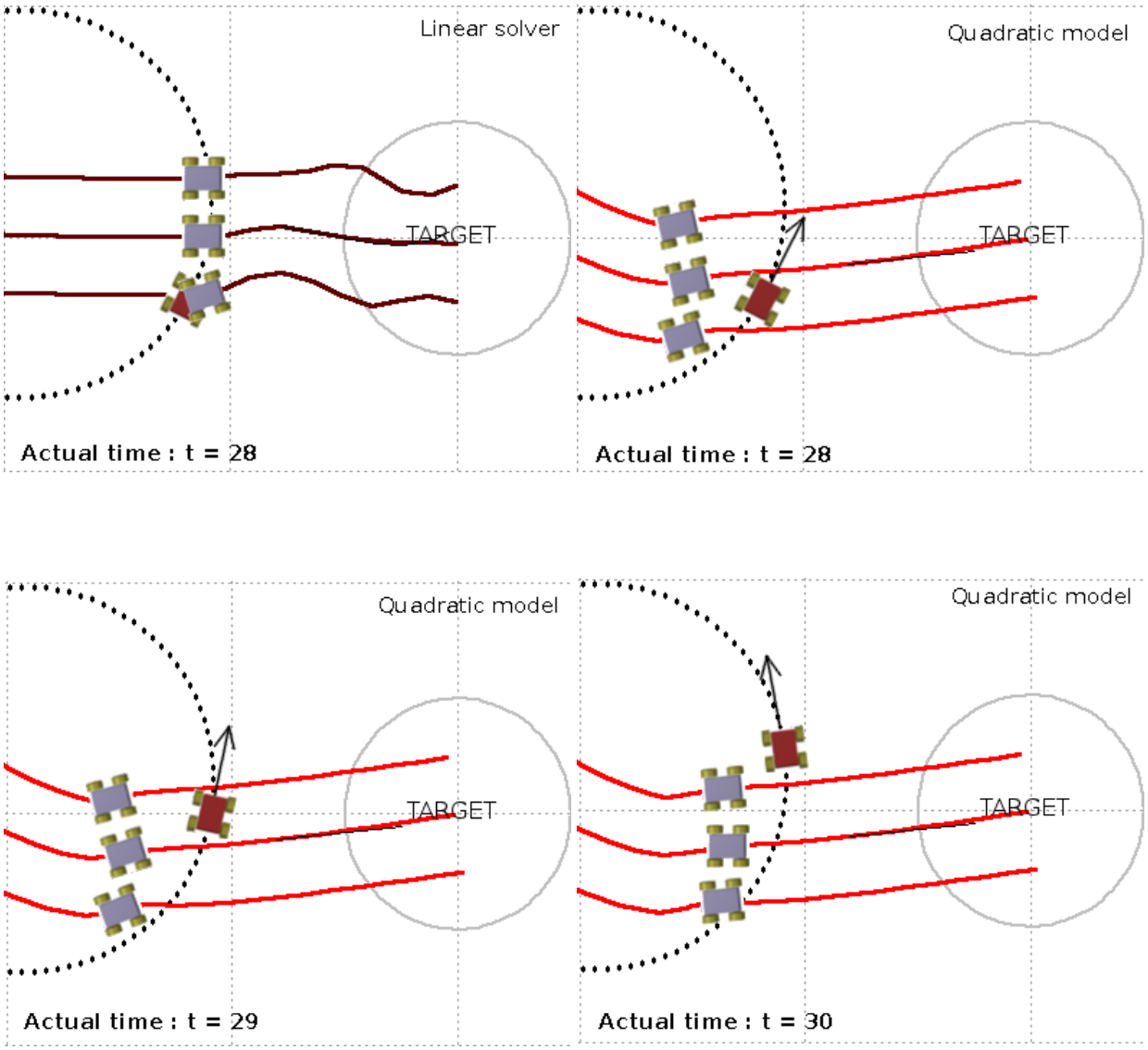


Figure 10: Demonstration of collision state of a badly tuned predictive model.

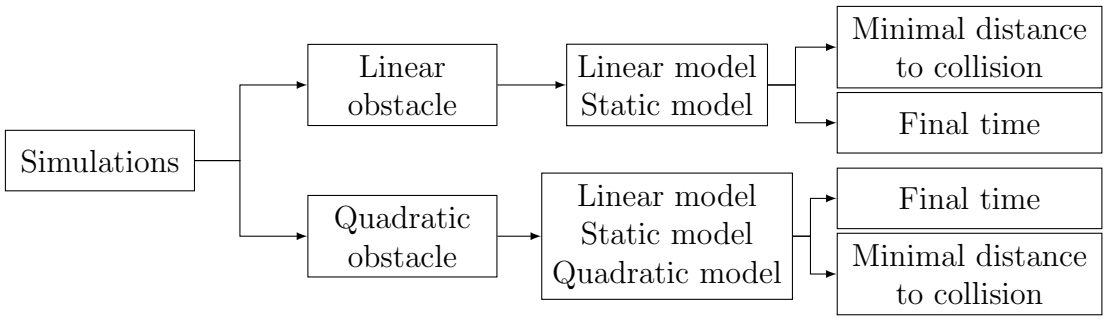
3.3 Results and graphs

To verify implementation of new rules of model predictive control for driving formation in environment with dynamic obstacles, several computer simulations were executed, that will compare non-predictive rules of MPC with new extensions to predicate non-static barriers, as was described in Section 2.3.1.

These two models will be compared in several situations by following criteria:

- the minimal distance from the formation to collision state with the obstacle
- the final time of the solved trajectory

Both test will be playing a crucial role in the resulting quality of new implemented rules for driving robotic formations. Tests in 2.3.1 are divided into two independent parts with different initial conditions. The first run of experiments was executed with linear model of dynamic obstacle movement, while the second one was set to simulate quadratic model. The comparison was made by the following configuration:



For objectivity of the final numbers, in sum 2300 experiments were executed. All results were entered into a bar graph. In each simulation, randomly generated start position of formation was chosen.

3.3.1 Static vs. linear model

The first executed simulations demonstrate a comparison between static and linear model prediction of dynamic obstacle movement. In the Figure 11 we can see an average time to achieve final position of both model comparison. According to the figure, in this series of simulations, implementation of linear predictive model improved solution and shortened the final time for which the formation traveled on solved trajectory.

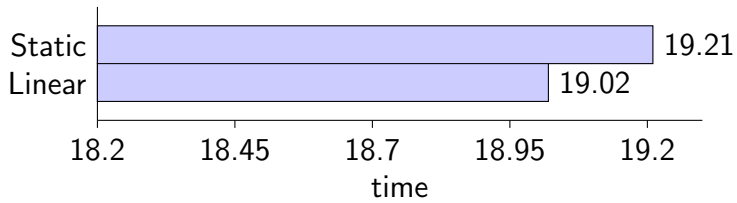


Figure 11: Graphic comparison of static and linear model for linear movement of the dynamic obstacle (in sense of final time).

On the contrary, difference between measuring of the minimal distance to collision of both models are non-significant in this case. Figure 12 demonstrates that even well tuned model can not necessarily improve the final solution but also does not make it worse.

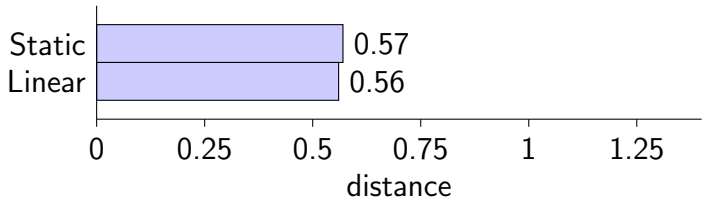


Figure 12: Graphic comparison of static and linear model for linear movement of the dynamic obstacle (in sense of minimal distance to collision).

3.3.2 All models comparison

Much more illustrative is comparison of all models for quadratic movement of the dynamic obstacle. As we mentioned in Section (3.3.1), well tuned model should not make the final solution worse then solver without any model (static). But if we use incorrectly tuned model, results can easily become worse or even strongly inappropriate (because of infeasible trajectory). In this series of simulation, we tested dynamic solver in situation with dynamic obstacle, which is following a circle trajectory (constant velocity, $k > 0$), for both implemented predictive solver variants. In this case, the linear model behaves as badly tuned quadratic model (zero curvature), which results in a much worse solutions then the solver without any model. This find sets out a conditions, where it is necessary to use as good model as possible to achieve the best resulting trajectory.

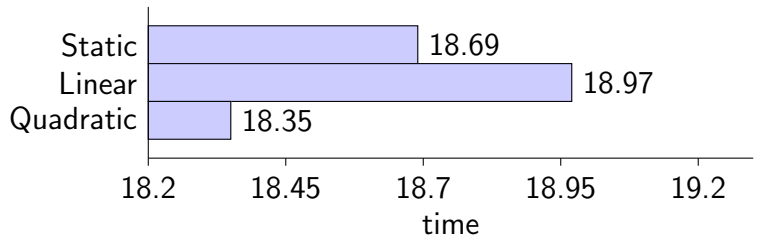


Figure 13: Graphic comparison of all models for quadratic movement of the dynamic obstacle (in sense of final time).

Measuring of minimal distance from formation to collision with the dynamic obstacle also brought an evidence that well tuned model improves the final solution while bad model (linear) can make the final solution make worse. In the Figure (14), solution by quadratic predictive model showed significant improvement.

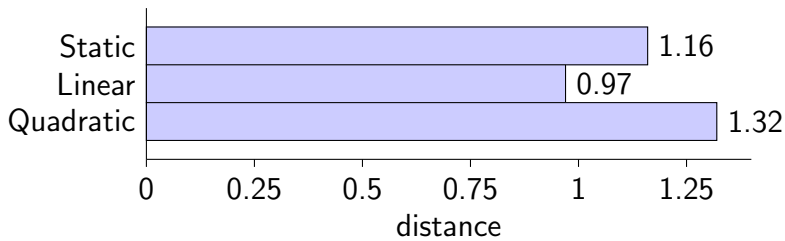


Figure 14: Graphic comparison of all models for quadratic movement of the dynamic obstacle (in sense of minimal distance to collision).

4 Experiments with SyRoTek

To verify all new implemented rules of model-predictive control framework, the SyRoTek[11] system was chosen as an appropriate real simulator.

Figure 15: One of twelve robots, that are forming an active part of a SyRoTek robotic system.

4.1 About SyRoTek

The SyRoTek system is a result of a project of the Czech Technical University in Prague, that allows students and scientists to develop their own algorithms on a multi-robot platform in a dynamic environment. System enables users to remotely control the whole platform without any external human intervention, because the maintenance is supposed to be fully autonomous.

Communication interface between users and robots is provided by the Player Stage, where Player is designed to provide all necessary IP communication between robots, server and users. Stage is an optional part of the software and enables to test and simulate algorithms before uploading to the real system.

Whole system consists of a twelve autonomous robots equipped with several sensors to merge distance, scan surface and detection of an actual orientation of a robot. In practice, most of all robots are fitted up with eight IR range sensors (five on the chassis and three in

[11] System for robotic e-learning

the front), six sonars(three on the chassis and three in the front), compass, accelerometer, two floor sensors and laser range finder. In the near future, all robots will also be equipped with cameras. Actual position of any robot in the arena can be detected through the on board odometry sensors or by the global camera positioning system, that is much more precise for long-time measuring.

Implementation of all algorithms is realizing in C++ programming language with an appropriate libraries included. Source codes are then compiled and executed by Player on the SyRoTek server. More information in detail can be found in [9].

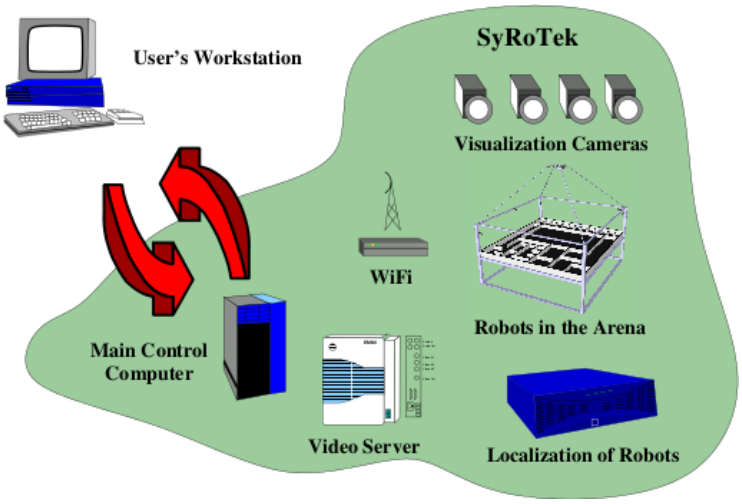


Figure 16: Communication scheme of the SyRoTek system.

4.2 Computer simulation

Before executing the real experiment on the SyRoTek system, all initializing conditions and functionality of the solver with new rules implemented were tested and analyzed through the virtual computer simulation as it's showed in Figure (17). As dynamic obstacle, one of the SyRoTek robots was chosen to move according to quadratic model. Solver was tuned with very accurate input parameters for the best possible solution. Robot hardware propositions such as dimensions or velocity limits were also taken into account. As initialization trajectory (tiny black line in the Figure (17)), visibility graph with Dijkstra path state search algorithm was chosen. To ensure the most feasible input trajectory, visibility graph is generated dynamically in each iteration of computation of MPC solver.

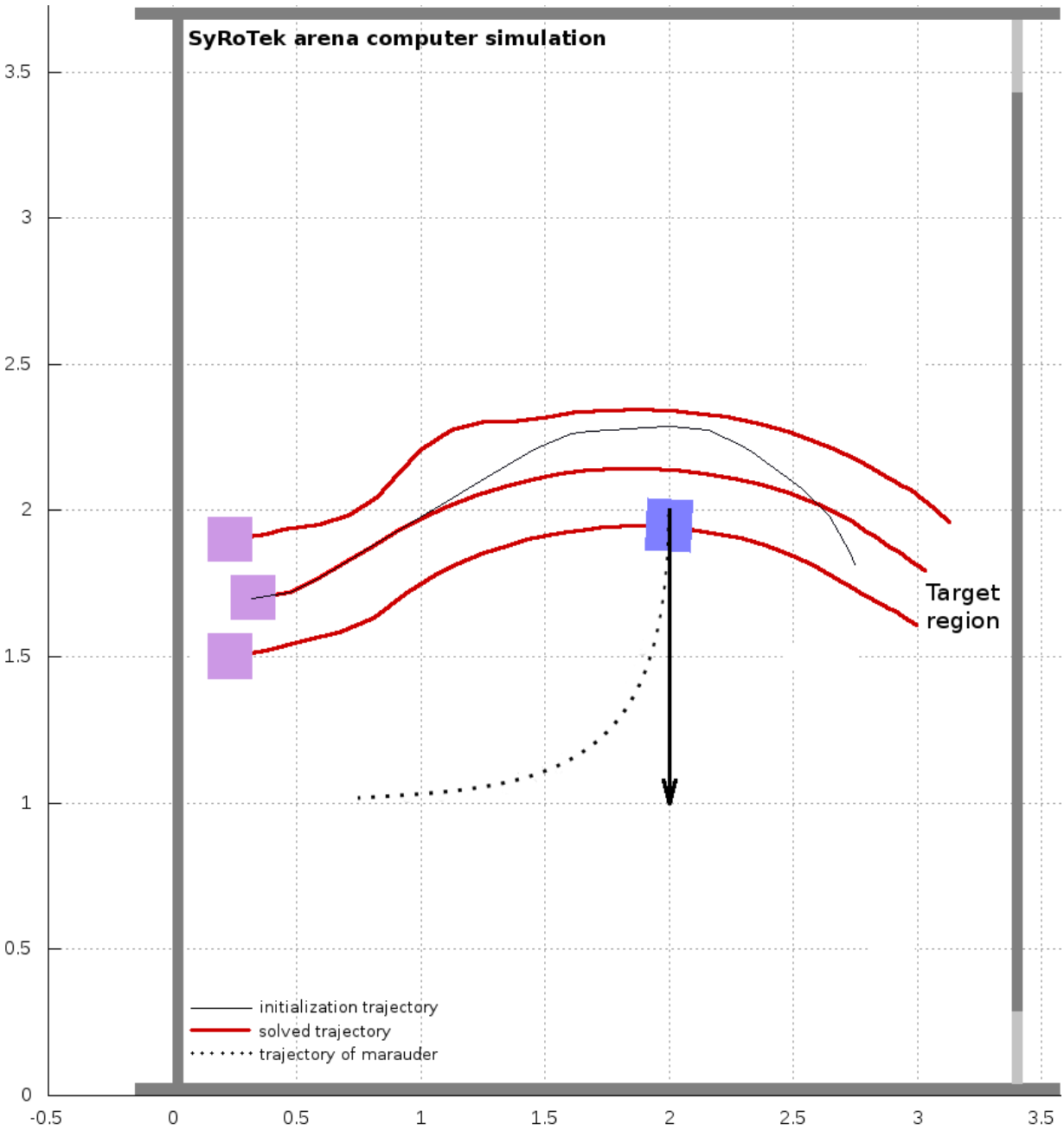


Figure 17: Computer simulation of experiment in the SyRoTek system arena

4.3 Experiments

4.3.1 Initialization conditions

In this thesis, practical examples on a real robotic system of the new rules of model predictive control theory, were executed. For purposes of the experiments, static obstacles in the arena were removed and localization system was properly modified. Tested formation consisted of three robots - one robot leader and two followers. As a dynamic obstacle, another robot was chosen to move across the optimal trajectory of the formation in which way the solver was forced to change the shape of path and ensure to be feasible for all members of the formation.

Experimental robotic mission was set up to achieve target region in cleared and empty robot arena. Without intervention of dynamic obstacle, an optimal trajectory leads straight from the start position. More complicated scenario could not be chosen, because of the limited space in the arena.

4.3.2 Implementation details

As was mentioned in Section 4.1, all algorithms are implemented using C++ language. The source is structured into two main sections. The first part provides communication with Player to drive robots, while the second part counts the trajectory (solver of model predictive control) on the base of input data from sensors. For each robot in the arena, individual thread was implemented. All threads must be synchronized to ensure, that all robots will be following the formation.

Final experiment was divided to offline and online computation. The main reason for this separation is significant computation time of new control point of the MPC solver. In practice, after each step, all robots had to stop and wait for solver to finish all calculations. Because of this deceleration, an offline version of experiment was executed to demonstrate fluent movement of the formation. To prevent anomaly hardware problems such as slipping wheels, an appropriate feedback controller was programmed. Both runs are documented and attached as video record on CD for this thesis.

4.3.3 Mission

During both experiments, robots successfully followed a trajectory computed by the solver. Noticeable anomalies caused by hardware were regulated through implemented feedback controller. In the figures 18 and 19 recorded situations can be seen during the experiments with graphical demonstration of executed trajectory that was calculated by the solver of model predictive control.

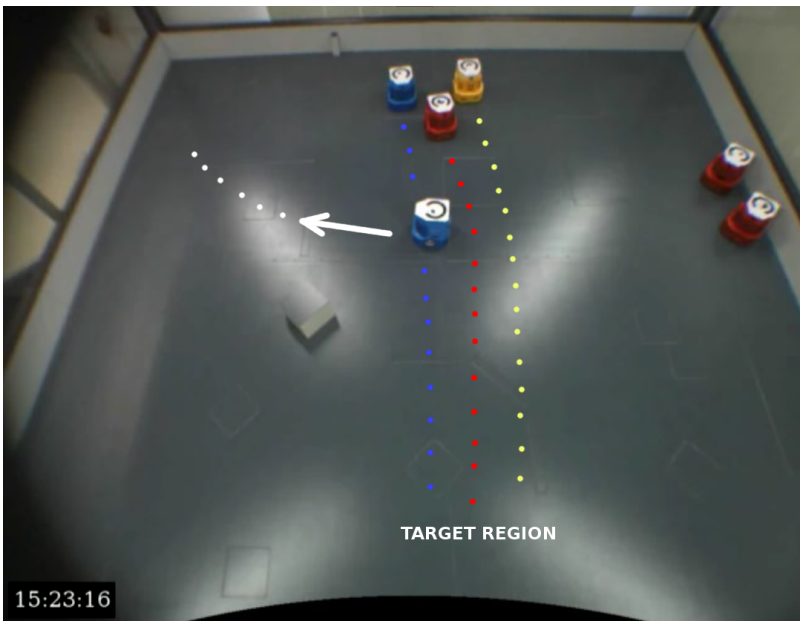


Figure 18: Camera record of SyRoTek system during the experiment.

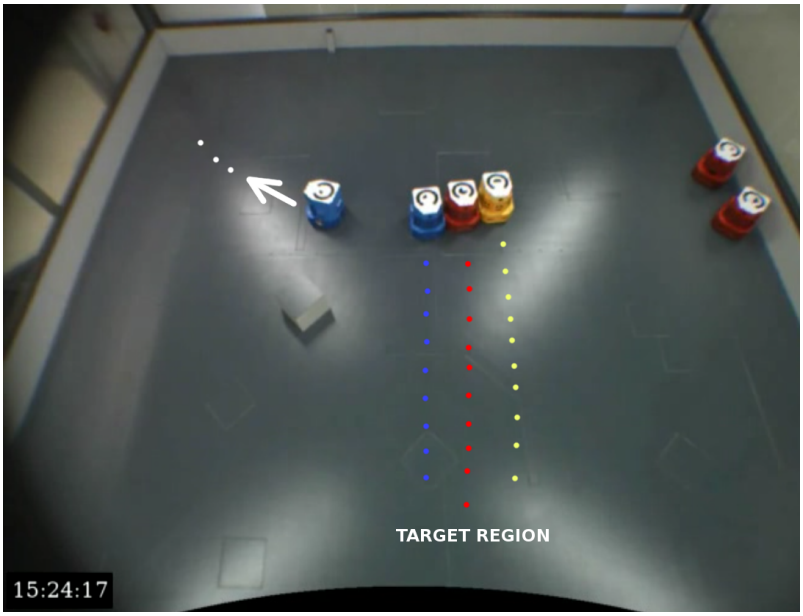


Figure 19: Camera record of SyRoTek system during the experiment.

4.4 Experiences with system

Although the SyRoTek system is almost at the final state of development, there is still a lot of space for further improvements. In this thesis, work with real robotic system and implementation of model predictive control algorithms into the SyRoTek system were one of the most important issues in this project. Based on this experiences, several recommendations will be brought to help with further improvements of the system.

4.4.1 Recommendations

At the beginning of this bachelor project, working with the SyRoTek system was very difficult and impractical. Problems started with official SyRoTek user manual which in detail described only special plugin for NetBeans IDE. But this plugin worked properly only on the specialized SyRoTek linux distribution, which was very slow and not suitable for working with larger project. The only usable form of controlling and programming the system was through the linux terminal. Unfortunately, this way of using SyRoTek would deserve much more complex documentation, because the only information about using Player platform was located on the official website with a lot of redundant data (redundant for system users, not developers).

Next issue which should be mentioned was programming the system. Like documentation, SyRoTek system would deserve more examples with multirobotic use. Services such as integrated dynamic obstacles, camera access, manual robot driving or using syrcontrol[12] were described nowhere. But this lack of information was apparently caused by the fact that this thesis was the first of its kind to work with the SyRoTek system.

The SyRoTek system is designed for 24 hours remote access and should provide an autonomous service, reservations management, docking robots and choosing an appropriate robots for the individual reservation. But there are also issues that system cannot solve properly. One of them is cleaning dust from the surface of the arena, that may cause slipping wheels of the robots. To solve this problem, a cleaning robot should by constructed. But the problem with sliding wheels is not only caused by dust. Next problem lies within the mechanical system for ejecting integrated dynamic obstacles. The system sometimes slides the obstacle lower then the surface of the arena, where a robot can easily get stuck.

Another problem that occurred during the work with the SyRoTek was loosing localization of robots influenced by imperfect shielding of the arena from sunrays and other external influences.

Due to the debugging of the whole system, the programming was accompanied by a lot of system failures, but they were usually very quickly solved by the service engineer Mr. Chudoba, whom I would like to thank for the willingness.

[12] Application for autonomous planning of a robot to user defined position.

4.4.2 Brief summary of recommendations

More structured and brief summary of all recommendations can be found in the following list:

- Brief and structured documentation to control SyRoTek system from the Linux terminal.
- Possibility to download all camera records without asking for root access.
- More C++ source examples of how to program SyRoTek.
- Cleaning robot for removing dust from the surface of arena.
- Better shielding to deflect sunrays from the arena.

5 Conclusion

The main aim of this thesis was an integration, simulation and final verification of the extension of the model predictive control focused algorithms to avoid obstacles in dynamic environment. For purposes of this thesis, a working framework that is developed by the Department of Cybernetics CTU in Prague was provided.

All integrated algorithms were tested and simulated and results can be found in Appendix of this thesis. Summarized results were discussed in Section (3). All results showed that new extended rules of MPC work properly according to theory.

Verifying of new implemented rules of model predictive control theory on the real robotic system brought several interesting findings. One of them was the execution of two individual experiments on the SyRoTek system, where online and offline computation had to be separated and executed individually. The main reason of this arrangement was an insufficient computing power of the SyRoTek system. Comparison of these experiments demonstrates an advantage of online computation, where establishment of feedback controller had significant influence to final solution and the shape of formation stayed unchanged during executing the whole trajectory.

For me, the experience with the SyRoTek robotic system was inspirational and provides valuable experiences with the real robotic platform. For this occasion, a brief list of recommendations for the future development of SyRoTek was written and can be found in Section (4.4.1). I hope this thesis will bring benefits and help for members of the SyRoTek project in sense of future improvements and development of the whole platform.

5.1 Future of the project

As was mentioned in the motivation in the beginning of this thesis in Section (1.1.2) about possible applications, all simulations and experiments presented in this thesis are applicable to practical use in the real world. The demonstration using the SyRoTek system provided a significant proof that with more computing power, driving the nonholonomic robotic formations can be realized through the model-predictive control approach with implemented rules for avoiding dynamic obstacles.

6 Appendix A

6.1 Tables of results

Table 1: Comparison of final times of all implemented models

Nr.	**no model**	**linear**	**quadratic**
1	19,5	19	20,5
2	19	19,5	18,5
3	18,5	20	17,5
4	19	19	20
5	18,5	19	18
6	18,5	21	21
7	18,5	19	18,5
8	18,5	20	18
9	20	19	20
10	19	19	17,5
11	18,5	20	18,5
12	20	18,5	18
13	20	20	18
14	20	19,5	18
15	20	20	18
16	19	19,5	18
17	18,5	19	20
18	20	19	19
19	19	19	18,5
20	19	19,5	19
21	18,5	19	22,5
22	19	19,5	18,5
23	18,5	19	18,5
24	18,5	20	19
25	19,5	22	18,5
..	..	..	..
230	17,5	18,5	18
average	**18,69**	**18,97**	**18,35**

Table 2: Comparison of minimal distances to collision of all models

Nr.	**no model**	**linear**	**quadratic**
1	0,53	1,05	0,28
2	1,09	0,47	1,05
3	1,33	0,74	1,77
4	0,98	1,02	0,66
5	1,10	1,07	1,68
6	1,10	0,68	1,03
7	1,09	1,02	1,37
8	1,13	0,26	1,46
9	0,41	1,05	0,64
10	0,99	1,04	1,82
11	1,09	0,25	1,03
12	0,43	1,12	1,50
13	0,73	0,69	1,52
14	0,44	0,45	1,42
15	0,37	0,72	1,49
16	1,00	0,70	1,75
17	1,10	1,06	0,44
18	0,43	1,01	1,03
19	1,06	1,03	1,32
20	1,05	0,41	0,98
21	1,11	1,04	0,33
22	1,05	0,43	1,08
23	1,08	1,02	1,05
24	1,09	0,26	0,99
25	0,49	0,72	1,09
26	1,07	0,44	1,46
27	1,07	0,44	1,41
28	1,01	1,06	1,10
29	1,05	1,01	0,19
30	1,09	1,09	1,50
31	1,32	0,69	1,39
32	1,07	0,70	1,83
33	1,09	1,03	1,46
34	0,82	1,12	1,01
35	1,02	0,74	0,99
..	..	..	..
230	1,75	1,10	1,40
average	**1,16**	**0,97**	**1,32**

Table 3: Linear movement of dynamic obstacle, comparison of final time and minimal distance to collision

Nr.	**static**	**linear**	**static**	**linear**
1	20	19,5	0,53	0,57
2	19,5	19,5	0,53	0,52
3	19,5	19	0,56	0,55
4	19,5	19	0,52	0,52
5	19,5	19,5	0,53	0,45
6	19,5	19,5	0,55	0,55
7	19,5	18,5	0,54	0,55
8	18,5	19	0,59	0,56
9	18,5	19	0,55	0,58
10	19,5	19,5	0,40	0,56
11	19	19	0,55	0,58
12	19,5	19	0,53	0,56
13	19	19,5	0,53	0,57
14	19,5	19	0,53	0,54
15	18,5	19	0,52	0,54
16	18,5	19,5	0,21	0,55
17	18,5	19	0,55	0,60
18	19,5	19	0,41	0,54
19	18	19	0,49	0,61
20	19	19	0,54	0,57
21	20	21	0,73	0,59
22	18,5	18,5	0,55	0,52
23	18	19	0,55	0,61
24	18	19,5	0,55	0,53
25	19,5	19	0,61	0,64
26	19,5	19,5	0,54	0,57
27	18,5	20,5	0,39	0,52
28	18,5	19	0,55	0,60
29	19,5	19	0,50	0,56
30	19	19,5	0,54	0,49
31	19,5	19	0,39	0,60
32	18,5	19,5	0,47	0,58
33	19,5	19	0,49	0,56
34	18	19	0,55	0,57
35	19	19,5	0,49	0,55
..	..	..	..	..
230	18	19	0,56	0,62
average	**19,21**	**19,02**	**0,57**	**0,56**

7 Appendix B

CD Content

In table 4 are listed names of all root directories on CD

Directory name	**Description**
bp	bachelor thesis in pdf format.
sources	source codes for MPC
syrotek	source codes for the SyRoTek system
tables	tables of results
video	the SyRoTek experiment record

Table 4: CD Content

References

[1] Saska M. Trajectory planning and optimal control for formations of autonomous robots. In *Schriftenreihe Wurtzburger Forschungsberichte in Robotik und Telematik. Wurzburg: Universitat Wurzburg. ISSN 1868-7466*, volume Band 3, 2010.

[2] Saska M., Mejía S. J., Stipanović M. D., and Schilling K. Control and Navigation of Formations of Car-Like Robots on a Receding Horizon. In *IEEE International Conference on Control Applications*, 2009.

[3] Dunbar D. W. and Murray M. M. Distributed receding horizon control for multi-vehicle formation stabilization. In *Automatica*, pages 42(4):549–558, 2006.

[4] Dimarogonas D. V., Loizou S. G., Kyriakopoulos K. J., and Zavlanos M. M. A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents. In *Automatica*, pages 42(2):229–243, 2006.

[5] Fierro R., Das K. A., Kumar V. R., and Ostrowski P. J. Hybrid Control of Formations of Robots. In *Proc. of IEEE Conference on Robotics and Automation,*, 2001.

[6] Jaeger M. and Nebel B. Dynamic Decentralized Are Partitioning for Cooperating Cleaning Robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA).*

[7] Hardin T. C., Cui X., Ragade K. R., Graham H. J., and Elmaghraby S. A. A modified particle swarm algorithm for robotic mapping of hazardous environments. In *Proc. of World Automation Congress*, 2004.

[8] Yang S. X. and Luo C. A neural network approach to complete coverage path planning. In *IEEE Transactions on Systems, Man, and Cybernetics*, pages 34(1):718–724, 2004.

[9] Kulich M., Košnar K., Chudoba J., Faigl J., and Přeučil L. On a Mobile Robotics E-learning System. In *Proceedings of the Twentieth European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetics Studies. ISBN 978-3-85206-178-8*, pages 597–602, 2010.