Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 21, Nr 3/2017, 59–68, DOI: 10.14313/PAR_225/59

Mobile Robot Path Planning with Obstacle Avoidance using Particle Swarm Optimization

Ewelina Chołodowicz, Daniel Figurowski
Zachodniopomorski Uniwersytet Technologiczny w Szczecinie, Wydział Elektryczny, ul. Sikorskiego 37, 70-313 Szczecin

Abstract: This paper presents a constrained Particle Swarm Optimization (PSO) algorithm for mobile robot path planning with obstacle avoidance. The optimization problem is analyzed in static and dynamic environments. A smooth path based on cubic splines is generated by the interpolation of optimization solution; the fitness function takes into consideration the path length and obstacle-generated repulsive zones. World data transformation is introduced to reduce the optimization algorithm computational complexity. Different scenarios are used to test the algorithm in simulation and real-world experiments. In the latter case, a virtual robot following concept is exploited as part of the control strategy. The path generated by the algorithm is presented in results along with its execution by the mobile robot.

Keywords: mobile robot, path planning, obstacle avoidance, particle swarm optimization, dynamic environment

1. Introduction

Mobile robots have been successfully applied in many areas such as medical and military applications, space exploration, public and domestic duties. They can perform difficult and hazardous tasks with complex requirements and often have to do so autonomously, without the aid of a human operator. To function in that manner they must be able to navigate the environment they are placed in.

Collision-free path planning plays an important role in mobile robots navigation and is often a fundamental requirement for proper task execution. The main goal of such planning in an environment with static (stationary) and dynamic (moving) obstacles is to find a suitable movement path from a starting location to a destination, while avoiding the collision with any of these objects. This complex task poses many difficulties: computational complexity, adaptation to changing environment and determining a reasonable evaluation function for the generated path.

Path planning is an active research area and many methods have been developed to deal with this problem. They can be classified into classical and heuristic based search algorithms [1], mainly discerned by the type of optimization techniques utilized.

Recently some classical approaches, such as cell decomposition [2], potential field method [3–5], road map [6] and sub goal network have been presented in the field of mobile robotics. In a cell decomposition method a two-dimensional map is divided into several grids and the path is created in them. Another case of a classical approach is a potential field method in which the controlled robot is attracted by the destination while simultaneously being repelled by the obstacles.

These path planning algorithms suffer from some drawbacks [1], e.g., a solution may not be optimal because the algorithm gets stuck in local minima or a new solution has to be generated again when the environment changes and therefore the original path can become infeasible.

As a result, many heuristic based methods, such as fuzzy logic [7], artificial neural network [8], nature inspired algorithms [9–12] and hybrid algorithms were created. These methods can overcome drawbacks of the classical ones, but they do not guarantee to find the best solution. Still, the result can be sufficiently close to the optimal one. In this paper the authors used one of those methods – the Particle Swarm Optimization algorithm; it serves as a base solver for collision-free path planning problem.

Particle Swarm Optimization (PSO) is a metaheuristic algorithm which is inspired by the social foraging behavior of some animals such as bird flocking and fish schooling. It was developed by Kennedy and Eberhart in 1995 and its description is presented in [13]. Since then, many approaches have been suggested by the researches to solve the collision-free path planning problem using the PSO algorithm [11, 14–17].

Further sections of this paper are arranged as follows: Section 2 describes the goal of this article, overall concept of the designed algorithm, workspace definition and world data preprocessing. Section 3 presents simulation results. Section 4 contains hardware information, control system description and real-world experiment results. Conclusions are presented in the last section (Section 5).

Autor korespondujący:
Daniel Figurowski, Daniel.Figurowski@zut.edu.pl

Artykuł recenzowany
nadesłany 21.08.2017 r., przyjęty do druku 11.09.2017 r.

2. Problem Description

In this article the following problem is considered: in a two-dimensional workspace the shortest collision-free path is to be found. The path should display a smooth curvature to ensure its realizability. The controlled robot, circular in shape, is to traverse this workspace from its initial position to an arbitrary destination point (Fig. 1). The environment is populated by static and dynamic obstacles (also circular). While full information about the positions of all objects in the workspace is available, the kinodynamic properties of the dynamic obstacles are unknown and no attempt is made to identify them. The task of finding this collision-free path is the responsibility of a path-generating algorithm, development of which is the focal point of this article.

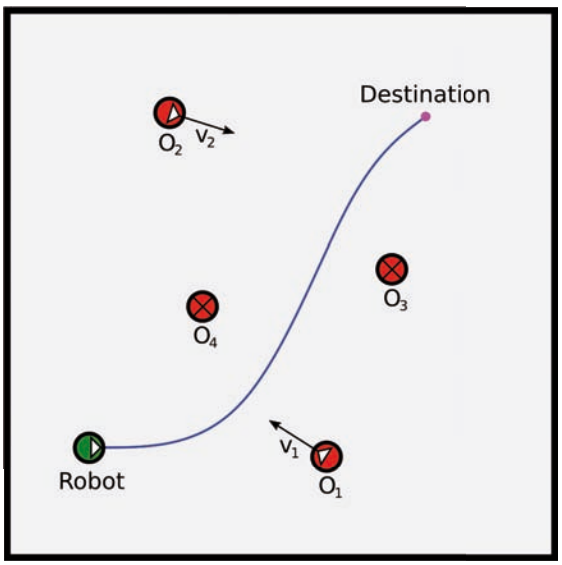


Fig. 1. Problem presentation
Rys. 1. Przedstawienie problemu

2.1. Solution Outline

The program workflow (Fig. 2) begins with the acquisition of the robot's current position, its destination and positions of obstacles. The evaluation of the world state is based on the collected data, which is earlier preprocessed and transformed (Fig. 3). A "state change" event in the environment occurs when a change in position of any object is greater than d_{SC}. If this condition is met the program checks whether the straight path is not obstructed by the obstacles. If it is not, it becomes the new path. In the opposite case, the transformed data is used as an optimization data for the Particle Swarm Optimization algorithm. The PSO routine searches for an optimal solution minimizing a fitness function and generates a reference path S_{ref}. Points from that path are used to calculate error signal for a PID controller. Finally, the control data is dispatched to the controlled robot. Program runs with parameters presented in Table 1.

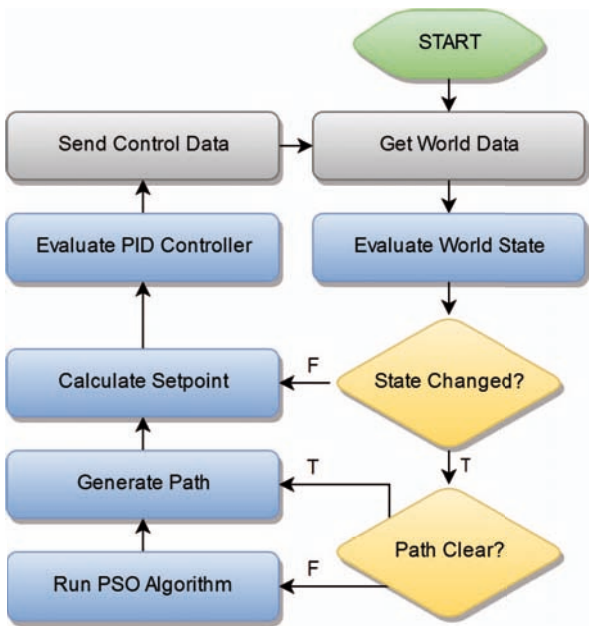


Fig. 2. Overall program flow chart
Rys. 2. Ogólny schemat blokowy programu

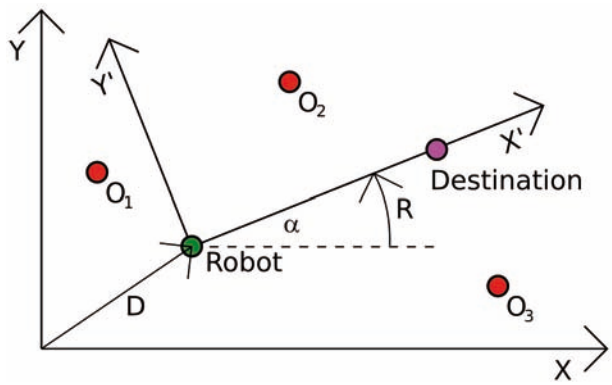


Fig. 3. World transformation
Rys. 3. Przekształcenie świata

Table 1. World parameters
Tabela 1. Parametry świata

Symbol	Description	Value
W_{size}	Workspace size	2 m × 2 m
d_r, d_o	Robot and obstacle diameter	0.1 m
t_s	Program cycle time	varying ≈ 0.1 s
n_{Pgen}	Number of cubic spline points	10
d_{SC}	Minimal obstacle distance triggering **State Changed** event	0.01 m

The obstacles have the same diameter as the controlled robot. In order to ensure that the path is not too close to the obstacles, the boundaries of obstacles have been extended by two robot radii. This expanded region is called a *critical area*. Program cycle time t_s is varying because a nondeterministic operating system and wireless communication utilizing TCP protocol were used.

Preprocessing of the world data is used to speed up the calculation of the Particle Swarm Optimization algorithm. It transforms the coordinate system from global (workspace) to local one (between robot and its destination point), with use of translation and rotation operations. That conversion allows to simplify the complexity of optimization process by reducing the dimensionality of search space. Similar approach can be found in [10, 17].

2.2. Optimization Algorithm

The problem of robot path planning is treated as a minimization problem and considered on the transformed search space limited by constraints. Each iteration of the PSO algorithm

generates a set of m points Pg_1, Pg_2, ..., Pg_m, where m is the number of obstacles present in the workspace. This set of points $P_{gen} = \{Pg_1, Pg_2, ..., Pg_m\}$ is then interpolated by a cubic spline function. The interpolated path consists of n_{Pgen} points forming S_{ref} set.

The fitness function to be minimized by the Particle Swarm Optimization algorithm consists of two parts. The first one evaluates the length of the path and the second part ensures the safety of the path, monitoring whether the obstacles are in an acceptable distance from the controlled robot.

The following set of equations describes the optimization algorithm. The fitness function f is given in the following form and minimized:

$$f(S_{ref},0) = s_{len}(S_{ref}) \cdot \left[1 + viol_{fa} \cdot viol(S_{ref},0)\right] \tag{1}$$

$$\min f(S_{ref},0) \tag{2}$$

where: S_{ref} – reference path points, O – obstacles positions, s_{len} – total path length function, $viol_{fa}$ – obstacle zone violation factor, $viol$ – violation function.

As mentioned before the fitness function f is split in two parts: path length and zone violation. The former is defined as following:

$$s_{len}(S_{ref}) = \sum_{i=1}^{n-1} dist\left[S_{ref}(i+1), S_{ref}(i)\right] \tag{3}$$

where: $n = n_{Pgen}$, $dist$ – Euclidean distance function.

While the latter one (zone violation):

$$viol(S_{ref},0) = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n}\max\left[1 - dist\left[O(i), S_{ref}(j)\right]\cdot\frac{3}{2}d_r\right] \tag{4}$$

where: m – number of obstacles, $n = n_{Pgen}$. It can be observed that the behavior of zone violation function is similar to a repulsive potential field.

The set of transformed points P_{gen} is given as following:

$$P_{gen} = \left\{P_i \in \Re : |P_i| < sp \wedge \forall Bnd : \left(P_i < Bnd_{low} \vee P_i > Bnd_{up}\right)\right\} \tag{5}$$

Each single value P_i of the set P_{gen} is subjected to constraints (Fig. 4). It must be placed in the search space sp and cannot lie within any of the obstacles' boundaries Bnd_{low}, Bnd_{up}. The search space sp equates to robot diameter plus half the distance between the robot and its destination.

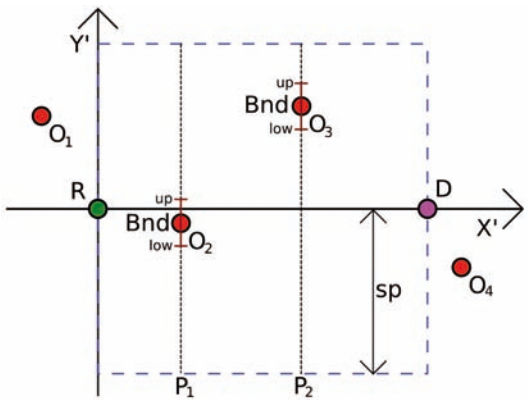


Fig. 4. Search space boundaries
Rys. 4. Ograniczenia przestrzeni poszukiwań rozwiązania

The following set of parameters in Table 2 was used during simulation and real-world experiments. The meaning of these parameters is discussed in [18].

Table 2. PSO algorithm parameters
Tabela 2. Parametry algorytmu PSO

Symbol	Description	Value
$n_{PSOiter}$	Number of iterations	50
sw_{size}	Swarm size	20
vel_{fa}	Search velocity factor	10%
vel_{damp}	Velocity damping factor	0.898
fs_{local}	Local fitness significance factor	1.5
fs_{global}	Global fitness significance factor	1.5
$viol_{fa}$	Obstacle zone violation factor	100

2.3. Comparison to Similar Solutions

As mentioned before, many researchers are concerned with the problem of path planning, and thus numerous solutions were developed. In this subsection a brief comparison with other approaches is provided in which the problem description is similar (albeit not identical) to the one presented in this paper. Two solutions based on PSO [15, 17] and one on Genetic Algorithm (GA) [10] are discussed in relation to the proposed method.

A brief summary of all the methods is given in Table 3; the discussion will be focused on differences between them and consequences thereof. First, however, it is useful to establish the common factors shared by all of the presented solutions:

— Workspace is two-dimensional,
— Both dynamic and static obstacles may be present,
— Underlying algorithms are heuristic in nature,
— Generated paths are evaluated by a fitness function.

The accuracy in the spatial description of the environment affects the quality of obtained solutions. Less precise methods rely on encompassing workspace objects in primitive shapes, thus the area occupied by such element is artificially extended. Better, more aggressive paths (i.e. nearer the obstacle's real boundary) may not be discovered with this approach. However, the more accurate environment description comes at the expense of computational complexity.

When the environment is populated by well-defined classes of objects (i.e. their shape is known) it is unnecessary to strive for a more complex description. This approach, adopted in the proposed method, is also present in [10] and [17]. Several shape classes are recognized in the latter solution. While polygons are considered in [15], they are ultimately encompassed in rectangular areas.

Additionally, an algorithm may also be supplied with dynamic obstacles' kinematics. While the spatial description describes only the current configuration space, kinematics offers a mean to predict the outcome of future time frames. This prediction allows the path generator to plan accordingly and thus results in safer paths overall. Some rudimentary kinematics information may also be inferred by comparing successive time frames. This would require the algorithm to memorize previous environment states and track the behavior of all objects. Additional information about the dynamic obstacles' kinematics is exploited in [15, 17]. Intersections of the controlled robot's and obstacles' paths

Table 3. Solutions comparison
Tabela 3. Porównanie rozwiązań

Solution	Algorithm	Environment	Workspace Transformation	Fitness Function Objectives	Path Smoothness
Proposed	PSO	2D workspace with circular static and dynamic obstacles, no information about kinematics of dynamic obstacles	yes (1D) varied spacing	path length distance to obstacles	cubic splines
[10]	GA	2D workspace with circular static and dynamic obstacles, no information about kinematics of dynamic obstacles	yes (1D) equal spacing	path length distance to obstacles smoothing	angle deviation minimization, smoothing operator
[15]	PSO	2D workspace with rectangle-enclosed static and dynamic obstacles, kinematics of dynamic obstacles is known	no	path length travel time	no
[17]	PSO	2D workspace with varied static and dynamic obstacles, kinematics of dynamic obstacles is known	yes (1D) equal spacing	path length distance to obstacles	parabolic function

are considered as points of potential collision and the algorithm checks whether one occurs in the future, if so, an alternative path is sought.

If the algorithm does not employ any mechanism to utilize the objects' kinematics, then the dynamic obstacles are treated as if they were static. Each time frame is treated independently from the other and path must be recalculated. This is the approach presented both in [10] and the proposed method.

In several of the solutions a workspace transformation is applied in order to further reduce the computational complexity; two-dimensional search space is converted to a single dimension. One of the algorithms' discerning features is the method for choosing points where the transformation is conducted. In the presented method, a nonuniform distribution is used, the selected points are tied to the obstacles' positions and number. This is contrary to other solutions in which a uniform distribution is utilized and the number of points is one of the algorithm's parameters.

Another important distinction is the definition of the path evaluation function. While all the solutions agree upon the minimization of the path length as one of the criteria, there are notable differences how other factors are evaluated, if at all. Method [15] is the only one that does not take into account the distance to the obstacles (path safety criterion); instead, any solution with a collision is simply discarded. Also, information about the obstacles' kinematics may be used to optimize the path in terms of time, and not only length. In [17] the path safety criterion is included in the evaluation function; however, its binary nature leads to the same behavior as in [15].

Beside the length, the Genetic Algorithm [10] optimization procedure takes into account the path's linearity – with the goal of minimizing the curvature. This further reinforces the smoothing effect, potentially elongating the path itself. The costs of violating the safe distance from each obstacle are aggregated and constitute the path safety criterion. Complete evaluation function is a sum of these three components.

In the proposed solution, a similar approach to the GA algorithm was taken in terms of path safety criterion. One major difference being that the obstacle distance penalization function is normalized and its value is scaled with the path's length. This ensures the correlation between the optimization parameters and results in only one user-defined parameter – the safety zone violation coefficient.

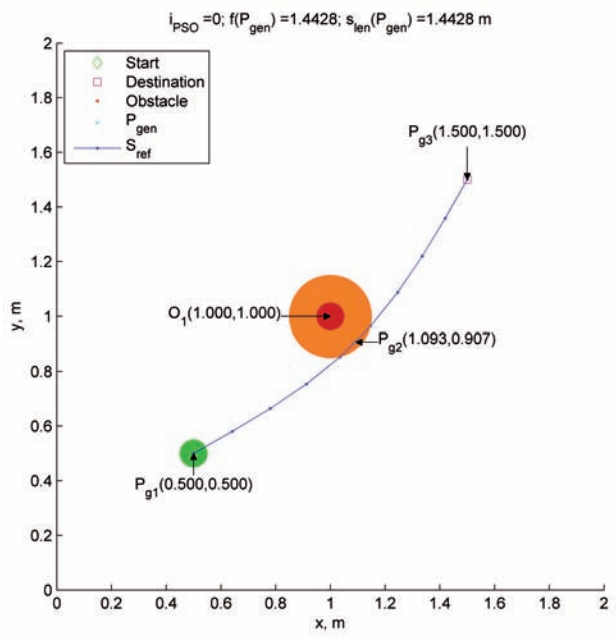


Fig. 5. Optimization result for scenario a)
Rys. 5. Wynik optymalizacji dla scenariusza a)

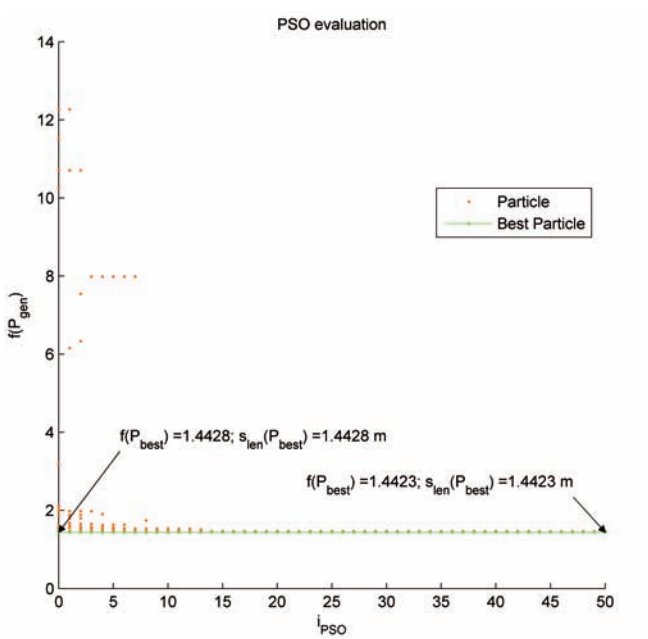
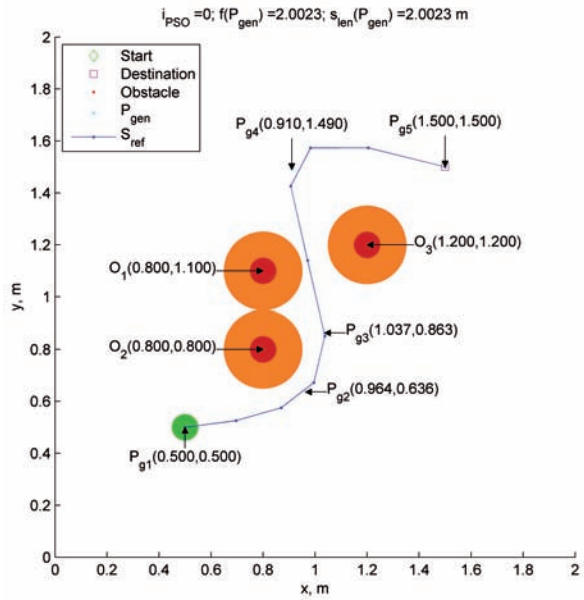
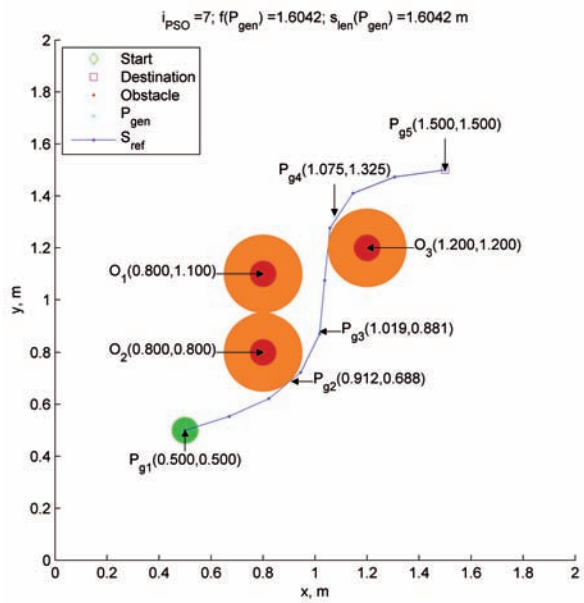


Fig. 6. Optimization process for scenario a)
Rys. 6. Proces optymalizacji dla scenariusza a)





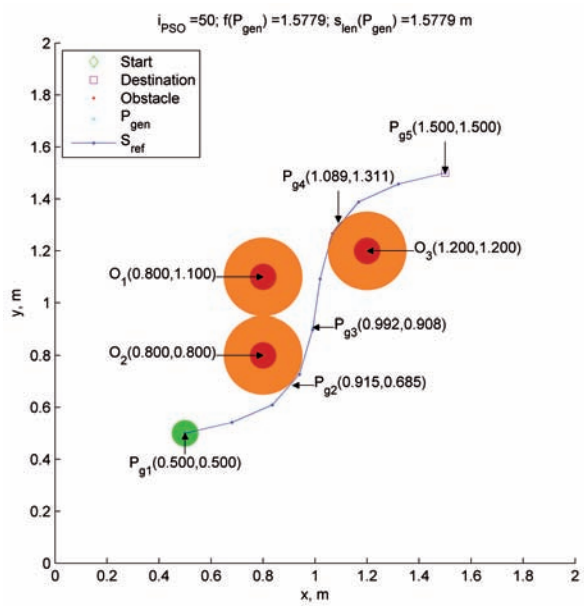


Fig. 7. Optimization result for scenario b)
Rys. 7. Wynik optymalizacji dla scenariusza b)

In most of the solutions, methods for smoothing the generated paths were applied in order to improve the paths' traversability. Authors of [17] employed a parabolic function calculated with the use of optimized path's points as well as obstacles' positions and velocities. This approach assumes that the velocity of dynamic obstacles is constant.

The algorithm presented in [10] utilizes two methods of path smoothing: one as an optimization criterion and the other as a genetic operator. The latter operates on the principle of replacing a curve node by additional three points. In the proposed solution, cubic spline functions were used which, in contrast to ordinary polynomials, avoid the problem of oscillations.

3. Simulation

This section presents results of simulation experiments conducted to validate the feasibility of the proposed method. The controlled robot is stationary and the path is only generated for its initial points. The following scenarios were tested:
a) one static obstacle,
b) three static obstacles,
c) one static and two dynamic obstacles.

The optimization result figures in this section depict the generated reference path between the starting and destination points. In addition a set of P_{gen} points used in interpolation method is shown. The areas occupied physically by the robot and obstacles are marked as well as the obstacles' critical areas. X and Y axes represent the 2D workspace. Current optimization iteration, fitness function value and reference path length are all presented in a figure's label.

An optimization process figure presents the fitness function values' evolution in subsequent iterations of the PSO algorithm. For every iteration, fitness of each particle is shown and the best solution is highlighted. In addition, fitness value and reference path length of first and last iteration are annotated.

It can be seen that with the rising number of obstacles (Fig. 7) optimization problem becomes more complex and not all particles converge to the best solution (Fig. 8). Still, the generated path is smooth and realizable. The problem is solved around 10th iteration of the PSO algorithm with an acceptable fitness value.

Last set of optimization results (Fig. 9) depicts a scenario with two moving obstacles. The velocity value for each one is anno-

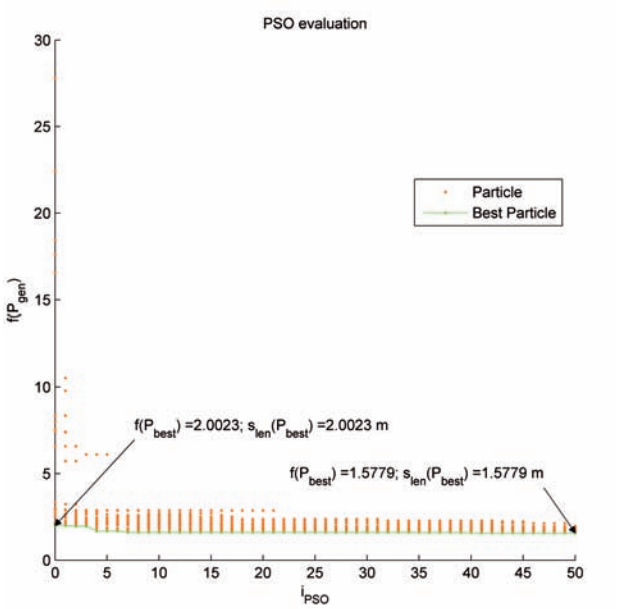


Fig. 8. Optimization process for scenario b)
Rys. 8. Proces optymalizacji dla scenariusza b)

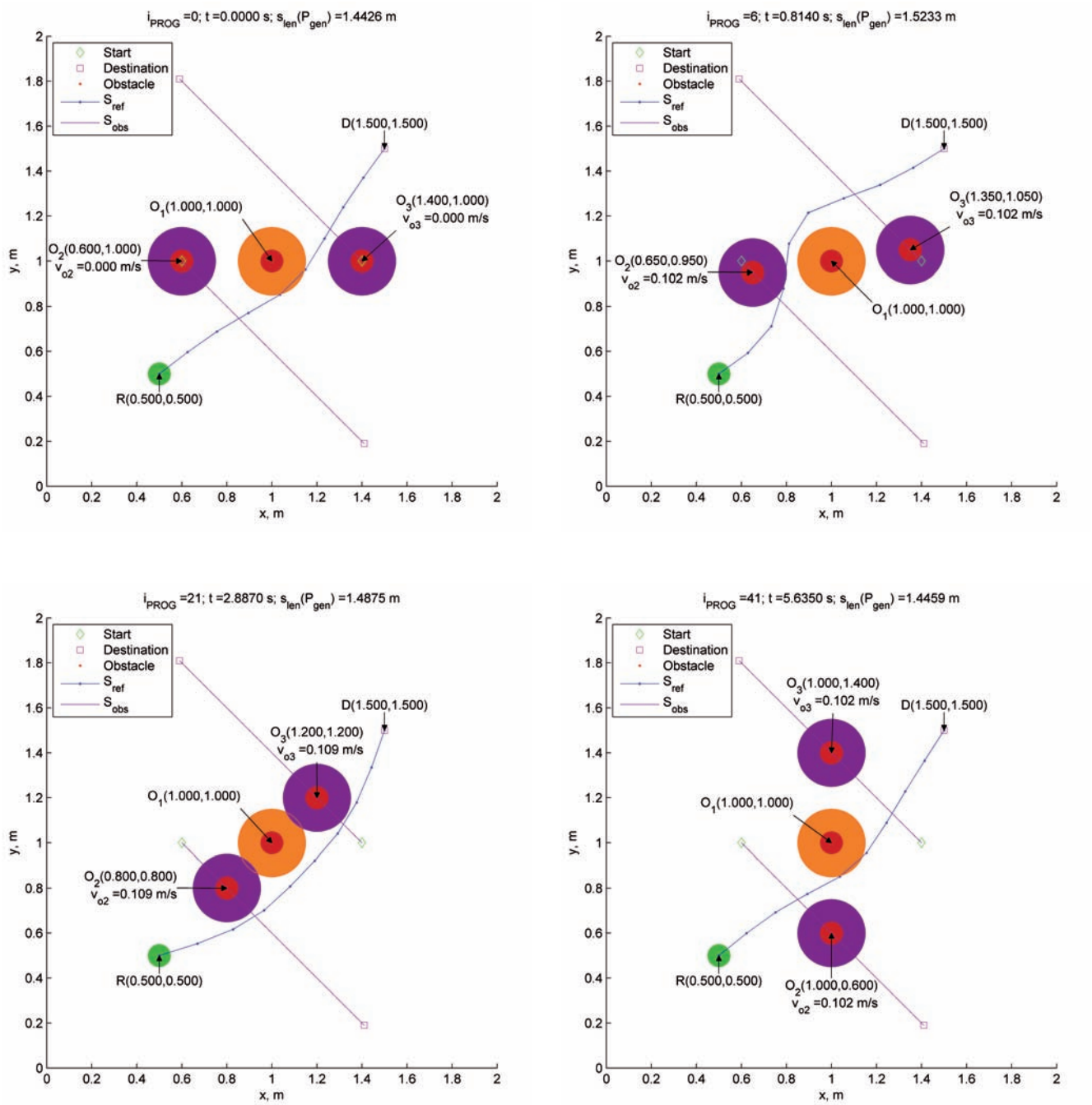


Fig. 9. Optimization result for scenario c)
Rys. 9. Wynik optymalizacji dla scenariusza c)

tated. Also paths of these obstacles with their corresponding starting points and destinations are presented. Current program iteration, time and reference path length are shown in the label.

While the stationary nature of the controlled robot remains the same, the algorithm must deal with a dynamic workspace. The presence of the two moving obstacles enforces the algorithm to reevaluate the planned path as changes in the environment state are detected. The PSO itself is not concerned with the kinematics of the obstacles, it treats each environmental setup as one with all obstacles static.

With the evolution of the environment (i.e. the obstacles are moving) the generated path also changes (Fig. 9). It can be seen that the more straightforward solution achieved at the end of the simulation is earlier denied by the dynamic obstacles' critical areas. While this leads to different path lengths, the curves of proposed solutions are relatively smooth and should be achievable by a real robot.

4. Real-World Experiment

In a real-world experiment, a mobile robot navigates in a dynamic environment. The information about the mobile robot and obstacles in the workspace is provided by the visual feedback. A detailed description of the laboratory stand used to test the presented algorithm can be found in [19].

4.1. Control Strategy

Calculation of setpoint values for control algorithm is based on the idea of following a virtual robot (Fig. 10) [20]. The virtual robot's center point is responsible for obtaining the velocity and orientation of the controlled robot; it is determined by the position of the controlled robot in relation to the reference path.

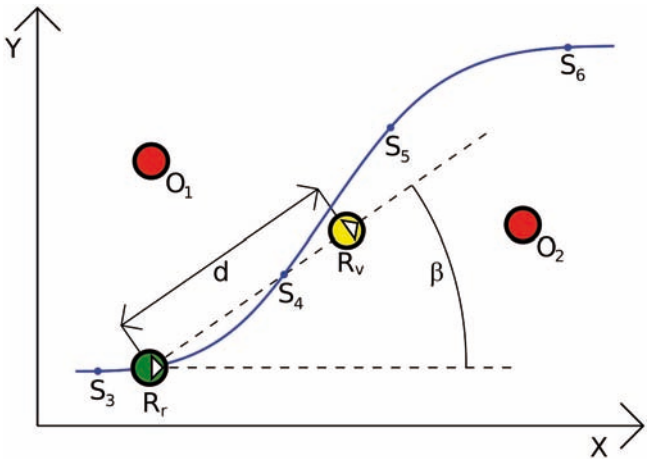


Fig. 10. Concept of following the virtual robot
Rys. 10. Idea podążania za robotem wirtualnym

The algorithm searches for the nearest point on the reference path which has not yet been reached by the controlled robot and calculates the angle between the controlled robot and that point. Second part of the algorithm determines the velocity which is in turn converted to the distance between the controlled robot and the virtual one. This velocity depends on the algorithm settings, remaining path distance and takes into account braking distance.

In order to perform the path following, a proportional-integral-derivative (PID) controller is implemented in a broader control structure with parameters given in Table 4. The controller equation is defined as follows:

$$y(k) = P \cdot e(k) + I \cdot \sum_{i=0}^{k} \left[e(i) \cdot t_s(i-1) \right] + D \frac{\left[e(k) - e(k-1) \right]}{t_s(k-1)} \quad (6)$$

where: t_s – varying program cycle time. In fact, after experimenting with the parameters, a simpler P controller was found to be sufficient.

Table 4. Control system parameters
Tabela 4. Parametry układu sterowania

Symbol	Description	Value
v_{set}	Robot nominal velocity	0.3 m/s
a_{acc} a_{dec}	Robot acceleration and deceleration	0.15 m/s^2
d_{ppoint}	Minimal distance to the current path point to trigger next point selection	0.05 m
P_{dist}	Proportional factor of distance controller	2
I_{dist} D_{dist}	Integral and derivative factors of distance controller	0
SO_{dist}	Switch on level of distance controller	0.01 m
P_{ang}	Proportional factor of angle controller	0.75
I_{ang} D_{ang}	Integral and derivative factors of angle controller	0
SO_{ang}	Switch on level of angle controller	10°

4.2. Results

A set of real-world experiments with different environmental setups was performed to verify the effectiveness of the proposed algorithm. In contrast to the simulation section, the robot actually traversed the generated path. Newly generated paths in next iterations of the algorithm use robot's observed position as their starting point. The following scenarios were tested:

a) three static obstacles,
b) two static and one dynamic obstacles.

In the real-world experiment optimization figures the controlled robot's actual movement path is shown to visualize the following of the reference path. Maximum and mean error values are presented in a figure's label.

It can be seen that the most problematic task for the controlled robot is driving over curves. The large overshoot in actual path traversed by the robot (Fig. 11) can be attributed to manually selected PID controller parameters. Despite this, the error is within an acceptable range.

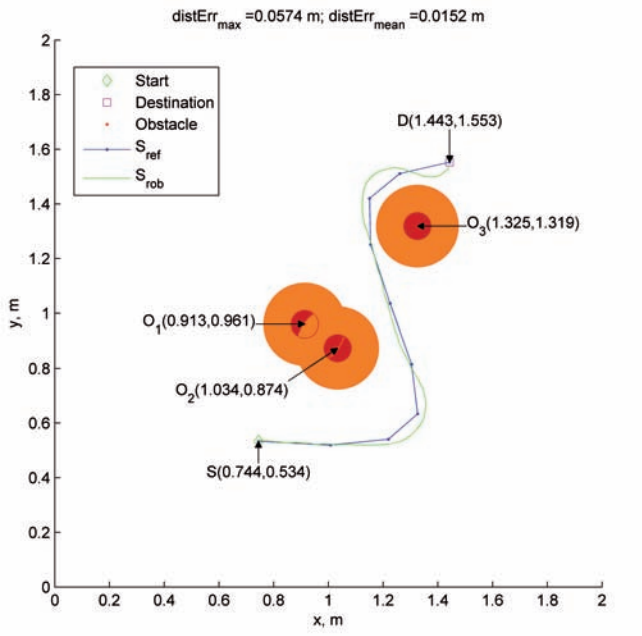


Fig. 11. Optimization result for scenario a)
Rys. 11. Wynik optymalizacji dla scenariusza a)

In second scenario (Fig. 12) the dynamic case is presented. Controlled robot velocity is annotated and path driven from starting position to destination is shown.

This result illustrates a real-world path adaptation to a moving obstacle. Similar to the simulation results, the generated reference paths are smooth and collision-free. The controlled robot was able to follow them easily while avoiding collision with any of the obstacles.

An interesting situation arose between the first two snapshots of the robot movement ($t = 0$ s and $t = 1.975$ s). It is obvious that the initial path was ultimately ignored in later iterations ($t = 1.975$ s), even though the dynamic obstacle was not in a position to distort it. This can be attributed to errors caused by the practical nature of the experiment, e.g., robot dynamics, vision system inaccuracies. However the fitness values of these two paths must be very similar as they are mostly symmetrical.

Another interesting feature of the presented approach is visible in the last snapshot. A straight line unobstructed by any obstacle is found between the robot's actual position and the destination point. In this case the presented algorithm chooses it immediately ignoring the cubic spline interpolation.

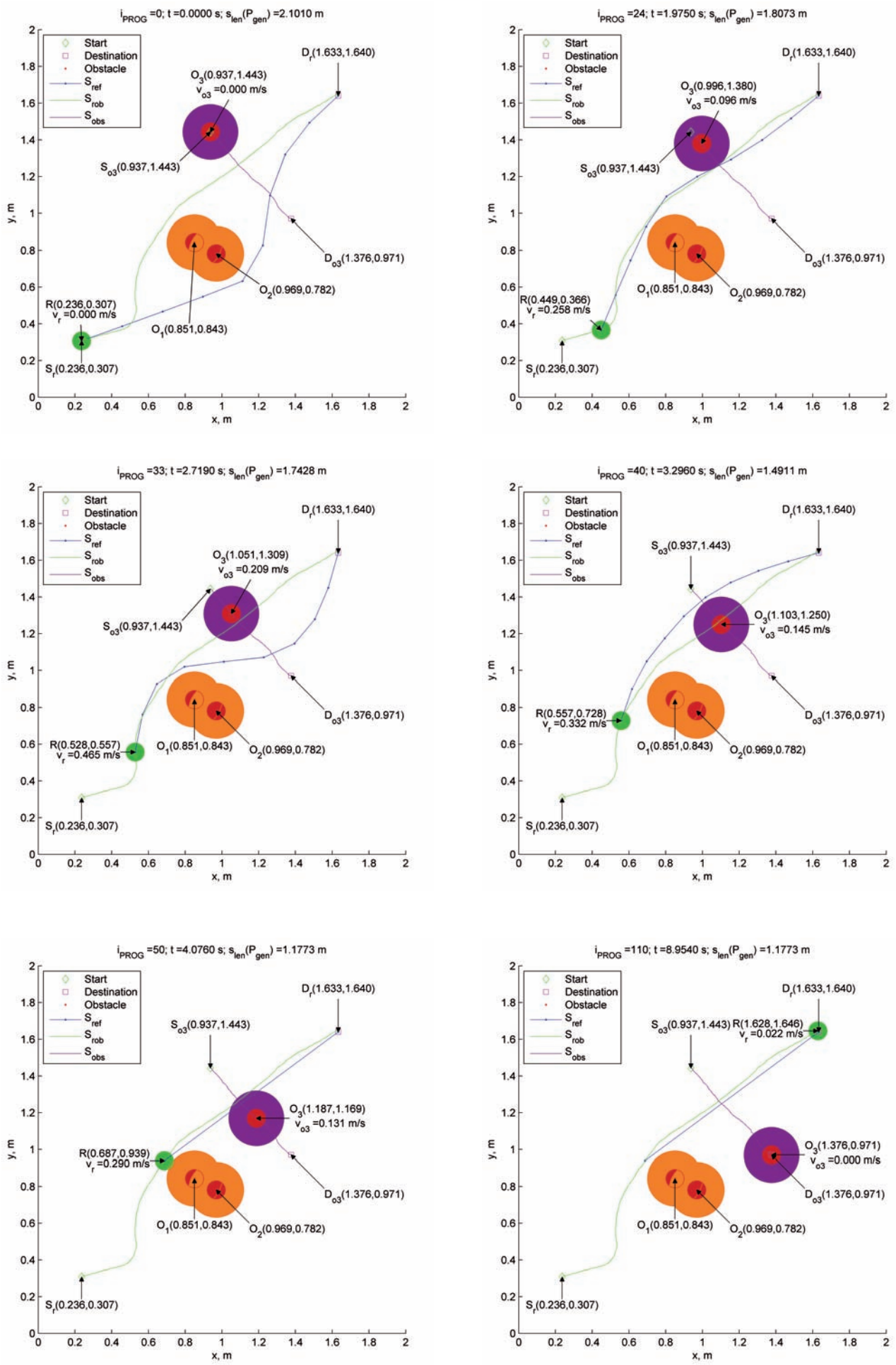


Fig. 12. Optimization result for scenario b)
Rys. 12. Wynik optymalizacji dla scenariusza b)

5. Conclusions

Overall, the implemented Particle Swam Optimization algorithm was successful at generating paths in an environment with static and dynamic obstacles. Simulation results show that the proposed method is feasible and real-world experiment confirms its efficiency; the generated paths are smooth and achievable by the robot. The results prove that the presented PSO-based algorithm is applicable to the field of mobile robotics for obtaining reasonable collision-free paths in a two--dimensional environment.

Bibliography

1. Mac T.T., Copot C., Tran D.T., Keyser R., *Heuristic approaches in robot path planning: A survey*, Robotics and Autonomous Systems, Vol. 86, 2016, 13–28, DOI: 10.1016/j.robot.2016.08.001.
2. Kloetzer M., Mahulea C., Gonzalez R., *Optimizing cell decomposition path planning for mobile robots using different metrics*, Proceedings of 19th International Conference on System Theory, Control and Computing, Cheile Gradistei, 2015, 565–570, DOI: 10.1109/ICSTCC.2015.7321353.
3. Zhang Y., Liu Z., Chang L., *A new adaptive artificial potential field and rolling window method for mobile robot path planning*, Proceedings of 29th Chinese Control And Decision Conference, Chongqing, 2017, 7144–7148, DOI: 10.1109/CCDC.2017.7978472.
4. Oborski P., Fedorczyk T., *Zmodyfikowana metoda pól potencjałowych do wyznaczania drogi robota mobilnego*, „Pomiary Automatyka Robotyka", R. 19, Nr 2, 2015, 57–64, DOI: 10.14313/PAR_216/57.
5. Szulczyński P., Pazderski D., Kozłowski K., *Real-time obstacle avoidance using harmonic potential functions*, "Journal of Automation Mobile Robotics and Intelligent Systems", Vol. 5, No. 3, 2011, 59–66.
6. Velagic J., Delimustafic D., Osmankovic D., *Mobile robot navigation system based on Probabilistic Road Map* (*PRM*) *with Halton sampling of configuration space*, Proceedings of IEEE 23rd International Symposium on Industrial Electronics, Istanbul, 2014, 1227–1232, DOI: 10.1109/ISIE.2014.6864789.
7. Hong C., Park C.W., Kim J.-H., *Evolutionary dual rule--based fuzzy path planner for omnidirectional mobile robot*, IEEE International Conference on Fuzzy Systems, Vancouver, 2016, 767–774, DOI: 10.1109/FUZZ--IEEE.2016.7737765.
8. Hendzel Z., Szuster M., *Neural sensor-based navigation of wheeled mobile robot in unknown environment*, "Pomiary Automatyka Robotyka", R. 17, Nr 1, 2013, 114–120.
9. Palmieri L., Arras K., *A novel RRT extend function for gefficient and smooth mobile robot motion planning* , IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, 2014, 205–211, DOI: 10.1109/IROS.2014.6942562.
10. Shi P., Cui Y., *Dynamic path planning for mobile robot based on genetic algorithm in unknown environment*, Chinese Control and Decision Conference, Xuzhou, 2010, 4325–4329, DOI: 10.1109/CCDC.2010.5498349.
11. Mac T., Copot C., Tran D., De Keyser R., *A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization*, "Applied Soft Computing", Vol. 59, 2017, 68–76, DOI: 10.1016/j.asoc.2017.05.012.
12. Contreras-Cruz M., Ayala-Ramirez V., Hernandez-Belmonte U., *Mobile robot path planning using artificial bee colony and evolutionary programming*, "Applied Soft Computing", Vol. 30, 2015, 319–328, DOI: 10.1016/j.asoc.2015.01.067.
13. Kennedy J., Eberhart R., *Particle swarm optimization*, IEEE International Conference on Neural Networks, Perth, Vol. 4, 1995, 1942–1948.
14. Arana-Daniel N., Gallegos A., López-Franco C., Alanis A., *Smooth global and local path planning for mobile robot using particle swarm optimization, radial basis functions, splines and Bézier curves*, IEEE Congress on Evolutionary Computation, Beijing, 2014, 175–182, DOI: 10.1109/CEC.2014.6900244.
15. Raja P., Pugazhenthi S., *Path Planning for Mobile Robots in Dynamic Environments Using Particle Swarm Optimization*, International Conference on Advances in Recent Technologies in Communication and Computing, Kottayam, 2009, 401–405, DOI: 10.1109/ARTCom.2009.24.
16. Saska M., Macas M., Preucil L., Lhotska L., *Robot Path Planning using Particle Swarm Optimization of Ferguson Splines*, IEEE Conference on Emerging Technologies and Factory Automation, Prague, 2006, 833–839.
17. Wang L., Liu Y., Deng H., Xu Y., *Obstacle-avoidance Path Planning for Soccer Robots Using Particle Swarm Optimization*, IEEE International Conference on Robotics and Biomimetics, Kunming, 2006, 1233–1238.
18. Shi Y., Eberhart R., *Parameter selection in particle swarm optimization*, Evolutionary Programming VII, 1998, 591–600.
19. Figurowski D., Brasel M., Kubicki M., *Stanowisko laboratoryjne do badań algorytmów sterowania robotami mobilnymi z wizyjnym sprzężeniem zwrotnym*, „Pomiary Automatyka Robotyka", R. 20, Nr 3, 2016, 71–76, DOI: 10.14313/PAR_221/71.
20. Bak M., Poulsen N., Ravn O., *Receding horizon approach to path following mobile robot in the presence of velocity constraints*, European Control Conference, Porto, 2001, 1151–1156.

Planowanie bezkolizyjnej ścieżki ruchu robota mobilnego przy użyciu algorytmu rojowego

Streszczenie: W artykule przedstawiono algorytm rojowy z ograniczeniami realizujący planowanie bezkolizyjnej ścieżki ruchu robota mobilnego. Problem optymalizacyjny został przeanalizowany dla środowiska statycznego i dynamicznego. Do stworzenia gładkiej ścieżki ruchu wykorzystano interpolację rozwiązania optymalizacji przy użyciu sześciennych funkcji sklejanych. Funkcja kosztu uwzględnia długość ścieżki ruchu oraz penalizację za naruszenie przestrzeni przeszkód. Wprowadzono transformację świata w celu redukcji złożoności obliczeniowej algorytmu optymalizacji. Przeprowadzono zróżnicowane scenariusze badawcze testujące algorytm w eksperymentach symulacyjnych i rzeczywistych. W przypadku tych ostatnich wykorzystano ideę podążania za wirtualnym robotem. Zaprezentowano wyniki obrazujące wygenerowaną ścieżkę ruchu oraz ocenę jej realizacji przez robota mobilnego.

Słowa kluczowe: robot mobilny, planowanie ścieżki ruchu, unikanie przeszkód, algorytm rojowy, dynamiczne środowisko

Ewelina Chołodowicz
cholodowicz.ewelina@gmail.com

Student at the Faculty of Electrical Engineering at West Pomeranian University of Technology Szczecin. Winner of the Mayor of Szczecin scientific scholarships for students – XI and XIV edition. Winner of the Ministry of Science and Higher Education scholarships in 2015/2016 and 2016/2017. Recent research topics are modeling, simulation, control of dynamic systems, optimization, logistics and mobile robotics.

Daniel Figurowski, Msc, Eng.
Daniel.Figurowski@zut.edu.pl

PhD student in Department of Industrial Automation and Robotics at West Pomeranian University of Technology Szczecin. His research interest revolves around the field of mobile robotics and focuses on the following problems: sensors data fusion, cooperative task execution, control algorithms, embedded controllers and image processing.