Proceedings of the VI Brazilian Conference on Neural Networks - VI Congresso Brasileiro de Redes Neurais
pp. 289–294, June 2–5, 2003 – Centro Universitário da FEI, São Paulo - SP - Brazil

Obstacle Avoidance in Dynamic Environment: a Hierarchical Solution

Areolino de Almeida Neto[1], Bodo Heimann[2], Luiz Carlos S. Góes[3], Cairo L. Nascimento Jr.[3]
[1]Universidade Federal do Maranhão, Av. dos Portugueses, Campus do Bacanga, S. Luís-MA-Brazil
[2]Mechatronik-Zentrum Hannover, Appelstrasse 11, Hannover-NI-Germany
[3]Instituto Tecnológico de Aeronáutica, Pça Eduardo Gomes 50, S. J. dos Campos-SP-Brazil
E-mails: areolinoneto@yahoo.com.br, heimann@mzh.uni-hannover.de, goes@mec.ita.br, cairo@ele.ita.br

Abstract

This article presents a concept for obstacle avoidance in dynamic environment suitable for mobile robot. The task of obstacle avoidance is divided in three principal groups: local, global and for emergencies. The local avoidance is here approached, in which the concept used is based on reinforcement learning, in such a way that the situations are divided into four states and two kinds of actions are possible. The states define in what situation the movement relationship between the robot and the dynamic obstacles is present, and the actions decide in which direction the robot must follow, in order to avoid a possible collision. And besides, here it is showed also how the state-action matrix was filled and its representation using neural network.

1. Introdution

In recent years many researches have their attention pointed to the solution of obstacle avoidance, principally obstacles which change their position. For static obstacles, many strategies were presented with excellent results. But for dynamic obstacle, there is too much to do, although many proposals have interesting results.

Different strategies were and are presented in order to solve this problem. It is possible to divide these strategies in two classes: model-based and learning-based. The model-based concept utilizes mathematical models to describe the robot and obstacles movement and to describe the possibility of an collision, and therefore how to avoid them. On the other hand, learning-based methods use the knowledge obtained in/from real situations and "learn" the way to avoid obstacles.

Regarding the model-based methods, interesting work is presented by Freund et al. [2], which deals with model and conditions for safety path planning. Shiller et al. present another excellent work [8], in which mathematical model of free places is developed.

An interesting learning-based method in obstacle avoidance is presented by Tang et al. [10]. In this work rules for the possibility of collision are detailed and implemented in a fuzzy system. Another interesting work is presented by Tsourveloudis et al. [11]. In this work a fuzzy system is combined with the electrostatic potential field method for path planning. In the work of Kawano and Ura [5], it is presented a path planning done by multi reinforcement learning (RL) modules, which can also accept external command. Yen and Hickey ([12]) propose improvements in the performance of traditional RL methods used for robotic navigation. Gachet et al. ([3]) propose a special Neural Network representation of a RL system.

One question is always present, when we see around this kind of problem, which can be solved by animals and is difficult to construct a machine with the same characteristic: do animals do hard calculus in order to find a solution for these "trivial" problems? Due to this question, this article presents an imitation of how human beings can solve these problems. Here, the most important tool is memory. With memory, a past experience can be used in a similar future situation, and the most successful solution applied in that case is reused in the present situation. It seems that our mind attempts to "remember the future" based on what has occurred in the past [4]. For such, it is necessary to have the following conditions:

- Plausible encoding of the information from the environment;
- Good information storage;
- Easy retrieval of the information.

And in order to make decisions, it is necessary too a proper evaluation of the decision taken in the past, such that it is possible to "learn" which decision is better than other, and then choose it in the future.

Considering the conditions above, specially the one about decisions, this work uses the learning-based method, more precisely RL method, for local obstacle avoidance. These methods try to acquire the knowledge necessary for the solution by trials and evaluation of each solution. In this work, the way to represent the movements, possibilities and situations (encoding) are presented, as well as the procedure of storage of evaluated decisions.

This article has the following organization: in section 2 the desired characteristic of learning methods are presented, followed by a discussion about the RL technique, as well as the states and actions used here are detailed. Then, the architecture of the obstacle avoidance system is shown. The results obtained in simulations are presented and the conclusions and future works are depicted.

2. Desired Characteristic of Learning Methods

Diverse techniques can be used in order to solve this problem. Neural Network, Fuzzy Logic and Reinforcement Learning are some of artificial intelligence techniques, which can be used. Each of these has particular characteristic, such that one can be more suitable for a type of problem. For obstacle avoidance, it is difficult to say which strategy is the best. However, it is possible to say the characteristics, which are desirable in a technique, such that it is more eligible for the obstacle avoidance problem:

- Intuitive data;
- Cumulative learning;
- Constructive solution;
- Direct knowledge acquisition.

Intuitive data means that the data used by a strategy should be “visible”, like distance between the robot and one obstacle, time for occurrence of impact and turns in robot movement. Quantities strongly modified by complex mathematical equations are good for precise calculations, but they become a value more artificial, less intuitive and, therefore, tend the interpretation of them to be difficult.

The problem of dynamic obstacle avoidance consists of several situations, which are different among them, but are interconnected too. Then the learning acquired for a given situation must be kept, while another knowledge is learned for another situation. That means cumulative learning. More, if a learning already acquired should be modified, the modification must not interfere in other knowledge, which doesn’t have relation with it.

Another characteristic of dynamic obstacle avoidance is that a unique action normally is not sufficient for a collision free movement. Many times it is necessary to have a serie of actions. This complex task can be performed by a set of primitive actions [3]. Therefore a solution can be well founded when hierarchical decisions are taken in a well-defined sequence. For example, for a certain obstacle, turn right and after go ahead makes a robot to avoid it, but the same cannot be true if it first goes ahead and then turns right.

An important characteristic of a learning strategy is the possibility of acquiring a knowledge at the first time it is presented to this strategy, or in some cases after few times. It is not good, when knowledge must be presented many times repeatedly, until the learning strategy can acquire this knowledge. If for an obstacle is coming in front of the robot, turn right or left should be learned at the first time, or in the worst case after three or four times.

Based on these four characteristics, it seems to be a good option, the reinforcement learning (RL) technique. Moreover, there are other conditions for this choice: no previous knowledge of the correct solution must be known by the designer, as Fuzzy Logic needs; and the environment also must not present the correct solution to the learner [12]. RL techniques can learn a solution without this information. They can learn by trial-and-error method, but with directions to the correct way of the solution.

3. The RL techniques, States and Actions

Diverse RL techniques exist nowadays. They can be classified as Direct Reinforcement (DR), value function methods and Actor-Critic [6]. Each one has advantages and characteristic suitable for a particular class of problem. An excellent reference on this topic is presented by Sutton and Barto [9].

Considering the characteristics presented in section 2, principally the last three characteristics, the value function methods seem to be a good option for obstacle avoidance and more precisely the Monte Carlo technique offers good reasons to be selected: it has cumulative learning, whereas it is possible, a solution for a given situation to be learnt today, and tomorrow another knowledge can be learnt for another situation, with no interference from previous knowledge acquired, except if an interference with relationship aspect is desired. The constructive solution can be performed defining primitive actions on robot’s movement, such that a set of these basic actions can perform all complex movements. And the most important characteristic is achieved because Monte Carlo updates the Q-value direct into Q-value matrix. But for excellent results, a proper function of the performance in obstacle avoidance must be defined.

The evaluation function utilized in this work during training, in order to classify the decisions taken by the robot in an obstacle avoidance situation, is the time spent by the robot from the beginning of the robot-obstacle transaction until the end of this transaction. Here, the most important thing, when an obstacle avoidance movement is taken, is not the absolute time value, but the order of these values, that means, the qualitative evaluation is more important than the quantitative value. And this classification will be used wherever another similar situation is presented to the robot and, therefore, is necessary to decide which action to take. In fact, the evaluation function f used is shown in equation 1 below.

$$f = \frac{-100}{t + a/10 + n/600} \qquad (1)$$

where t is the time spent during the avoidance, a is the number of actions taken and n is the number of trials performed. The terms a and n are used, because it is possible, two actions for the same robot-obstacle situation spent the same time, thus, in order to have different values for all actions ever, these components are added to the time spent. The reason of the inversion and the constant –100 is due to the initialization of the matrix used for saving the states-actions values. This initialization was made with zero for all elements of this matrix, except to those, which correspond to the action "go ahead". This way permits the decision with the best evaluation to have a minimal value and to be inferior to zero.

Considering the constructive solution of complex tasks based on primitive behaviors, an obstacle avoidance solution is obtained with a series of actions in directions and velocities. Thus, it is necessary first to know if a sequence of actions was able to avoid a collision. If so, then the function *f* above is calculated; if not, then it is not possible to say about the actions taken and no action is updated. And more, for consistency reasons, the evaluation values are saved into matrix only for the last action taken in one state. That means, in a sequence of decisions, it is possible for a given state S1, the action A1 to be taken and after, but yet in the same problem, the robot can be in the same state S1 and another action A2 is taken, then only this action is evaluated, because it conducted the robot to safety place. This procedure is made for all states visited by the robot for specific obstacle avoidance.

Moreover, an action chosen for updating for a given state, according to the explanation above, is updated only if the present evaluation is better than the previous one already saved in state-action matrix.

The most problem (and even so not easy to solve it) in designing with RL is to determine the representation of the situations (states) and the attitudes to be made by the agent (actions), in order to solve the problem, or in this case to avoid an obstacle.

For robot navigation, the states and actions chosen were based on probable human representation of the situation and the decision made by them. The actions are simple: bounded velocities of the robot in two directions, in such a way that the robot can move in 2-dimensional space, or stop.

The states, which can represent a situation in obstacle avoidance, are more complex. However, when we humans do obstacle avoidance, we look into the environment, detect obstacles and extract important information based on their position and velocity with respect to us. The objective is to avoid an collision (two bodies in the same point and at the same moment), then the following information can be inferred from the environmental data: 1) if there is a possibility of collision; 2) where is the possible collision; 3) when is the possible collision. Based on these inferences, a proper decision is taken. In order to demonstrate the applicability of the states chosen, it is convenient to follow the steps below.

Imagine you in a collision avoidance situation. You are a robot in a space with one dynamic obstacle. For better understanding, it is shown this situation from a top view. The robot is the square and the unique obstacle is the circle. The arrow indicates the present direction of one body and if one body has no arrow, its direction is unknown. For the following situation, which direction would you take ?

Figure 1: Robot-Obstacle transaction where the obstacle's direction is unknown

The obstacle's direction is very important. For the four situations in Fig. 2, it is possible to take easily a correct decision.

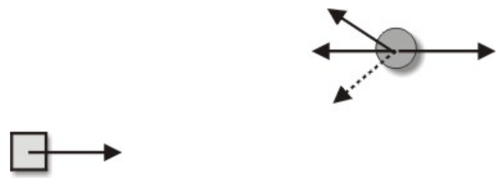

Figure 2: Robot-Obstacle transaction with the obstacle's directions known

For the obstacle's direction pointed by a dashed arrow, a collision is imminent, while for others directions, if no changes happen, it is not necessary a correction in robot's course.

As consequence of the direction of the obstacle, it is possible to say that more important than the robot-obstacle distance is the distance from the present robot's position and the probable point of impact. The figure 3 shows this importance.

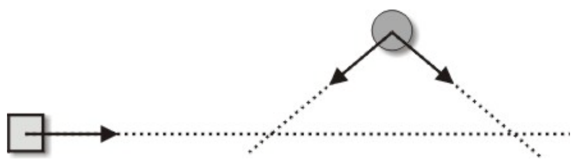

Figure 3: Two different distances to collision for the same obstacle's position

In this figure, for the same distance between the robot and the obstacle, there are two different points of impact, and possibly two different behaviors for collision avoidance. When the point of impact is nearer than another point, our preoccupation for collision avoidance is different too.

Another interesting consideration is how far the obstacle is from the robot's way. This can be seen as a degree of how free is the robot's way. If the obstacle is near from the presumable path of the robot, the path is less free than if it is far. Depending on how free is the robot's path, different avoidance may be performed. The Fig. 4 shows this situation.

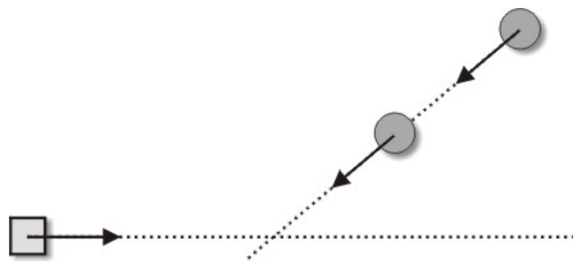

Figure 4: Two different configurations of freedom

The last important data useful for obstacle avoidance has relationship with time and it is dependent on both robot's and obstacle's velocities. Here, the most important is not to know the velocity values, but if the robot can arrive at the point of impact at the same time of the obstacle or if he arrives before or after it. If it is more possible to arrive before than the obstacle at the point of impact, then the decisions that brake the robot are less interesting.

So, in order to represent the items from section 2 and considering the description above, the four states below are used in this work:

- di: distance to the possible point of impact;
- beta: direction of an obstacle;
- y_dro: shortest distance between obstacle and the robot's path;
- ti: condition of arriving: if the robot will arrive before, at the same time or later related to the obstacle at the impact point.

However, one common problem with RL representation is the explosion of the number of states and actions, that means, too much number of states and actions are necessaries for a good representation. In this work, in order to decrease the amount of memory used to save the state-action matrix, a neural representation of this matrix is used. This representation permits to use fewer bytes compared to the matrix direct in memory storage.

Moreover, it is used also a scheme of neural networks (NN) in parallel, that can make a cooperative training. The following figure shows this scheme with two NNs., but if necessary, more NNs can be added to it. In this scheme Y is the output of the set of NNs, which is equal to sum of each NN. D is the desired output for the scheme and E is the output error of the set of NNs.

In this scheme, the training is a little bit special: in order to avoid the NNs make a dirty competition, in which one tries to destroy the other and so they can't construct a solution together, only one NN is trained, while the other NNs produce their results with their weights unaltered. This procedure has two great advantages: avoids dirty competition among the NNs and, therefore, no necessity of an element for coordination. In many schemes, in which more than one NN is used, a coordinator is necessary for the organization. The coordinator can be another NN or other element that can apply a linear or non-linear combination among the NNs' output. This scheme was already utilized in control of a flexible link [1]. An important reference on use of more than one NN can be found in [7].

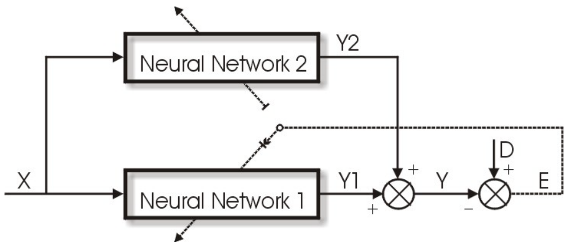

Figure 5: A scheme of two NNs in parallel

The scheme begins with only one NN and this is trained alone with a partial set of the training data. When the training can't advance, another NN is added to the scheme, whose outputs are zero for all inputs. This second NN is trained, while the first produces its output. If the training of the second NN can't advance too, it is possible to continue the training of the first NN, while the weights of the second NN become invariant, or another NN is added and the process repeats. This procedure causes another advantage of this scheme: when a NN is trained, it utilizes the knowledge acquired by the others NNs, that means, the training of each NN doesn't reject the knowledge already learned by the others NNs because it is used in the present training. The equations below show this. When the first NN was training, the output error was $E = D - Y1$ and the teaching signal for this NN was

$$Y1 = D \tag{2}$$

The output error is $E = D - Y1 - Y2$ with the second NN, and the teaching signal for this NN is

$$Y2 = D - Y1 \tag{3}$$

If the first NN is trained again, the second NN is not trained and the teaching signal for the first NN is

$$Y1 = D - Y2 \tag{4}$$

Or if another NN is added, and the teaching signal for the third NN is

$$Y3 = D - Y1 - Y2 \tag{5}$$

As it can be seen, the desired output of each NN is different from each one. Even when a NN is trained again, its teaching signal is different from that used in previous training.

4. Obstacle Avoidance Architecture

There are three different obstacle avoidance approaches: local (decision), global (planning) and for emergencies (failure). The local avoidance tries to avoid the most imminent collision. The global avoidance chooses an action that can be used for the local avoidance (not necessarily the best one), but that also avoids the others obstacles or has a greater chance to avoid them. However, if the action chosen, or the sequence of actions, drives to a collision, because the action was not properly chosen or the environment has another context, another type of actions is made considering the emergency situation. The action chosen has many times no relationship with the destination point. The goal in this moment is to avoid an immediate collision.

The avoidance above must combine with the path defined for the static environment. That means, there is a path defined *a priori*, which dictates the robot's movement in a well-known and static environment. And, in complement of this path, avoidance are made based on the pre-defined path and the dynamic obstacles condition. The following explanation clarifies how both systems can work together.

First, a path for the static environment is made. Normally, the algorithm for this case uses the initial (present) and final positions and velocities of the robot, as well as a map with description of free and occupied positions.

Then, the path produced feeds the dynamic avoidance system. For each time step, this system uses the present robot's and environment's states and the trajectory to be done by the robot at this time. Based on this information, it classifies the situation in one of the three possibilities: no collision, possible collision or collision. In this work, this classification is made by the following rule: if the robot's semi-line and the obstacles' semi-line have no intersection point, then the situation is classified as no collision. If there is at least one intersection point and the distance between the robot and the obstacle is inferior to 4 m, then there is a possible collision; and if the distance is less than 0.8 m, then the classification is collision. The figure below shows an example of possible collision. Here the semi-lines are drawn as dashed lines and the arrows indicate the direction of the objects.

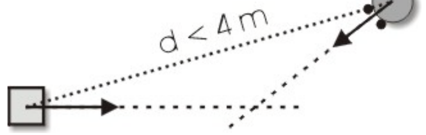

Figure 6: A classification of "possible collision"

For the first type of classification – no collision –, no problem is detected and the robot can follow the movement defined by the pre-defined trajectory. If the situation is classified as collision, then an escape path must be done. This escape tries to avoid an immediate collision, even if it follows an opposite direction relative to the desired destination.

However, if the classification is as possible collision, then avoidance must be done. The avoidance must consider the destination of the robot's path, the movement of the dynamic obstacles and the position of the static obstacles. Based on this information, a sequence of actions is taken, which can drive the robot to the destination through a safe path. After the collision is surpassed, generally another pre-defined path to the destination is necessary.

The sequence of actions is defined by the RL technique. During the training, the robot is put in diverse situation with only one punctual dynamic obstacle. After several trials, a classification of the actions for each set of states is available.

For the avoidance in a real situation, first three tracking points should be defined: left, central and right. These points define a non-punctual obstacle. In last figure, these points are the three black circles around the obstacle. The point to be avoided is the central point, while the others define the boundaries of the obstacle. The central point moves along the obstacle's surface, according to the avoidance taken by the robot. The central point movement continues until no more impact condition is encountered or until it is equal to one of the other two points, when the tracking points are redefined.

Trying to avoid a point, which moves to a boundary, performs the first case of solution of a complex problem using a combination of primitive solutions. The other case concerns the global avoidance. In this case, before an action is taken, it is verified the consequence related to the others obstacles. From the set of actions, that can avoid the central point, it is chosen one that has less or no consequences, that implies it has the longest duration to the point of impact or has no collision conditions for the other obstacles. It is possible that the action chosen is not the best action concerning to the central point, but for the other obstacles the future implications are small or none.

5. Simulations

In this work, only the local and for emergencies avoidance were implemented. Therefore, only the avoidance of one obstacle is presented.

The following figures present the avoidances tested in simulation. In these figures, the different faced colors indicate the different positions in time and a drawing of the robot and the obstacle with the same color indicates their position for the same instant. Hence, first they are in position indicated by the white color and go to the position indicated by black color. From one position, the time spent to the next position was 2 s. The arrows mean that after this position, they probably can follow the indicated direction. The goal in these simulations is to achieve the X point. The obstacle's velocity was 0.6 m/s in modulus.

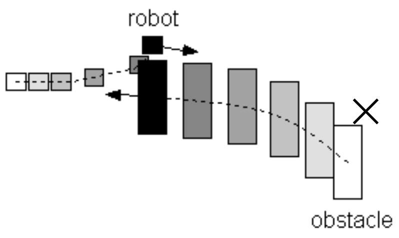

Figure 7: Obstacle avoidance – case 1

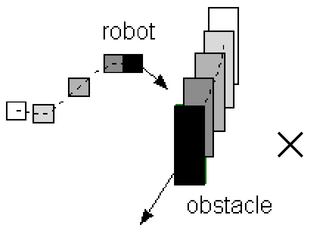

Figure 8: Obstacle avoidance – case 2

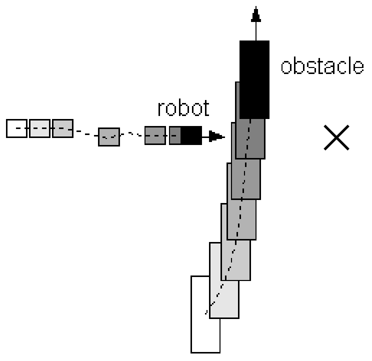

Figure 9: Obstacle avoidance – case 3

6. Conclusions and Future Works

This work showed learning-based methods for dynamic obstacle avoidance, which can avoid more complex obstacles based on primitive actions. These primitive actions were obtained with RL technique trained with Monte Carlo algorithm. A combination of primitive actions and a well-structured logic permitted the robot avoid obstacles didn't used during the training, like ones with different trajectories or/and dimensions.

However, more is still necessary. The concept presented here must be proved in situation with dynamic and static obstacles. The insistence of performing a solution for more complex problem utilizing primitive actions combined with some procedure will be tested. The success of avoidance of bigger obstacle, while the training was performed using only punctual obstacles, is a motivation of this strategy.

References

[1] A. de Almeida Neto, L. C. S. Góes and C. L. Nascimento Jr. Multiple neural networks in flexible link control using feedback-error-learning. In *Proceedings of the 16th Brazilian Conference on Mechanical Engineering*, v.1, pages 307-315, 2001.

[2] E. Freund, M. Schlusse and J. Rossmann. Dynamic collision avoidance for redundant multi-robot systems. In *Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System*, v. 3, pages 1201-1206, 2001.

[3] D. Gachet, M. A. Salichs, L. Moreno and J. R. Pimentel. Learning emergent tasks for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ/GI Int. Conference on Intelligent Robots and Systems '94*, v. 1, pages 290-297, 1994.

[4] D. H. Ingvar. Memory of the future: an essay on the temporal organization of conscious awareness. *Hum Neurobiol*, (4):127-136, 1985.

[5] H. Kawano and T. Ura. Dynamics control algorithm of autonomous underwater vehicle by reinforcement learning and teaching method considering thruster failure under severe disturbance. In *Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System*, v. 2, pages 974-979, 2001.

[6] J. Moody and M. Saffell. Learning to trade via direct reinforcement, *IEEE Transactions on Neural Networks*, (12):875-889, 2001.

[7] A. Sharkley. Combining artificial neural networks. In *Proceedings of the 6th Brazilian Symposium on Neural Networks*, 2000

[8] Z. Shiller, F. Large and S. Sekhavat. Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System*, v. 4, pages 3716-3721, 2001.

[9] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, Cambridge-USA, The MIT Press, 1998.

[10] P. Tang, Y. Yang and X. Li. Dynamic obstacle avoidance based on fuzzy inference and transposition principle for soccer robots. In *Proceedings of the 10th IEEE Int. Conference on Fuzzy Systems*, v. 2, pages 1062-1064, 2001.

[11] N. C. Tsourveloudis, K. P. Valavanis and T. Hebert. Autonomous vehicle navigation utilizing electrostatic potential fields and fuzzy logic, *IEEE Transactions on Robotics and Automation*, 17(4):490-497, 2001.

[12] G. Yen and T. Hickey, Reinforcement learning algorithms for robotic navigation in dynamic environments. In *Proceedings of the 2002 Int. Joint Conference on Neural Networks*, v. 2, pages 1444-1449, 2002.

[13] J. M. Zurada. *Introduction to Artificial Neural Networks*, West Pub. Co., New York, 1992.