Reactive Vision-Based Navigation Controller for Autonomous Mobile Agents

Ruben Nuredini
Department for Software Engineering, Heilbronn University of Applied Sciences, Heilbronn, Germany.

Corresponding author: Tel.: +49 7131 504 236; email: ruben.nuredini@hs-heilbronn.de
Manuscript submitted January 10, 2017; accepted April 8, 2017.
doi: 10.17706/jcp.13.2.161-167

Abstract: Initial results of an ongoing research in the field of reactive mobile autonomy are presented. The aim is to create a reactive obstacle avoidance method for mobile agent operating in dynamic, unstructured, and unpredictable environment. The method is inspired by the stimulus-response behavior of simple animals. An obstacle avoidance controller is developed that uses raw visual information of the environment. It employs reinforcement learning and is therefore capable of self-developing. This should result with obstacle avoidance behavior that is adaptable and therefore generalizes on various operational modalities. The general assumptions of the agent capabilities, the features of the environment as well as the initial result of the simulation are presented. The plans for improvement and suitable performance evaluation are suggested.

Key words: Intelligent mobile agent, reactive obstacle avoidance, reinforcement learning.

1. Introduction

One of the most challenging characteristics of mobile agents is their autonomous operation. The key feature that leads to autonomous mobile agents is the autonomous navigation ability. It is of "eminent necessity since, by definition, a robot accomplishes its tasks by moving in the real world [1]".

Creating an autonomous navigation system involves building and synchronizing separate modules that tackle system's specific subtasks: mapping – deals with creating the map of the surroundings; localization – used for estimation of the current position and orientation of the mobile agent; and path planning – or breaking down a desired movement task into a sequence of discrete motion actions. When operating in complex and dynamic environments, such as the real world, the path planning module should detect the presence of many obstacles that are not stationary and avoiding collisions with them. Hence, obstacle avoidance (OA) is a subproblem of path planning that aims to compute a motion control sequence that is free of collisions. It arises when the path planning task is performed in an unstructured and non-static environment and either: a) the full map of the environment is not available to the agent, or b) the dynamics of the environment is not perfectly predictable a priori.

Due to the complex and hazardous nature of this task, OA requires complex decision schemes which involve large number of parameters. Typical for classical robotics are manually derived schemes that use hand programmed algorithms that turned out to be a very tedious and impractical process. Such methods are applicable to one operational modality, are not directly portable to others, and do not generalize well.

An ideal OA method should be able to detect obstacles and produce appropriate reaction in form of an evasive motion command. It should also not be computationally intensive and able to operate in real-time. It

should be easily adaptable and capable of generalizing to new environments and operating conditions.

Natural evolution provided simple animals with skills whose qualities are considered to be essential for every modern artificial system. Their behavioral repertoires display goal-oriented, adaptive, opportunistic, plastic, and robust behaviors. This observation is the basic motivation of the presented method - using the features of the animal's stimulus-response reflexive behavior and applying it to the senso-motoric loop of an OA controller.

This paper presents a novel method for solving the OA task in dynamic, unstructured, and unpredictable environment as well as the results of the early phase of experimentation. The method is centered around an intelligent and self-developing controller that maps sensory information to motoric actions. Several biologically plausible principles were used as inspiration for the design of the controller. The Gibsonian principle of direct perception [2] which claims that "the pattern of light reflected from these surfaces, provides adequate information for controlling behavior without further inferential processing or model construction" was inspiration for utilization the visual projection as the sole input to the controller. The main idea of ethology that "the behavior of each animal is adapted to its surroundings and is adjusted to the requirements of its survival" is the inspiration for providing the controller with the ability to learn the mapping between projection and a suitable action while operating in its environment. The instrumental conditioning learning paradigm that "observes the learning process through which an animal's behavior is changed by the consequences, or results, of that behavior" was the reason for using reinforcement learning algorithm. Such algorithm will enforce the agent's learning process by self-developing a control policy in order to maximize the notion of long-time reward over a period of time.

Section 2 summarizes the most noteworthy and elaborated previous studies on this topic. Section 3 contains a short introduction to reinforcement learning algorithm as well as artificial neural networks. In Section 4 the main assumptions of the problem and the key features of the simulation are presented. The challenges and prospects for further development of the method are presented in the conclusion.

2. Previous Work

Taking inspiration from nature for designing man-made artifacts has been practiced since centuries. Moravec [3] presented one of the initial works in which he eloquently highlighted the importance of mobility in natural animals as he argued that mobility affected the process by evolution. His claim that "mobility inevitably led to intelligence" led to a very radical change in how researchers approach mobility. However, he also stated that "problems that require high-level reasoning are easy to be solved by computers" but "low-level senso-motorical skills typical for living beings require enormous computational power" – a claim that pointed out the non-triviality of employing biologically plausible principles in artificial intelligence. Brooks added on the idea by introducing a new methodology that was based on analogy with natural evolution that allows design of "self-sustaining" mobile agents operating in real-time fashion in real-world environment. This set the course of for a new paradigm of using animal behavior as a basis for development of control strategies for mobile robots. The paradigm builds on the observation that simplest animals already exhibit most of the control properties essential for mobile agents.

One of the earliest attempts for autonomous navigation inspired by the earthworm's simple navigation system is the Rob-Nav system [4]. Simple navigation routines were hierarchically organized and linked by failure defaults. Succeedingly more difficult problems would invoke more sophisticated routines only on failure of the lower levels. The algorithm did not require object identification and planning and was therefore sensor-independent.

The idea of organizing simple tasks in a hierarchical structure by combining them in order to produce advanced task-achieving robot behaviors was elaborated with the concept of "subsumption architecture" [5]

– a layered and distributed control architecture. Similar approach is the behavioral schemata which combines primitive behaviors in order to yield a more complex behavior. This concept was successfully adopted in context of autonomous control for navigation task. The theory that animals possess inherited responses to certain situations and combine them with the ability to adapt to situations for which no response is available is applied in a layered architecture of predefined stimulus/response mechanisms [6]. Concurrent activation of primitive reflexive behaviors caused the mobile agent to display emergent behaviors such as wandering, simple navigation, perimeter following etc.

Bio-inspired robotics embodied itself as a sub-discipline of robotics that considers applying biologically plausible principles to autonomous robots. It deals with building computational models of individual biological features and behaviors as well as physiological and evolutionary processes of humans and simpler natural animals. Notable examples of bio-inspired behavior modeling build on the neurobiological assumption that perception and action (visuo-motoric) are coupled system that has to be investigated in a closed loop [7]. In [8] OA was achieved through simultaneous adaptation of viewing direction of the sensors and the sensorimotor coupling. The agent's visual input was processed by motion detectors further coupled via transmission weights to the motors. The weighted connection between sensory input and motor output were optimized by genetic algorithm (GA). Another line of work explores the concept of artificial evolution for development of behavioral abilities without any constrains on the architecture and functioning modalities [9]. Vision is used as an input to a network of spiking neurons employing GA to evolve it. The spiking controller was capable to perform OA task in lab environment.

Strategies for OA inspired by features of insects' visual systems based on mapping monocular optic flow into control signals [10], [11], and depth estimation from 3D reconstruction of images acquired by a modular stereo-vision [12] were also presented.

3. The Approach

No previous work utilizing raw visual data and reinforcement learning for a self-developing controller for reactive OA was encountered. Using raw data causes high-dimensional input space imposing one inherent problem of reinforcement learning when employed in real-world environments. "The curse of dimensionality" causes vast memory and computational requirements. Function approximation techniques are typically employed for overcoming it. The proposed method uses ANNs due to their ability to generalize learning across similar states.

3.1. Reinforcement Learning

Reinforcement learning (also referred to as Neural-Dynamic Programming [13]) is a sub-area of machine learning that describes a class of computational and mathematical approaches and methods for solving similar problems. It deals with agents and how they ought to take actions in order to maximize some notion of long-time reward or minimize costs over a period of time. The central idea of reinforcement learning is that agents can be programmed by reward and punishment without the need to explicitly detail how the task should be achieved. Generally speaking, reinforcement learning represents the mutual interaction between an agent an environmental system. In the standard reinforcement learning model the agent is connected to the environment via perception and action. With each interaction the agent receives input i which is a depiction of the current state s of the environment. The agent then takes action a and the environment changes its state to s'. The outcome of the transition between states is then communicated to the agent in form of reinforcement signal r.

There are three classes of methods to solve reinforcement learning problems all based on the notion of a value function: Dynamic Programming (DP), Monte Carlo (MC), and Temporal Difference (TD) methods [14]. TD-methods are inspired by the associative learning of natural animals. They combine the advantages of DP

and MC and are model-free as the updates are performed on each time step. This makes them suitable for non-episodic tasks in partially unknown systems. Q-Learning is the most popular temporal difference algorithm that utilizes the state-action value function $Q(s,a)$. SARSA (State-Action-Reward-State-Action) [15] is an on-policy version of Q-Learning since it learns state-action values relative to the policy it follows. Here, the value of the next state is the Q-value associated with the action chosen with an ε-greedy strategy at next time step $t+1$. If the experience tuple (s,a,r,s',a') is considered, the update rule for the SARSA algorithm is:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma Q(s',a')) \quad (1)$$

where r is the immediate reward for taking action a in state s; γ is the discount factor $(0 \leq \gamma \leq 1)$; $Q(s',a')$ is the Q-value of the state s' when taking action a'; $Q(s,a)$ the previous Q-value; and α the learning rate.

3.2. Artificial Neural Networks

Artificial neural networks (ANN) are information-processing systems which are inspired from biological neural networks and have been evolved as generalizations of mathematical models of human cognition or neural biology. ANNs can be represented as a massive system of parallel distributed processing units connected in graph topology. They are based on the following four assumptions: a) information processing occurs at many simple elements called neurons; b) signals are passed between neurons over connection links; c) each connection link has an associated weight, which multiplies the transmitted signal; and d) each neuron applies an activation function to its net input to determine its output signal. ANN is characterized by its structure, the learning algorithm, and the activation function. In our work we employ feed-forward neural networks (FNN), the backpropagation learning algorithm (BP), and the sigmoid activation function.

BP algorithm is a gradient-descent error-correction algorithm that minimizes the errors between the desired outputs and the actual computed outputs by modifying the weights between the units in the network. It requires a readily available dataset consisting of input/output pairs or a teaching signal or correct response in the form of human output to a specific input. The learning with backpropagation algorithm consists of two phases. First, the network is presented with an input and the activation is propagated forward through the hidden layers of the network up to the output layer where the actual output is calculated. This determines the network's response.

In the backpropagation phase, the network's response is compared with the known correct response. If the network's actual response does not match the correct response, the parameters of the network are slightly tuned to produce a response more closely matching the correct response. This is done by modifying the weights and the internal thresholds by using the errors between the desired and the actual output. The two phases (feed-forward and back-propagation) are repeated until the errors are below some threshold.

3.3. SARSA and Backpropagation

Despite the fact that temporal difference (TD) learning methods (such as SARSA) have no inherent connection to ANN architectures, they can still be combined with backpropagation. By this the temporal credit assignment problem solving capability of TD is combined with the structural credit assignment problem solving capability of BP. An additional benefit is the ANN capability to generalize the learning across similar states. This way the agent can flexibly learn to maximize reward over multiple time steps and also learn structural similarities in input patterns that allow it to generalize its predictions over novel states.

4. Method Implementation

4.1. General Assumptions

The proposed method in this work has been designed under several assumptions. The agent has no self-knowlegde such as the arrangement of the sensors or its physical dimensions. It is equipped with visual sensor used to obtain the projection from its immediate proximity as a sole input of the environment. It also moves with uniform speed in one direction. The environment is an endless 2-D corridor–like plane that is filled randomly positioned obstacles. The obstacles are represented by a closed curve with finite length and are filled with color that allows to discriminate them from the environment. No discrimination of certain objects in the environment on the basis of their semantic significance is done. There is no exact model of the environment. The environment is assumed to be partially observable to the agent. The task is continuing and runs to infinity without goal position that the agent aims to reach. The agents goal is to maximize the distance (time) without collision.

4.2. Simulation

A typical way to offset the cost of real-world interaction is to use approximate models as simulators that whoul ideally allow the agent to learn in a simulated environment and then subsequently transferred this behavior onto a robot controller. Creating a sufficiently accurate model of the robot's real world is challenging since it often needs high amount of data samples. Small errors in the simulated model can accumulate and result with a behavior that is not suitable for the real environment. Usually, significant modifications have to be made to the controller before it is ready to be ported on the robot.

The simulation consists of a real-time ANN-based controller created with PyBrain [16] that uses SARSA algorithm as update rule. PyBrain is highly customizable as it offers the possibility to connect the ANN controller to an agent operating in a custom environment. To test the initial performance of the proposed method a game-like environment and agent was created using PyGame (Fig. 1). The simulated agent is depicted by a single pixel point and operates in its world. The obstacles are rectangular in shape and are randomly generated. As the agent moves, in every discrete time-step it scans its immediate surroundings using its visual sensor. The visual sensor sampling is of equally distributed rays whose angular resolution is customizable.

The color value of the inverse retinal projection is provided as input to the ANN. The output layer produces one of the actions of the action space of the agent. The winner takes it all strategy is used for selecting the left, right, or stay actions. The weights of the ANN are updated according to the previously discussed update rule.

Initial experiments were performed with: (1) a three-layered fully connected FNN with: 16 units in the input layer (corresponding to the visual sensor resolution), 10 units in the hidden layer, 3 units in the output layer (corresponding to the agent's action space); (2) a SARSA learner employing a linear greedy explorer with: α=0.5, λ=0.9, and γ=0.9; and (3) a reward function with: +0.1 for each step without collision, and -1 for each collision.

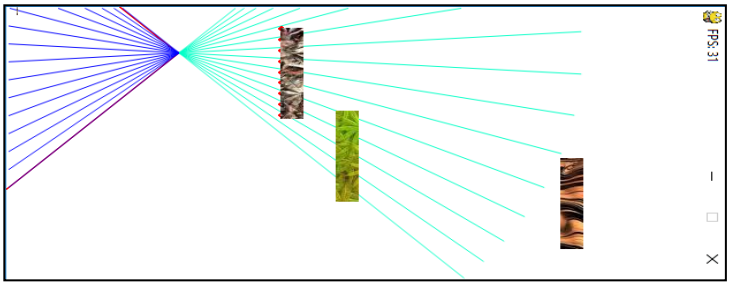

Fig. 1. The simulation environment.

5. Conclusion

In the initial learning stage the agent displays explorative behavior as it moves without any noticeable pattern causing multiple collisions with the obstacles. As the exploration decreases, the agent movement is more dictated by the learned policy. The agent exploits the fact that least obstacles are generated near the edges of its environment as it remains there with occasional movement caused by its explorative nature.

After several minutes the agent displays better performance in avoiding obstacles even in complicated situations. However, a suitable metrics needs to be introduce in order to correctly evaluate the performance of the method. One problem is that no standardized tool for performance metrics, benchmark database, or comparison methodologies for OA methods exists. Despite several efforts [17]-[20] by this date there is no universally accepted framework for performance evaluation of OA. The main idea behind one should be measuring the crucial quantitative parameters of the solution given the precise working conditions.

The good results obtained during the experimental phase are very encouraging to further develop the simulation environment and experiment with carious settings. The first step is to introduce a suitable evaluation process that will combine several performance measurements. The evaluation process should consider the aspects such as: detailed description of the assessment procedure, specification of the agent used in the experiments, the parameters of the metrics, and the constraints in the task. This will provide the means to evaluate the impact that various learning parameters have in the overall success of the method.

The agent's physical features are also subject to experimentation. The primary focus is set on comparing the performance by altering between mono and binocular vision and comparing the corresponding OA performance of the agent. This will allow exploring whether the controller will be able to develop capability for depth estimation based optical flow or triangulation. The underlying architecture of the neural network should be restructured in order to capture and match the setup of visual sensor. The optimal size of the hidden layer should also be determined. It is usually done empirically by initially assigning an arbitrary large number of nodes to the hidden layer, studying the convergence of the system during the learning process, and reducing the number of nodes in the hidden layers to see whether or not the network still converged with the reduced number of nodes. Future development should result with transferring all the experimentation in a real context.

References

[1] Latombe, J. C. (1991). *Robot Motion Planning.* Kluwer Academic Publishers.

[2] Gibson, J. (1979). *The Ecological Approach to Visual Perception.* Boston: Houghton Mifflin.

[3] Moravec, H. P. (1976). *The Role of Raw Power in Intelligence.*

[4] Cahn, D. F., & Phillips, S. R. (1975). ROBNAV: A range-based robot navigation and obstacle avoidance algorithm. *IEEE Trans. Systems, Man, and Cybernetics, 5,* 544-551.

[5] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, 2(1),* 14-23.

[6] Anderson, T. L., & Donath, M. (1990). Animal behavior as a paradigm for developing robot autonomy. *Robotics and Autonomous System, 6(1-2),* 145-168.

[7] Reynolds, C. W. (1993). An evolved, vision-based model of obstacle avoidance behavior. *Proceedings of the 2nd International Conference on Simulation of Adaptive Behavior.*

[8] Neumann, T. R., Huber, S. A., & Bülthoff, H. H. (1997). Minimalistic approach to 3d obstacle avoidance behavior from simulated evolution. *Proceedings of Conference on Artificial Neural Networks.* Switzerland: Lausanne.

[9] Floreano, D., Epars, Y., Zufferey, J.-C., & Mattuissi, C. (2006). Evolution of spiking neural circuits in autonomous mobile robots. *International Journal of Intelligent Systems, 21,* 1005-1024.

[10] Beyeler, A., Zufferey, J.-C., & Floreano, D. (2009). Vision-based control of near-obstacle flight. *Autonomous Robots, 27(201),* 201–219.

[11] Milde, M. B., Bertrand, O., Benosman, R., Egelhaaf, M., & Chicca, E. (2015). Bioinspired event-driven collision avoidance algorithm based on optic flow. *Proceedings of International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP).* Poland: Krakow.

[12] Chessa, M., Murgia, S., Nardelli, L., Sabatini, S. P., & Solari, F. (2014). Bio-inspired active vision for obstacle avoidance. *Proceedings of Conference on Computer Graphics Theory and Applications.* Portugal: Lisbon.

[13] Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming, Belmont.* MA: Athena Scientific.

[14] Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning.* Cambridge: MIT Press.

[15] Rummery, G. A. (1995). *Problem Solving in Reinforcement Learning.* Cambridge University Press.

[16] Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstie, T., & Schmidhuber, J. (2010). *Journal of Machine Learning Research, 11,* 743-746.

[17] Freire, E. O., Carvalho, E. A., Alves, M. V., Montalvão, J., & Molina, L. (2010). Human based benchmark for robot navigation assesment. *Proceedings of ISSNIP Biosignals and Biorobotics Conference.* Spain: Vitória.

[18] Calisi, D., & Nardi, D. (2009). A unified benchmark framework for autonomous mobile robots and vehicles motion algorithms (MoVeMA benchmarks). *Autonomous Robots, 27,* 465–481.

[19] Jimenez, J. L., Rano, I., & Minguez, J. (2007). Advances in the framework for automatic evaluation of obstacle avoidance methods. *Proceedings of Irons Workshop on Benchmarks in Robotics Research.*

[20] Muñoz, N. D., Valencia, J. A., & Londoño, N. (2007). Evaluation of navigation of an autonomous mobile robot. *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems.*

Ruben Nuredini obtained the master of information & communication technologies at Faculty of ICT, FON University in Skopje in 2011. He earned his bachelor degree at the CST Faculty at SEEU, Tetovo in 2007. Currently, he works as a teacher and research associate in the Faculty for Informatics, Heilbronn University, Germany. He is active in research related to his PhD thesis in the field of adaptive mobile autonomy. His other research interest fields include machine learning, intelligent systems, parallel programming, application of CS in robotics and mobile systems, computational neuroscience. He published several research papers in international conferences and journals.