Adaptive, Neural Robot Control – Path Planning on 3D Spiking Neural Networks

Lea Steffen(✉), Artur Liebert, Stefan Ulbrich, Arne Roennau, and Rüdiger Dillmannn

FZI Research Center for Information Technology, Karlsruhe, Germany
{steffen,liebert,stefan.ulbrich,roennau,dillmann}@fzi.de

Abstract. Safe, yet efficient, Human-robot interaction requires real-time-capable and flexible algorithms for robot control including the human as a dynamic obstacle. Even today, methods for collision-free motion planning are often computationally expensive, preventing real-time control. This leads to unnecessary standstills due to safety requirements. As nature solves navigation and motion control sophisticatedly, biologically motivated techniques based on the Wavefront algorithm have been previously applied successfully to path planning problems in 2D. In this work, we present an extension thereof using Spiking Neural Networks. The proposed network equals a topologically organized map of the work space, allowing an execution in 3D space. We tested our work on simulated environments with increasing complexity in 2D with different connection types. Subsequently, the application is extended to 3D spaces and the effectiveness and efficiency of the used approach are attested by simulations and comparison studies. Thereby, a foundation is set to control a robot arm flexibly in a workspace with a human co-worker. In combination with neuromorphic hardware this method will likely achieve real-time capability.

Keywords: Cognitive robotics · Neural motion control · Spiking Neural Networks · Wavefront algorithm

1 Introduction

Nowadays, robots are crucial for the production in several major industries. Until recently, robots were applied isolatedly, enabling work with exclusively predefined paths. As product individualization and diverse product needs increase, the production moves away from repetitive high precision tasks evolving into more complex processes. Additionally, the demand for human-robot interaction requires flexible and real-time capable robot control, considering humans as dynamic obstacles, to meet safety precautions. State-of-the-art algorithms following the Sense-Plan-Act cycle do not meet these requirements. Nature's sophisticated manner of fast and reactive motion control has been an inspiration to scientists for decades. Hence, it is not a new idea to use Artificial Intelligence (AI), and more precisely, Artificial Neural Networks (ANN), to solve path

I. Farkaš et al. (Eds.): ICANN 2020, LNCS 12397, pp. 509–520, 2020.
https://doi.org/10.1007/978-3-030-61616-8_41

planning [9] and also motion planning problems [23] for robotics. However, it is still an active field of research as well in navigation [19] as in motion control [3,4,18]. Algorithms creating collision-free motions as the A* algorithm [12], Rapidly Exploring Random Trees [16] (RRT) and the Wavefront Algorithm [15] already exist but are far from realtime-capable on conventional hardware due to extensive computing times caused by high complexity.

The fundamental concept of path planning, motion control and navigation in humans is formed by the *Place Cells* [14] discovered in 1971 by O'Keefe. These cells are located in the hippocampus. If an organism enters a particular location, the respective place cells fire simultaneously. The location responsible for this impulse is called *Place Field*. Place Cells depict the environment as a cognitive map [21], which is resistant to rotation and changes in lightning, and therefore not relying on visual input. ANNs, are usually applied to model an artificial system for robotic control but recently, also Spiking Neural Networks (SNN) are used [1]. A major advantage is their massive parallelism, but to benefit from it, dedicated hardware, referred to as neuromorphic, is needed to compute the underlying differential equations efficiently. Neuromorphic hardware has already been an active research topic for years, as shown by the development of SpiNNaker [6]. However, as of recently, companies like Intel (Loihi [2]) and IBM (TrueNorth [13]) have also invested in this technology.

Neural adaptations of the Wavefront algorithm using SNNs have been applied successfully to path planning problems in 2D [8]. In this work, we present an extension of this approach where the network represents a topologically organized map of the navigational space of a robot cell in 3D. This system forms the basis of a reactive real-time capable technique to control a robot arm flexibly considering humans and, in general, static as well as dynamical obstacles.

2 Related Work

Already in 1995, a Hopfield-type ANN has proven as effective for path planning and obstacle avoidance [7]. A biologically plausible alternative is presented in [22], whereby random exploration is used to learn the state space. However, there is a lack of neural path planning algorithms in 3D as up until now neural path planning has mostly been applied to 2D surroundings. Our method is based on the work of Qu et al. [8], a mathematically profound technique using SNNs. Nonetheless, there are several related methods employing a similar network architecture. [11,17,24] cover a wide spectrum from simplified to complex and biologically plausible neuron models. Furthermore, they differ in how the membrane potential is determined, weight adaptation and path calculation.

Most methods introduced in this section rely on Spike-time-dependent Plasticity (STDP), a neurobiological process that regulates the strength of synaptic connections in the brain [10]. If an incoming spike into a neuron occurs immediately before the neuron's own spike, STDP amplifies the respective synapse's weight. If the incoming spike into a neuron occurs immediately after its own spike, the synapse responsible for the incoming signal gets weakened. The process thus reinforces the relevance of the inputs potentially responsible for the

excitation of the postsynaptic neuron. As a consequence, such inputs will have a higher impact on the postsynaptic neuron in the future.

In [17], Ponulak et al. introduce an approach for parallel path planning using the Wavefront algorithm and neural plasticity. The neurons of the applied SNN are organized as a 2D topological map and represent biological Place Fields. In this method, the environment has to be learned before path planning is executable. Hence, an initial exploration phase creates a cognitive map of the surroundings by strengthening neurons which represent nearby locations through STDP. Afterwards, a *neural wave*, travelling the entire network, is initiated by activating the neuron representing the target location. Synapses are strengthened by anti-Hebbian STDP [5]. As the wave travels from the goal to the start position and anti-Hebbian STDP strengthens weights in the opposite direction, the optimal path can be determined by retracing the strongest synapses from the start to the target neuron. The network's architecture and synaptic connections are similar to our approach, but everything else is not. In particular, the calculation of the membrane voltage, and how synaptic connections are altered is different. Our approach and [8] determine the optimal path by following the parent of every neuron. In [17] the path is found by following the strongest synapse weights, which are a result of learning with STDP.

Another method for neural path planning based on Place Cells and cognitive maps was presented in [24]. Also here, impulses travel wave-like through a 2D network. The unique feature of this work is an additional same-sized network, referred to as the *occupancy map*. Synaptic connections, between a neuron of the main network and the occupancy grid are only established if they cover the same part of the environment. These connections are either inhibitory, in case of an obstacle, excitatory in case of a robot or neutral for empty spaces. Learning, and thus updating the synaptic weights is achieved by STDP. The similarities of this method and our work lie in the network architecture. They differ through the application of an occupancy grid and STDP as the learning rule in [24].

A scale-free navigational approach for planning by neural waves was introduced in [11]. The navigational space is represented by a topological graph where exciting synapses connect neurons of the free space representing portions of the surrounding which are close to each other. Neurons representing obstacles are isolated. What distinguishes this technique the most is that each neuron is excited periodically. The target neuron is excited with a higher frequency than the others, influencing their respective frequencies. The optimal path emerges along the phase-shifted frequencies.

3 Methodology

The core idea of our approach is mostly based on the work of [8]. However, our work differs to [8] in several ways. The nature of the network is merely outlined mathematically in [8], hence we used our own implementation to generate a neural representation of the environment but kept in line with the mathematical features of [8]. Furthermore, it is not stated in [8] how neighbor neurons are

determined. We solved this issue initially with Euclidean vector metric to ensure an easy transition from 2D to 3D environments. As we managed to reduce the generation time by using a direct mapping we replaced the Euclidean vector metric with that. To boost the performance even more, we used precise instead of equidistant time steps. The time steps are adapted to the occurrence of spikes, thus the network is only simulated if spikes are emitted. The method of [8] uses the differential equation in Eq. (7) to calculate the internal activity of every neuron in every single time step. Hence, even intern activities of neurons which have not had an input yet are calculated resulting in $U(t) = 0$. In our work we assigned the value of neurons which had no input yet to zero. Another aspect that differs from the method of [8] is that in their method the simulation iterates over every single neuron in every single time step. In contrast, we delete neurons which have already spiked which reduces the simulation time massively. Finally, we want to emphasize the most important difference, that there is no execution on 3D environments in [8].

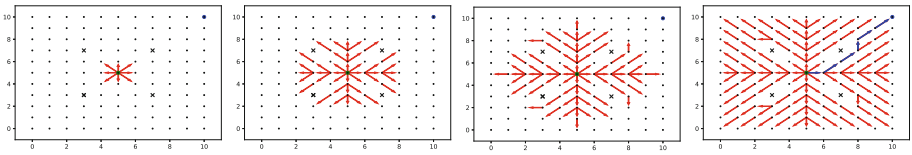
3.1 Network Architecture and Synaptic Connections

The network's finite set of neurons is called N. Every neuron $i \in N$, used for our system is a modified pulse-coupled neuron. We will describe their specific features in more detail in Sect. 3.2. Our algorithm operates on a discrete topologically organized map which is arranged as a 2D or 3D grid with solely local lateral connections among neurons. The synaptic weight between the neuron i and j is called ω_{ij} and is corresponding to the Euclidean distance of the neurons. Hence $\omega_{ij} = \sqrt{2}$ if i and j are diagonally connected and $\omega_{ij} = 1$ if i and j are crosswise connected. As all weights are symmetrical, $\omega_{ij} = \omega_{ji}$, the system with its synaptic connections corresponds to an undirected graph. While neurons representing free space are connected to their neighbors, neurons representing obstacles are isolated. Two network structures are possible, the Manhattan method and the Chamfer Method. The former only considers direct neighbors, in contrast to the latter, which additionally takes diagonal neighbors into account. It is noteworthy that, in 3D, the diagonal connections are alongside two axes, meaning they run on the convex hull of the 'cube' between two neurons.

3.2 Neural Features and Information Processing

Every neuron i has a set of neighbors R_i which is expressed by $R_i = \{j \in N, j \text{ is connected to } i\}$. This set can be separated into two subsets $R_i^r = \{j \in R_i \mid \omega_{ij} = \sqrt{2}\}$ and $R_i^l = \{j \in R_i \mid \omega_{ij} = 1\}$. R_i^l contains every neighbor connected crosswise to i and R_i^r holds all diagonal neighbors of i. A neuron i is said to fire at time $T \geq 0$, if $\exists \epsilon \geq 0$ such that

$$Y_i(t) = \begin{cases} 0, & \text{if } T - \epsilon \leq t < T \\ 1, & \text{if } t = T \\ 0, & \text{if } T < t \leq T + \epsilon \end{cases} \tag{1}$$



(a) $t = \sqrt{2}$ (b) $t = 1 + 2\sqrt{2}$ (c) $t = 5$ (d) $t = 2 + 4\sqrt{2}$

Fig. 1. Expansion of a neural wave in the network at different stages from the initiation in (a) to the termination in (d). The start neuron is marked in green while the target neuron and the resulting path are indicated in blue and obstacles are marked by an x. The parameter t states the number of time steps passed since the wave's onset. The time is unit less as explained in Sect. 4. (Color figure online)

The firing time is t^i_{fire}. The output function $Y_i(t)$ and neurons' output Y_i are explained in more detail later on.

To describe the internal neural activity, we need to introduce an additional concept. The first neighbor of neuron i that fires is denoted as R^F_i. It is referred to as the *pseudo parent neuron* of i. If a spike emitted by i is a direct consequence of the stimulation from R^F_i the pseudo parent becomes the *parent neuron* R^P_i of neuron i. However, if another neighbor j spikes after the initial pseudo parent R^F_i causing an earlier firing event of neuron i, then j is called the new pseudo parent neuron R^F_i. The entirety of every potential parent of a neuron i at time t is called the *changing set* $\xi(t, i)$. This set is time-dependent, at first it contains every neighbor of i. When t approaches t^i_{fire} the changing set $\xi(i, t)$ empties.

Auxiliary fields are needed to describe the internal activity $U_i(t)$ of a neuron i. The linking field $L_i(t)$ and feeding field $F_i(t)$ of each neuron are expressed as

$$L_i(t) = f(Y_{r^1}, \cdots, Y_{r^k}, t) = \begin{cases} 0, & \text{if } t < t^{R^F_i}_{fire} \\ 1, & \text{else} \end{cases} \tag{2}$$

$$F_i(t) = -g(\omega_{ir^1}, \cdots, \omega_{ir^k}, t)U_i(t) \tag{3}$$

where $\omega_{ir^1}, \cdots, \omega_{ir^k}$ are linking strengths from neuron i to every k neighbors. The internal activity of each neuron determines if a spike is emitted and can be solved via an initial value problem

$$\begin{cases} \frac{dU_i(t)}{dt} & = F_i + CL_i = -g(\omega_{ir^1}, \cdots, \omega_{ir^k}, t)U_i(t) + CL_i, \text{ for } t \geq t^{R^P_i}_{fire} \\ U\left(t^{R^P_i}_{fire}\right) & = 0 \end{cases} \tag{4}$$

where C is a positive constant and $g(\cdot)$ is a function which uses connection weights from neighboring neurons and the time as an input. It has a positive output. The function g can be depicted as

$$g_i(\omega_{ir^1}, \cdots, \omega_{ir^k}, t) = \begin{cases} 0, & \text{if } t < t^{R^F_i}_{fire} \\ \mu(\omega_{ij}), & \text{if } t \geq t^j_{fire} \text{ for a } j \in \xi(i, t) \end{cases} \tag{5}$$

where $\mu(\omega_{ij})$ is given by

$$\mu(\omega_{ij}) = \frac{B}{\omega_{ij}} = \begin{cases} B, & \text{if } j \in R^l \\ \frac{B}{\sqrt{2}}, & \text{if } j \in R^r \end{cases} \tag{6}$$

In this case, B is also a positive constant. The particularity of the system is that if the parent of neuron i changes, the intern activity U_i of i is reset to 0 and the process of solving the changed differential equation starts all over again.

The final differential equation for the internal activity $U_i(t)$ for a neuron i results from this derivation and is expressed as

$$\begin{cases} U_i(t) = 0, & \text{if } 0 \leq t < t_{fire}^{R_i^F} \\ U_i(t) = 0, & \text{if } t = t_{fire}^{j} \text{ for any } j \in \xi(i,t) \\ \frac{dU_i(t)}{dt} = -\mu(\omega_{iR_i^P})U_i(t) + C, & \text{if } t \geq t_{fire}^{R_i^P}. \end{cases} \tag{7}$$

Since the membrane potential behaviour of a neuron has been described, the next step is to formulate the process of spiking.

It requires a output function $Y_i(t)$ of a neuron i which has the time t as input and compares the internal activity $U_i(t)$ of a neuron i with its threshold function $\theta_i(t)$ at time t as follows

$$Y_i(t) = \text{Step}(U_i(t) - \theta_i(t)) = \begin{cases} 1, & \text{if } U_i(t) \geq \theta_i \\ 0, & \text{else} \end{cases} \tag{8}$$

The threshold function $\theta_i(t)$ is expressed as

$$\theta_i(t) = \begin{cases} A_{init}, & \text{if } t < t_{fire}^{R_i^F} \\ A_{ij}, & \text{if } t_{fire}^{j} \leq t < t_{fire}^{i},\ j \in \xi(i,t) \\ V_\theta, & \text{if } t \geq t_{fire}^{i} \end{cases} \tag{9}$$

for all i, where A_{init} and V_θ are real-valued, positive constants. V_θ is set to be a very large value, such that every neuron will not spike a second time, once they have already spiked. A_{ij} is expressed as

$$A_{ij} = \begin{cases} A^r, & \text{falls } j \in R_i^r \\ A^l, & \text{falls } j \in R_i^l \end{cases} \tag{10}$$

where A^r and A^l are also real-valued, positive constants. A_{ij} describes the changing value of the threshold which depends on the type of connection of i to its neighbor j.

The fundamental difference to a pulse coupled neural network (PCNN) [20] lies in the structure of the threshold function. In modified PCNN (MPCNN) the threshold function can receive simulated inputs from neighboring neurons, which can then artificially change their value. This is not possible with PCNN. The presented neuron model is described in its entirety from the output function, the threshold function, the linking- and feeding fields, g, μ and the final form of the differential Eq. (7). However, a special case has to be considered here. If more than one neuron fires at the same time $t_{fire}^{R_i^P}$, the neuron with the lowest connection strength is selected as the parent neuron R_i^P.

3.3 Expansion of the Neural Wave and Path Calculation

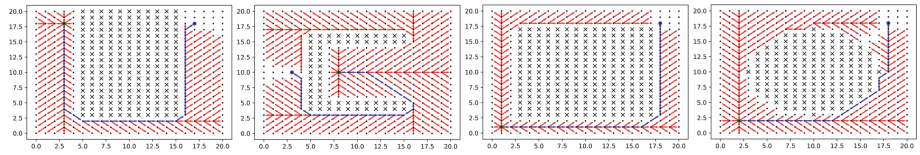
In order to generate a neural wave in the network, its origin, the initial start neuron which represents the robot cell, referred to as *start*, must be declared. This is done by raising the internal current of the neuron in question, such that $U_{start}(0) > A_{init}$. This excites its neighbors activating the neural wave, impulses forwarded by the neurons via spikes. Four stages of a neural wave expansion in an environment with four dot obstacles is visualized in Fig. 1. Here, the start neuron is marked in green and the target neuron as well as the resulting path in blue. In Fig. 1(a), the artificially altered membrane potential of the start neuron surpasses a threshold starting the wave expansion in all directions generating a vector field. The wave is terminated when the target neuron is reached, as shown in Fig. 1(d). During this process, each neuron stores internally its *parent*. Finally, the path is determined by following the stored parent-child connections from the start neuron to the target neuron.

4 Evaluation and Simulation

For all experiments presented in this section, the necessary parameters are chosen as follows:

$$\begin{aligned} A_{init} &= 0,5 & V_\theta &= 20 \\ B &= 1 & A^r &= \sqrt{2}C\left(1 - e^{-1}\right) \\ C &= 10 & A^l &= C\left(1 - e^{-1}\right) \end{aligned}$$

The thresholds A^r and A^l were chosen such that the time the wave needs to travel from neuron to neuron corresponds to the Euclidean distance of the neurons. It is hence unit less. The other constants are chosen such that the three conditions from the mathematical analysis of the proposed model hold [8]. The naming of Table 1 and 2 is uniform. *Generation time* refers to the amount of time (in seconds), needed to construct the 2D or 3D network, the neural representation of the environment. *Simulation time* is the time necessary to simulate the wave and calculate the optimal path through the network.



(a) (b) (c) (d)

Fig. 2. 2D environments with different complexity. In (a), the obstacle is formed like a rectangle and is placed on the upper border. In (b), an obstacle shaped like a horseshoe is enclosing the start position. In (c), a square- and in (d) a circle-shaped obstacle are placed in the center.

4.1 Experiments in 2D Environments

All experiments presented in this section are carried out on 2D maps, thus they are quite similar to the results of [8]. However, these tests provide a basis for the more advanced results of Sect. 4.2 and 4.3. The experiments carried out on 2D maps are all performed on a network connected by the Chamfer method. 2D Experiments regarding the Manhattan method are neglected due to triviality. In Fig. 2(d) four experiments in different 2D environments are shown. It is obvious that in each one of them the optimal path (marked in blue) was found. It is noticeable that mostly lateral connections are used in Fig. 2(a) and Fig. 2(c) as the optimal path is alongside axis. Diagonal connections are solely applied to get around the corners. A more balanced combination of lateral and diagonal connections is necessary for the optimal path in Fig. 2(b) and 2(d). Furthermore, it can be seen that the neural wave, displayed by red vectors did not reach all free neurons as the expansion of the wave is terminated immediately when the target neuron is found.

Details about the experimental results are provided in Table 1. The simulation on the map displayed in Fig. 2(c) requires the least amount of time as the majority of neurons could be discarded by the algorithm. In contrast to that, the performance shown in Fig. 2(a) is quite bad even though many neurons could be ignored here due to a huge obstacle as well. This can be explained by the fact

Table 1. All results of this table were obtained on a 2D grid connected by the Chamfer method. The respective environments are visualized in Fig. 2.

	Rectangle	Horseshoe	Square	Circle
Generation time	1.93 s	2.38 s	1.71 s	2.49 s
Simulation time	26.04 s	22.72 s	14.56 s	15.80 s
Start neuron	(3, 18)	(8, 10)	(1, 1)	(2, 2)
Target neuron	(17, 18)	(3, 10)	(18, 18)	(17,1 7)
Path length	42	27	32	26

that the length of the optimal path also influences the simulation time. These results lead to three conclusions about the general performance of our system. The generation time, the time needed to generate a neural representation of a map, increases with more complex obstacles. The simulation time on the other hand decreases if the number of neurons representing obstacles rises. This is not surprising as a large amount of obstacles decreases the neurons which need to be considered by the algorithm. Lastly, the simulation time is also related to the size of the resulting vector field.

4.2 3D Experiments with the Manhattan Method

Networks connected via the Manhattan method are visualized in Fig. 3(a), (b), (c) and (d). The results of these experiments are shown in Table 2. All grids of Fig. 3, also the once covered in Sect. 4.3, have the size $12 \times 12 \times 12$. The map of Fig. 3(a) is completely free of obstacles and serves as a basis for comparisons. The optimal path is of length 33 and changes its direction twice. In Fig. 3(a) a wall, parallel to the x-y plane, is inserted. Even though the length of the optimal path does not change from Fig. 3(a) to (b), the generation time increases and the simulation time decreases. The second observation can be explained by the fact that neurons representing obstacles are neglected by the algorithm. In (c) a second wall, parallel to the first one, is added extending the optimal path to 55 steps. The last experiment regarding the Manhattan method is carried out on a 3D map with a sphere located at the center, as shown in (d).

The data of Table 2 shows that the generation time does not always increase with complexity. If many neurons are removed, the algorithm does not need to determine all their values and neighbors. As the generation times are all within 3.7 s to 4.36 s they show only minor variations.

The worst simulation time, 0.71 s, is obtained in (a), the map without any obstacles. This is because while simulation, the wave could expand over the entire network meaning vectors needed to be calculated for every neuron. In contrast, the shortest simulation time of 0.51 s was achieved in (c). The neural wave only expanded over a subset of the network's neurons. It is remarkable that this simulation time was achieved even though the optimal path of (c) is the longest. However, the length of the optimal path does have a great influence on the simulation time as a short optimal path terminates the expansion of the wave. This effect is noticeable in (c) and the last column of Table 2.

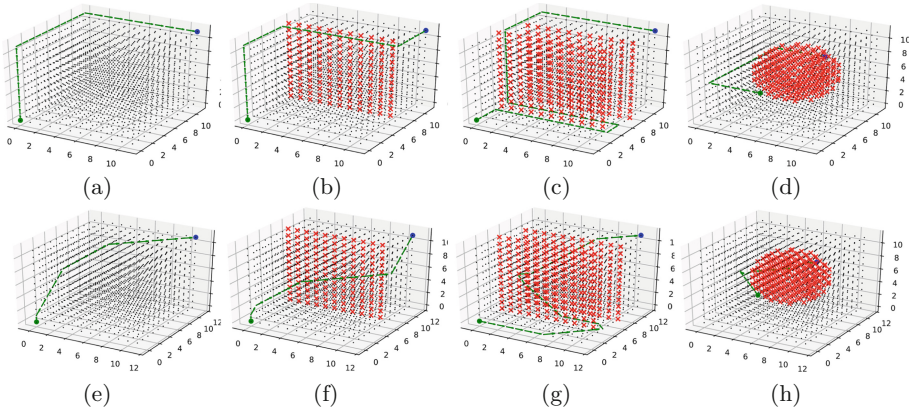


Fig. 3. The 12 × 12 × 12 network includes 1728 neurons. The networks of (a), (b), (c) and (d) apply the Manhattan, whereby (e), (f), (g) and (h) the Chamfer method. All obstacles are marked in red and the optimal path is visualized in green. The networks (a) and (e) consist only of free space while the environment of (b) and (f) embody a wall. In (c) and (g) two walls are present and the obstacle in (d) and (h) is a sphere. For all experiments, except (d) and (h), the start neuron is (0, 0, 0) and the target neuron is (11, 11, 11). (Color figure online)

Table 2. Results about experiments on a 3D grid connected by the Manhattan vs the Chamfer method. The first three rows correspond to Fig. 3(a), (b), (c) and (d) while the results of the last three rows belong to Fig. 3(e), (f), (g) and (h).

		None	Wall	2 walls	Sphere
Manhattan	Generation time	4.02 s	4.36 s	4.0 s	3.7 s
	Simulation time	0.71 s	0.62 s	0.51 s	0.53 s
	Path length	33	33	55	21
Chamfer	Generation time	5.5 s	5.36 s	5.43 s	5.12 s
	Simulation time	7.34 s	6.79 s	10.59 s	4.78 s
	Path length	16	16	33	12

4.3 3D Experiments with the Chamfer Method

The experiments displayed in Fig. 3(e), (f), (g) and (h) are connected by the Chamfer method. Hence the network including 1728 neurons is not only connected with lateral but also diagonal synapses. Information about time and path length for those tests is given in the lower half of Table 2. The most obvious difference between the results with the Manhattan method and the Chamfer method (see Table 2) is the length of the optimal path. For the map of Fig. 3(a) and (e) as well as Fig. 3(b) and (f), the optimal path is reduced by half. Also the other experiment show a significantly shorter optimal path if the network is connected via the Chamfer method. The effect on the computation time, even though it was

expected, is less desirable. Comparing the results of Table 2 shows that the generation time is increased by ca 20% while the simulation time rises massively. In some cases, the simulation time is increased by factor 10. This is simply because each neuron has up to 12 synapses more due to the Chamfer method.

In Fig. 3(e), a map with no obstacles, the approach only finds a poor solution. This observation can be explained as the synaptic connections of the Chamfer model in 3D were intentionally chosen to resemble those in 2D. For (e) the resulting path would be improved if the synaptic connections would go along the 3D diagonal. However, in cases where the optimal path is approximately along the 2D-diagonals the opposite is true. As an extension of the network to embody both diagonals for distant neurons would decelerate computations two potential solutions exist. Firstly, this issue is easily overcome with an increased number of neurons, as the approximation of the optimal path would improve. Secondly, it is possible to switch between connection types by simply changing the neighborhood definition without adversely affecting the performance.

5 Discussion

The progress with spiking neurons has resulted in a model that can be used effectively for robot path planning. MPGNN are a strong simplification of the biological basis. By eliminating superfluous influences and neuronal properties, the computing time is greatly reduced. This neuron model is effectively tailored to the problem of path planning and accordingly delivers results efficiently without being hindered by disturbing factors. If a target neuron can be reached from the start neuron, the optimal path is found without exception. Although the system does not yet achieve real time, neuromorphic hardware still promises a significant improvement in this respect. It was found that the generation time hardly depends on the type of obstacle, but on the number of neurons and, above all, the type of connection. In general, it was shown that the MPGNN-based model can be successfully used for motion planning in 3D.

Acknowledgments. The research leading to this paper received funding as the project *NeuroReact* from the Baden-Württemberg Stiftung under the research program *Neurorobotik*.

References

1. Bing, Z., Meschede, C., Röhrbein, F., Huang, K., Knoll, A.: A survey of robotics control based on learning-inspired spiking neural networks. Front. Neurorobot. **12**, 35 (2018)
2. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
3. De Momi, E., Kranendonk, L., Valenti, M., Enayati, N., Ferrigno, G.: A neural network-based approach for trajectory planning in robot-human handover tasks. Front. Robot. AI **3**(JUN), 34 (2016)

4. Ewerton, M., et al.: Learning trajectory distributions for assisted teleoperation and path planning. Front. Robot. AI **6**, 89 (2019)
5. Feldman, D.: The spike-timing dependence of plasticity. Neuron **75**(4), 556–571 (2012)
6. Furber, S., Galluppi, F., Temple, S., Plana, L.: The SpiNNaker project. Proc. IEEE **102**(5), 652–665 (2014)
7. Glasius, R., Komoda, A., Gielen, S.C.: Neural network dynamics for path planning and obstacle avoidance. Neural Netw. **8**(1), 125–133 (1995)
8. Qu, H., Yang, S., Willms, A., Yi, Z.: Real-time robot path planning based on a modified pulse-coupled neural network model. Trans. NN **20**(11), 1724–1739 (2009)
9. Janglová, D.: Neural networks in mobile robot motion. Int. J. Adv. Robot. Syst. **1**(1), 15–22 (2004)
10. Jost, J.: Temporal correlation based learning in neuron models. Theory Biosci. **125**(1), 37–53 (2006)
11. Khajeh-Alijani, A., Urbanczik, R., Senn, W.: Scale-free navigational planning by neuronal traveling waves. PLoS ONE **10**(7), e0127269 (2015)
12. Koenig, S., Likhachev, M.: Incremental A*. NIPS2001, pp. 1539–1546 (2002)
13. Merolla, P.A., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)
14. O'Keefe, J., Dostrovsky, J.: The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. Brain Res. **34**(1), 171–175 (1971)
15. Pal, A., Tiwari, R., Shukla, A.: A focused wave front algorithm for mobile robot path planning. In: Corchado, E., Kurzyński, M., Woźniak, M. (eds.) HAIS 2011. LNCS (LNAI), vol. 6678, pp. 190–197. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21219-2_25
16. Polyakova, M., Rubin, G., Danilova, Y.: Method for matching customer and manufacturer positions for metal product parameters standardization. In: AIP, vol. 1946 (2018)
17. Ponulak, F., Hopfield, J.: Rapid, parallel path planning by propagating wavefronts of spiking neural activity. Front. Comp. Neurosci. **7**, 98 (2013)
18. Qureshi, A., Simeonov, A., Bency, M., Yip, M.: Motion planning networks (2019)
19. Raković, M., Savić, S., Santos-Victor, J., Nikolić, M., Borovac, B.: Human-inspired online path planning and biped walking realization in unknown environment. Front. Neurorobot. **13**, 36 (2019)
20. Subashini, M., KumarSahoo, S.: Pulse coupled neural networks and its applications. Expert Syst. Appl. **41**(8), 3965–3974 (2014)
21. Tolman, E.C.: Cognitive maps in rats and men. Psychol. Rev. **55**(4), 189 (1948)
22. Weber, C., Triesch, J.: From exploration to planning. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008. LNCS, vol. 5163, pp. 740–749. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87536-9_76
23. Zeller, M., Sharma, R., Schulten, K.: Motion planning of a pneumatic robot using a neural network. IEEE Control Syst. **17**(3), 89–98 (1997)
24. Zennir, M., Benmohammed, M., Boudjadja, R.: Spike-time dependant plasticity in a spiking neural network for robot path planning. In: AIAI, vol. 1539 (2015)