Surface Light Barriers

Theo Gabloffsky[a], Britta Kruse and Ralf Salomon
University of Rostock, 18051, Germany

Keywords: Light Barrier, Neural Network, Backpropagation Network, Radial Basis Function, Speed Measuring, Curling.

Abstract: It should be known to almost all readers that light barriers are commonly used for measuring the speed of various objects. These devices are easy to use, quite robust, and of low cost. Despite their advantages, light barriers exhibit certain limitations that occur when the objects of interest move in more than one spatial dimension. This paper discusses a physical setup in which light barriers can also be used in case of two-dimensional trajectories. However, this setup requires rather complicated calculations. Therefore, this paper performs these calculations by means of different neural network models. The results show that backpropagation networks as well as radial basis functions are able to achieve a residual error less than 0.21 %, which is more than sufficient for most sports and everyday applications.

1 INTRODUCTION

In modern sporting events, technology is present in numerous applications, for example in scoring boards, video proof, or just for measuring distance and time in competitions. In addition, regular practice is interested in recording and analyzing physical data, in order to enhance performance. These physical data can be manifold, for example time, weight, speed, and distance.

However, the recording of the speeds of objects and athletes can be considered of special interest in various sports, such as bobsleigh, tennis, and running. From a physics point of view, an object's speed $v = \Delta s/\Delta t$ is defined as the distance Δs it travels during the time interval Δt. Commonly used speed measuring technologies include RFID transponders, wearable GPS devices or a video-based tracking. Another tool for measuring the speeds of objects are light barriers which are comparatively simple, sufficiently precise, and easy to install. As shown in Fig. 1, a light barrier basically consists of three components, a light source, a sensor, and an evaluation unit. The first two elements are aligned in a way that the light source projects onto the sensor. If a moving object travels through the light barrier, the beam of light is interrupted, which is recognized by the evaluation unit, which in turn generates a timestamp t_1. With two light barriers in place, the speed v reads: $v = (x_2 - x_1)/(t_2 - t_1) = \Delta x/\Delta t$.

[a] https://orcid.org/0000-0001-6460-5832

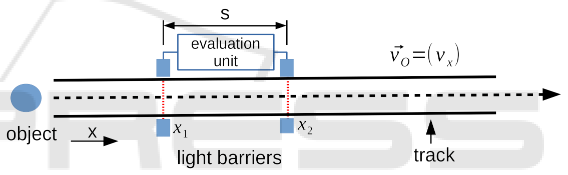

Figure 1: Usually two light barriers are used to measure the speed of an object that is moving in a narrow path. On crossing a beam of light, the respective light barrier records a timestamp t_i. With two timestamps t_1 and t_2 and the traveled distance s, the resulting speed is $v = (t_2 - t_1)/s$. The computations are done by the evaluation unit.

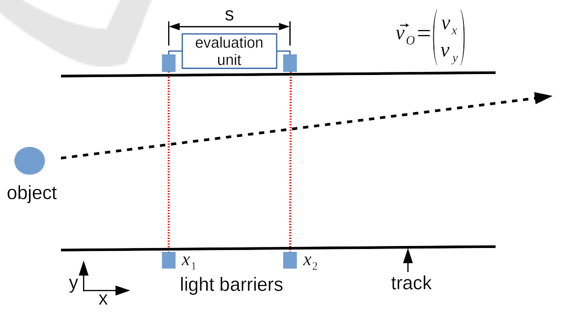

Figure 2: The physical setup shows the possible path of the object through a wide path. Here, an object can move in the x-direction and also in the y-direction. Thus, the velocity $\vec{v} = (v_x, v_y)^T$ is a two dimensional vector.

The examples mentioned above are simple by their very nature: the moving direction is always along the physical environment, and thus perpendicular to the orientation of the light barriers. Therefore, both speed v and velocity are identical quantities

$v = |\vec{v}|$.

The situation is not always as simple as described above. For example in the sports of shot put or curling, the objects might also have a speed component v_y that is perpendicular to the main orientation of the physical setup as illustrated in Fig 2. This component might assume significant values, which invalidate the speed measurements as given by the light barriers, at least to some degree. In curling, for example, the scientific support of the training process of the german national teams require the precise measurement of both values v_x *and* v_y.

This paper presents a new light barrier system called *Surface Light Barriers* (SLB), which offers a possible solution to this problem. As Section 2 shows, the system consists of four light barriers, which are set up in different angles with respect to the x-direction of the track. When an object travels through the light barriers, the system generates *four* timestamps from which it calculates the object's two-dimensional velocity $\vec{v} = (v_x, v_y)^T$. In addition, it calculates the objects starting position $p = (p_x, p_y)^T$ and end positions $q = (q_x, q_y)^T$. Since the required calculations are rather cumbersome, this paper explores various neural network models. To this end, Section 3 briefly reviews two different network models, i.e. backpropagation networks and radial basis functions.

For the evaluation, the neural networks were trained with an exemplary dataset generated on the basis of the sport of curling. The specific configurations are summarized in Section 4.

The results, as shown in Section 5 and discussed in Section 6, indicate that in terms of the average error, the best neural network is a radial basis function network, which achieves an average error of 0.21 %. In terms of the speed to error ratio, the best network is a backpropagation network that employs two hidden layers and achieves an error of 0.901 %.

Finally, Section 6 concludes this paper with a brief discussion.

2 SURFACE LIGHT BARRIERS: PHYSICAL SETUP

As already discussed in the introduction, light barriers face limitations in observing two-dimensional velocities when they are used in their usual setup. The *Surface Light Barriers* system, or SLB for short, solves this problem with a physical setup, as presented in Fig. 3. This setup consists of the following hardware configuration:

1. Two light barriers L1 and L4 are positioned at x_1 and x_4 and are aligned orthogonal to the main direction of movement. The distance between these light barriers is called s_2.
2. Two light barriers L2 and L3 are positioned at x_2 and x_3, with an alignment angle of $\gamma \neq 90°$. The distance between these light barriers equals the distance $s_1 = s_2/2$.
3. An evaluation unit, which includes a microcontroller with a time tracking mechanism and a neural network.

In essence, the evaluation unit employs a microcontroller to which every light barrier is connected via an input port. When an object crosses a light barrier, the interruption of the beam of light sends a signal to the corresponding port of the microcontroller. The microcontroller detects signal changes by either periodical polling of the input pin or the execution of hardware interrupts.

In case of a signal change, the evaluation unit generates a timestamp. After the object has crossed all four light barriers, the evaluation unit has generated a total number of four individual timestamps $t_{1..4}$. Without loss of generality, these timestamps can also be presented as differential quantities:

$$\begin{aligned} \Delta t_1 &= t_1 - t_1 \\ \Delta t_2 &= t_2 - t_1 \\ \Delta t_3 &= t_3 - t_1 \\ \Delta t_4 &= t_4 - t_1 \end{aligned} \tag{1}$$

Equation (1) provides a transformation in time such that an object enters the system at the *virtual* time $t = 0$. It might be helpful to mention, that due to the physical setup,
$t_1 < t_2, t_3, t_4$ always holds, and that thus all differential quantities Δt_i are always positive.

3 SURFACE LIGHT BARRIERS: NEURAL NETWORKS

The purpose of the evaluation unit is to deliver the velocity $\vec{v}$, i.e., the speeds v_x and v_y, as well as the starting and end points p_y and q_y of the object. However, the mathematical approach is rather cumbersome, and is, therefore, presented in short in the appendix. As an alternative, this paper employs various neural networks for this task. It explores the utility of backpropagation networks as well as radial basis function networks.

Training Data: Training data are elementary for training neural networks. Verification data are also

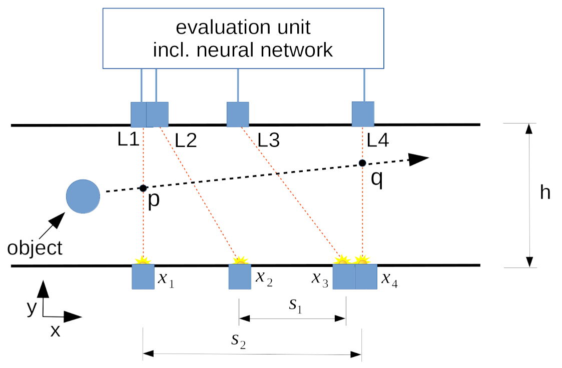

Figure 3: Example setup for the SLB-system, which basically consists of four regular light barriers and an evaluation unit. Two of the light are arranged as usual, whereas the other two light barriers are inclined in regard to the other light barriers.

required for testing the generalist capabilities of the final networks. Ideally, both datasets differ from one another. In addition, the datasets should be drawn such that they cover the relevant input range of the target application.

Backpropagation Network: A backpropagation network consists of an input layer, a choosable count of hidden layers, and one output layer. Each layer can consist of a different number of neurons. Every neuron of a layer has weighted connections to all neurons of the previous layer.

The output of neuron i is calculated in two steps: (1) It calculates it net-input $\text{net}_i = \sum_{j=1}^{n} o_j w_{ij}$ calculates through the summation of the outputs o_j of the neurons j of the previous layer. (2) It determines its output $o_i = f(\text{net}_i)$, with $f(\text{net}_i)$ denoting the logistic function $f(\text{net}_i) = 1/(1+e^{-\text{net}_i})$. These calculations are done for every neuron in all hidden layers and output layers.

The connections are trained with the well-known backpropagation algorithm:

$$\vec{w}^{t+1} = \vec{w}^{t} - \eta \nabla(\vec{w}^{t}) \tag{2}$$

For further details on backpropagation, the interested reader is referred to the pertinent literature (Wythoff, 1993).

Radial Basis Function: Radial basis functions (RBFs) differ from backpropagation networks quite a bit. In addition to the input layer, an RBF network consists of only one layer of neurons. All neurons in this layer, also referred to as *kernels*, maintain a weight vector $\vec{w}$. The vector connects each neuron to all input neurons. In contrast to calculating the weighted input sum, as it is done in backpropagation networks, every neuron calculates its *distance* net_i to the current input value:

$$\text{net}_i = \sum_j (w_{ij} - I_j)^2, \tag{3}$$

Its activation θ is then set to:

$$\theta_i = e^{\frac{\text{net}_i}{\beta}}, \tag{4}$$

with β denoting a kernel width. In addition to the weight vector $\vec{w}_i$, every RBF neuron also maintains an output vector $\vec{O}_i$. For the sake of simplicity, the following mathematical description assumes that the network has only *one* single, scalar output value. Consequently, all RBF neurons can collapse their associated output vectors into a single output scalar O_i. The final *network* output o is then determined by applying the softmax function:

$$o = \frac{\sum_i O_i \theta_i}{\sum_i \theta_i} \tag{5}$$

In other words, the softmax function calculates a weighted output, which considers the target output value O_i of every RBF neuron as well as its current activation θ_i.

In many applications, all the neurons' output values O_i and their weight vectors $\vec{w}_i$ are subject to the learning process. Without loss of generality, the SLB system assumes that all input values are possible within a given range. Thus, only the target output value O_i is subject to the learning process. The weight vectors $\vec{w}_i$ are equally distributed over the *three* input domains $\Delta t_{2..4}$ The SLB system applies the following learning rule:

$$O_i^{t+1} = O_i^t + (h - o_i)\eta\theta_j \tag{6}$$

Every output value o_i is adapted towards the known target value h by an amount that is proportional to the remaining difference $E = (h - o_i)$ as well as the current activation of the RBF neuron θ_i. For further details the interested reader is referred to the pertinent literature (Tao, 1993).

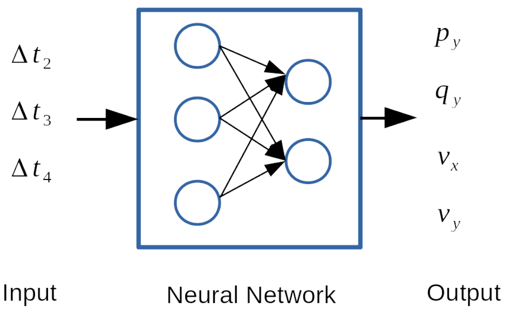

Figure 4: Overview of the input ($\Delta t_{2..4}$) and output (p_y, q_y, v_x, v_y) parameters of the neuronal network. All values are normalized in respect to their maximum value.

4 METHODS

This section describes the datasets as well as the neural networks in full detail as far as required for the reproduction of the results presented in Section 5.

Application Area: The SLB system has been developed as part of a project in the sport of curling. Therefore, all key values with respect to physical quantities, such as dimensions and speeds, have been drawn from this project.

Simulation Hardware and Software: All measurements were done on a system containing an Intel i7 4710HQ with a clock speed of up to 3.5 GHZ and 8 GB of DDR3 RAM. The software used for simulating the networks was written in the C programming language and no external tools were used. The simulations were performed on a single core of the mentioned processor and no parallel calculation was used.

Dataset: The dataset was created with a mathematical model of the SLB-System including an object which passes through the light barriers, as presented in Fig. 3. The setup of the SLB consists of the following dimensions:

$$\begin{aligned} s_1 &= 500\,[mm] \\ s_2 &= 1\,000\,[mm] \\ h &= 5\,000\,[mm] \end{aligned} \tag{7}$$

The trajectories of the object were generated randomly bound to the following parameters:

$$\begin{aligned} 1\,000\ [mm/s] &\leq v_x \leq 8\,000\ [mm/s] \\ -500\ [mm/s] &\leq v_y \leq 500\ [mm/s] \\ 0\ [mm] &\leq p_x \leq 5\,000\ [mm] \\ 0\ [mm] &\leq q_x \leq 5\,000\ [mm] \end{aligned} \tag{8}$$

Out of these trajectories the timestamps $t_{1..4}$ and differential timings $\Delta t_{2..4}$ were calculated. To fit into the value range of the neural networks, the differential times were normalized $t_{2..4_n} = \Delta t_{2..4}/t_{\max}$ by dividing each time with the longest possible occurring time $t_{\max}$. This time $t_{\max} = s_2/v_{x_{\min}} = 1\,000\,[mm]/1\,000\,[mm/s] = 1\,[s]$ is derived from the distance s_2 between the light barriers L1 and L4 as seen in Fig. 3 and the minimum speed of the object $v_{\min} = 1\,000\,[mm/s]$.

The following equation describes the normalization of the output parameters, where the maximum values depend on the generated trajectories with respective $v_{x_{\max}} = 8\,000\,[mm/s]$, $p_{y_{\max}} = q_{y_{\max}} = 5\,000\,[mm]$:

$$\begin{aligned} v_{x_n} &= v_x/v_{x_{\max}} \\ v_{y_n} &= v_y/y_{y_{\max}} \\ p_{y_n} &= p_y/p_{y_{\max}} \\ q_{y_n} &= q_y/q_{y_{\max}} \end{aligned} \tag{9}$$

In total, 10 000 trajectories were created as a training set and another 1 000 as a verification set. In summary, the datasets consist of the following normalized parameters: $\Delta t_{2_n}, \Delta t_{3_n}, \Delta t_{4_n}, v_{x_n}, v_{y_n}, p_{y_n}, q_{y_n}$

Naming Convention: In the following, the names of the networks are denoted as follows: The prefix before the network name denotes the number of neurons in the hidden layers.

As an example, the name *64-32-BP* refers to a backpropagation network with two hidden layers, with 64 neurons in the first hidden layer h_1 and 32 neurons in the second hidden layer h_2. The name *14-RBF* refers to a radio basis function network with 14 neurons per axis, which resolves into a total of $14^3 = 2744$ neurons in its neuron layer.

Backpropagation Network: The generated datasets have been employed with different backpropagation architectures, varied between one to three hidden layers with four to 64 neurons in each layer. The initial values of the weights w_{ij} between the neurons were set randomly between $-0.1 < w_{ij} < 0.1$. The networks were trained with a fixed learning rate of $\eta = 0.05$ and a momentum of $\alpha = 0.9$. Further information about the general functionality of the momentum can be found in (Phansalkar and Sastry, 1994).

All backpropagation networks utilize the quadratic error as the loss function. The error of a neuron j is calculated by $E_j = 1/2\sum_j(h_j - o_j)^2$ where h_j is the target value of that neuron.

As the activation function, all backpropagation networks use the logistic function: $f(net) = 1/(1 + e^{-net})$.

Radial Basis Function Network: The three input-parameters are interpreted as the *dimensions* of the network. Each dimension ranges from 0 to 1. Similar to the backpropagation networks, multiple radial basis function networks were trained. The networks differ in the number c_{dim} of neurons per dimension, ranging from four to 20 neurons per dimension. This results in a total of up to $c_{total} = c_{dim}^3 = 20^3 = 8\,000$ neurons in the largest network. The distance between the neurons is set to $\beta = 0.1\,c_{dim}$. The learning rate of the RBF was set to $\eta = 0.5$. The error of an output j in the radial basis function network $E_j = (h_j - o_j)$ is equal to the absolute of the difference between the target value h_j of the output and the softmax of the activation of the network o_j.

Total Error of the Networks: The total error of a network describes the average error of the output parameters of that network:

$$E_{total} = \frac{1}{4}\left(E_{p_y} + E_{q_y} + E_{v_x} + E_{v_y}\right) \qquad (10)$$

Measurement of Computational Costs: Though the target computer-architecture of the neural networks will be a microcontroller, the computational costs of the networks are an important factor. Therefore, the processing time of the networks were measured with the help of the C-standard library. All timing measurements included the calculation of 1000 iterations of an entry of the dataset.

Error to Cost Ratio ζ: To evaluate the performance of the networks, the ζ-factor is introduced. The factor is calculated by $\zeta = t_{calculation} E_{total}$ and describes the ratio between the calculation time and the resulting total error of the network. The lower the ζ-factor, the better the error-to-time ratio of that network is.

5 RESULTS

Backpropagation Network: Figure 5 presents the average errors which occur during the training process in the different backpropagation networks. As an example, the network with four neurons shows a steady average error of approx 12.5 %. from epoch 1 to approximately 2 500. Between epoch 2 500 and 2 700 a sudden decrease of the error occurs and after epoch 4 000, the error reaches a plateau. The other networks have a similar behaviour, where the difference lies in the required epochs to reach the plateau and the resulting error.

In all neural networks, fluctuations of the errors are visible. This can been seen when looking at the graph of the largest network 64-32-16-BP with three hidden layers. Between epoch 5 800 and 6 200, the error is fluctuating around an error of approximately 3 %, before it decreases to a plateau of 0.72 %. The mentioned network also requires remarkably more epochs for the training than the other networks.

As presented in Fig. 6, the network shows a sudden peak of error of the parameters p_y,q_y and v_x between epoch 550 and 600. Though the three parameters still decrease after the peak, the parameter q_y remains at a higher error than before. This occurs due to an overtraining of the parameter.

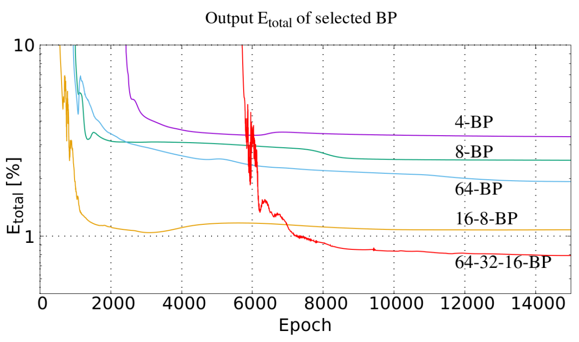

Figure 5: The summary of the resulting average error of the parameters v_x, v_y, p_y and q_y of different networks.

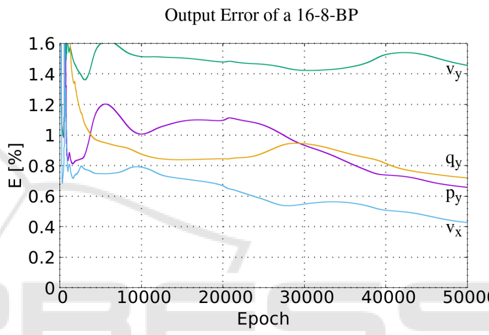

Figure 6: The resulting error of a backpropagation network with two hidden layers with $h_1 = 16$, $h_2 = 8$ neurons during the learning process.

Radial Basis Function Network: Figure 7 presents the average errors which occur during the training process in the radial basis function networks. Figure 8 shows the error of the different output parameters of the RBF in the learning process. The graph shows no fluctuations as well as no error plateaus nor signs of overtraining. In contrast to the BPs, the RBF took more epochs to learn the output parameters with a constant decrease of the error.

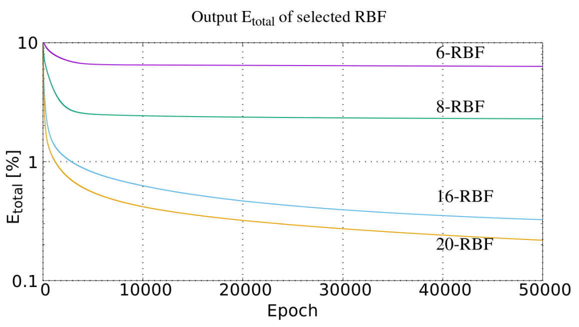

Figure 7: The summary of the resulting average error of the parameters v_x, v_y, p_y and q_y.

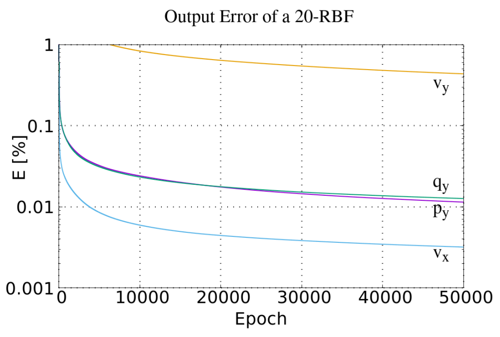

Figure 8: The resulting error of the radial basis function with 20 neurons over the learning process.

Comparison of Errors: Both types of the discussed network architectures, were able to learn and calculate the data as seen in Figs. 5 and 8. Remarkable is that all networks required significantly more time to learn the v_y parameter compared to the other three parameters. The reason for this is yet unclear, as the parameters p_y and q_y depend on the same timing inputs as v_y.

Table 1 summarizes the final resulting errors of the networks after 50000 epochs. The results show that the best error was achieved with the radial basis function network with a total error of 0.218 %. The best backpropagation network is the one containing three hidden layers with $h_1 = 64$, $h_2 = 32$ and $h_3 = 16$ neurons resulting in an error of 0.72 %. The backpropagation network with fewer layers and neurons show a higher error rate. The maximum error has the backpropagation network with four neurons of approx 3.2 %.

Comparison of Calculation Time: As presented in Table 2, the examined network architectures show a significant difference in the calculation time. Figure 9 presents the time that the networks require to calculate 1000 entries of the dataset. The amount of neurons in the 4-BP and 4-RBF are the same and also the calculation time is roughly the same. This is not the case for the 8-BP and 8-RBF networks. Both have the same count of neurons, but the difference in calculation time is significant: $t_{8-BP} = 0.7[ms]$ to $t_{8-RBF} = 12.8[ms]$. The calculation time of the RBF rises exponentially with the count of neurons.

Error to Speed Ratio ζ: In addition to the calculation times, Table 2 also presents the ζ-factor.. Further, Fig. 10 gives an overview over the ζ-factors for the different networks. The factor is not the same for every network. The lower the value, the better the calculation time to error ratio is. In general, the backpropagation networks have a significant lower ζ-factor than the radial basis function networks. The RBF show two sweet spots of the performance that peak at the neuron counts of four and 12 neurons. By contrast, the backpropagation networks show no real sweet spots. The 16-8-BP shows the best performance of the BP networks.

Table 1: Summary of the percentual error of the BP and RBF with different configurations after 50,000 epochs.

network	p_{yStart}	p_{yEnd}	v_x	v_y	E_{total}
4-BP	2.83	3.06	3.20	3.93	3.28
8-BP	2.09	2.52	1.74	3.23	2.46
32-BP	1.97	2.54	1.39	4.17	2.72
64-BP	1.41	1.96	0.45	1.08	1.34
16-8-BP	0.65	1.45	0.42	0.71	0.90
32-16-8-BP	0.32	1.51	0.23	0.33	0.79
64-32-16-BP	0.17	1.52	0.15	0.24	0.78
4-RBF	3.08	3.48	2.74	20.8	10.7
6-RBF	0.59	0.79	0.43	12.5	6.32
8-RBF	0.15	0.20	0.08	4.57	2.29
10-RBF	0.05	0.07	0.02	1.74	0.87
12-RBF	0.02	0.03	0.004	0.94	0.47
14-RBF	0.02	0.02	0.005	0.77	0.38
16-RBF	0.01	0.02	0.004	0.71	0.36
18-RBF	0.01	0.01	0.004	0.60	0.28
20-RBF	0.01	0.01	0.0003	0.43	0.21

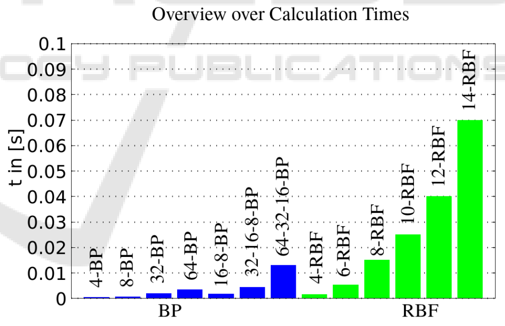

Figure 9: Speed comparison for different neural networks for calculating 1000 entries of the dataset.

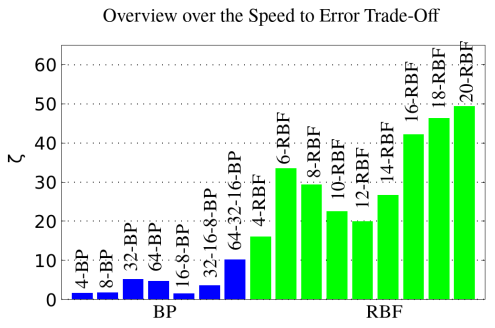

Figure 10: Comparison of the speed-to-error ratio ζ of the networks. The lower the value the better the ratio.

Table 2: The table presents the required calculation times of the networks to calculate 1000 entries of the dataset. In addition, the total error and an error to calculation-time are presented.

network	t in [ms]	E_{total}	$E_{total} * t$
4-BP	0.5	3.28	1.6
8-BP	0.7	2.46	1.7
32-BP	1.9	2.72	5.1
64-BP	3.5	1.34	4.7
16-8-BP	1.7	0.90	1.5
32-16-8-BP	4.4	0.79	3.5
64-32-16-BP	13.1	0.78	10.2
4-RBF	1.5	10.72	16.0
6-RBF	5.3	6.32	33,5
8-RBF	12.8	2.29	29.3
10-RBF	25.7	0.87	22.4
12-RBF	42.1	0.47	19.9
14-RBF	70.1	0.38	26.6
16-RBF	117.4	0.36	42.1
18-RBF	165.3	0.28	46.2
20-RBF	226.4	0.21	49.3

6 DISCUSSION

This paper has presented a system that measures the two-dimensional velocity of an object on the basis of timestamps generated by light barriers. In comparison to the conventional approach, these light barriers are set up not only in an orthogonal angle with respect to the direction of movement, but also inclined to it. When an object moves through these light barriers, the system generates four timestamps which are passed on to a neural network. The network retraces both the path of the object and its two-dimensional velocity.

In addition, multiple neural network architectures were explored in order to investigate the effects of the number of layers and neurons on the learning time and formal precision as well as the execution time of the networks. Therefore, the ζ-factor was introduced to measure the speed-to-error ratio. The created datasets were used to train and test these different architectures.

Table 1 shows that the overall best performance was achieved by the radial basis function network with an error of $0.218\,\%$. The lowest error of the backpropagation networks was achieved by network which consists of three hidden layers with the following neuron count: $h_1 = 64$, $h_2 = 32$, and $h_3 = 16$. Although the network had a lower average error than the others, the result of a network with four neurons is sufficient for an application such as curling.

On the one hand, the difference between the average errors of backpropagation network 4-BP and 64-32-16-BP with $3.28\,\%$ versus $0.78\,\%$ have a low impact on the resulting error of the system. On the other hand, the reduced amount of neurons have a high impact on the required computational resources of the evaluation unit, e.g., the memory footprint of the program and the calculation time for re-training. This is shown by the ζ-factor: a larger network does not decrease the error in the same amount as the calculation time rises. However, the network with four neurons could be used on a smaller microcontroller, e.g., ESP8266, whereas a larger network requires a larger and more expensive hardware solution. The best speed-to-error trade-off is achieved by a 16-8-BP network. The same line of argumentation holds to the radial basis function networks, where the best trade-off is achieved with an 12-RBF. Therefore, for the present application, the best network choice is a backpropagation network with merely one hidden layer with four neurons.

The results presented in this paper are generated by simulations and without a practical validation. In comparison to a physical application, the simulation contained an abstraction of the objects in regard to size and dimension. The simulated objects were comparatively smaller than curling stones which have a diameter of $28.8\,cm$. Without re-training of the network with real trajectories, the output of the network is prone to error. In addition, the simulated trajectories of the objects have a constant velocity while moving through the light barriers. In a practical scenario, friction effects would occur. However, these might be neglectable due to the light barriers dimensions, at least in the sport of curling.

As described in Section 4, the networks were trained with a fixed learning rate, which results in fluctuations of the errors as has been further addressed in Section 5. Future work will be devoted to the implementation of an adaptive learning rate to speed up the training process, to reduce the fluctuation as well as error plateaus and to reduce the resulting error. Furthermore, future research will consider the extension of the simulation with more light barriers to calculate accelerations. The final step will consists in building such a system in a real-world application.

7 CONCLUSION

In summary, this paper closes with the following conclusions: The system of *Surface Light Barriers* is able to calculate the two-dimensional speed of an object through a smart positioning of multiple light barriers.

Both, backpropagation neural networks and radial basis function networks are able to learn the model of the *Surface Light Barriers*. A neural network with four neurons is able to learn the model with an acceptable error rate of $E < 4\%$, which is sufficient for sport application. These small neural networks can be used on microcontrollers with low-power hardware. For applications which require a higher accuracy, more complex neural networks with multiple layers are able to achieve an error of $E < 0.3\%$.

ACKNOWLEDGEMENT

The authors gratefully thank the German Curling Association (Deutscher Curling Verband) for providing this engineering task.

REFERENCES

Phansalkar, V. and Sastry, P. (1994). Analysis of the backpropagation algorithm with momentum. *IEEE Transactions on Neural Networks*, 5(3):505–506.

Tao, K. (1993). A closer look at the radial basis function (rbf) networks. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 401–405 vol.1.

Wythoff, B. J. (1993). Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18(2):115–155.

A VECTOR CALCULATIONS

In the following, each light barrier is seen as a vector consisting of a starting point $P_i = (p_{ix}, p_{iy})^T$ and a direction of the beam of light. $\vec{B}_i = (b_{ix}, b_{iy})^T$.

$$LB_i = P_i + s_i\vec{B}_i \tag{11}$$

The object O moves through the light barriers with a starting point and a constant speed vector $\vec{V} = (v_x, v_y)^T$:

$$O = P_O + t\vec{V}_O \tag{12}$$

The entrance point of the object in the section of measurement is called $P_{start} = (0, y_{start})^T$. The point of leaving is called $P_{end} = (p_{4x}, y_{end})^T$. To the differential timings, as stated in Eq. 1, can be added the following differential timing:

$$\Delta t_{23} = t_3 - t_2 \tag{13}$$

The final formulars for the calculations are:

$$v_x = \frac{p_{x4} - p_{x1}}{\Delta t_4} \tag{14}$$

$$v_y = \frac{(\Delta t_{23} v_x - p_{3x}) b_{3y}}{b_{3x} \Delta t_{23}} \tag{15}$$

$$y_{start} = p_{2y} + \left(\Delta t_4 v_x - \frac{p_{2x}}{b_{2x}}\right) b_{2y} - \Delta t_4 v_y \tag{16}$$

$$y_{end} = y_{start} + v_y \Delta t_4 \tag{17}$$