A New Approach to Decision Making in Basketball - BBFBR Program

Branko Markoski[1], Predrag Pecev[2], Laszlo Ratgeber[3], Miodrag Ivković[4], Zdravko Ivanković[5]

[1]University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia, markoni@uns.ac.rs

[2]University of Novi Sad, Faculty of Sciences / Department of Mathematics and Informatics, Novi Sad, Serbia, predrag.pecev@gmail.com

[3]University of Pécs, Faculty of Health Sciences, Pécs, Hungary, ratgeber.laszlo@gmail.com

[4]University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia, misa.ivkovic@gmail.com

[5]University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia, misa.ivkovic@gmail.com

Abstract: The developed solution is named BBFBR, which stands for Basketball Board for Basketball Referees. The current implementation of the solution is based on drawing the ball's movement on the court, which is the input vector of the neural network, whilst the output vector of the neural network consists of the movement coordinates of the referees. The key segment of this paper describes, in detail, the structure of the input and output vectors of the neural network used in the BBFBR project, as well as the methods, and the advantages and flaws that where noted during the training of the neural network. There are two methods used while training the neural network: the method of sequential repetition. The current solution enables calculating optimal ways of the referee's movement in the case that the movement of the ball in an action consists of not more than 15 key points, e.g. guiding the ball from one point to another, passing the ball to another player, shooting etc. The application value of the current solution bears only educational purpose because it is possible to apply it to training young basketball referees in the terms of their movement, to enable them to be aware of an action and to be able to analyze it. This paper also describes the methods and the developed software which, based on the action and the movement of the ball, using the neural network, determines the movement of a basketball referee on the court, in order to gain the best view of the action. The solution is developed in Microsoft Visual Studio 2010, written in the programming language C# referring to AForge .NET Framework for the support in the aspects of configuring, training and usage of neural networks. AForge .NET Framework is published with LGPL v3 licence.

Keywords: neural networks; basketball; path prediction

1 Introduction

Most solutions that are based on the applications of neural networks in basketball use neural networks to predict the outcome of the game, to analyse the score, to draft score predicting, etc. [2] [3] One of the well-known solutions is called Basketball Predictor and it predicts outcomes of games using neural networks, based on the data on the teams that play, the previous games, teams' selection, etc. The solution covers the American NBA and the WNBA league, and the European leagues: Austria, Czech Republic, France, Germany, Greece, Italy, Poland, Russia, Spain and Sweden.

In the paper [1], it is determined that the most common elements of a basketball game are shots for 2 points under the hoop and the defensive rebound, by analysis of the first B basketball league for men from 2005 and 2010, using a feed forward neural network.

The paper [4] describes how using various neural networks predicts the outcome of a game, emphasizing that the predictions of the trained neural networks were more precise comparing to the basketball experts' predictions. Practically, trained neural networks predicted the outcome of the game correctly in 74.33% cases, while the basketball experts were precise in 68.7% cases. In the paper, the use is presented of feed forward, radial, probabilistic, regressive neural networks, and also the fusion of them.

Based on what has been here mentioned, it can be concluded that the application of neural networks in basketball is a fairly popular field and that the leading trend is predicting the outcome of a game. This goal is also the most profitable considering the gambling industry and the teams' desire to be as high ranked as possible. Using the methods mentioned, it is possible to determine whether a team is able to win another one by simulations based on neural networks. It also enables to turn the game in progress around or find a way to improve the score if losing the game is inevitable.

Various factors influence the outcome of a game: the players' performance, tactics, the time zone, whether the game is played at home or not, etc. However, the impact of the referees who decide on the regularity of the points, and in the end, the final score, is not emphasized enough. It is common that teams lose games because of the referee's wrong decision about a single point. Therefore, the question arises: What can be done to make the basketball referee's work easier?

Watching the basketball game is often disturbed by the crowdedness on the court during an action, so practically nothing can be observed on the referee's part. This can result in referee making a bad decision.

The idea of the BBFBR solution is in fact that some actions can be observed better from certain positions. Therefore, using neural networks it is possible to calculate the ideal path movements of the referees who observe a certain action, regarding the data on basketball actions. The reason to use neural networks is the fact that basketball actions are prone to variations and improvisations, so, very often, new actions are created. The usage of neural networks in these cases is desired because then, based on experience of the neural network and the similarities with other basketball actions, it is easier to find the optimal path for the referee's movement with prediction in real time. Capabilities and needs mentioned are almost impossible or barely possible to realize via conventional solutions, by writing algorithms.

The use of the solution mentioned has potential wide area of application:

- Great application value in training young basketball referees in the terms of how to move and where to look during the certain actions, in order to observe and make the best decision about the validity of the point, in case of the fault or in other situations.
- Retroactive analysis of the referee's decision, when there is a suspicion that the decision was wrong. For example, could a certain basketball referee have adequately observed the action when it was decided not to take the point into consideration, although it is suspected that there was a fault or any other technical inconsistency previously made? Adding the element of prediction enables one to gain the prediction of the neural network about whether the referee is in the optimal position to observe a certain action in real time, during the action itself. Further development of this application could result in developing a system similar to the HawkEye system, which already exists and is applied in tennis. Applying the idea mentioned, it is possible to minimize the coefficient of human mistake while determining the validity of a point to a certain extent, thereby minimizing unfair game, cheating and favouring certain teams, and it will be very easy to find corrupted referees.
- If it is possible to determine the ideal movement of the referee in a certain action, by implementing predicting neural networks, it is possible to convert the movement of the referee into movement of the cameras, which follow the basketball action in real time. This, in addition to the static cameras that are pointed at players by cameramen during the game, could enable the viewers to see the referee's view, or the first row view. A similar principle is seen in Formula 1, when viewers can see the view that their favourite driver sees from the vehicle.

2 The BBFBR Program - Structure and Class Diagrams

The program BBFBR consists of four parts: BBFBR and NN Trainer developed in one project, nnUtility developed as a separate dll file and AForge .NET framework, which offers support for training realization of the neural network. The structure of the BBFBR program is presented in Figure 1.

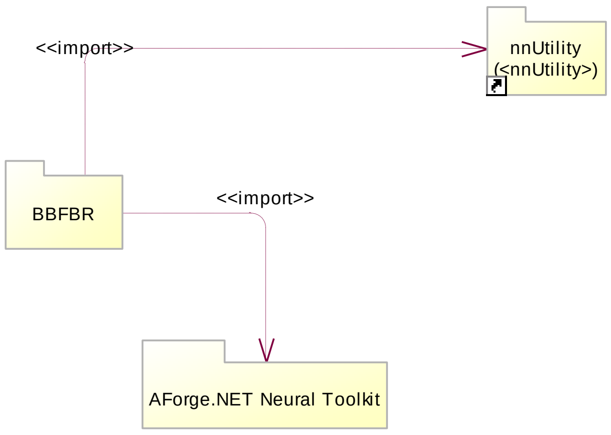


Figure 1
Structure of the BBFBR program

nnUtility package, which offers program's logistics in BBFBR solution and consists of six packages, is presented in Figure 2. Those six packages are:

- BallData – Package contains the structure and logic that is required while describing the movement of the ball in the basketball court.
- RefereeData – Package contains the structure and logic that is required to describe the movement of the referees in the basketball court.
- Calculations – Package contains static methods that are used for detailed calculations of the ball movement during rendering.
- ImageRotation – Package contains useful methods for the rotation of the pictures in the memory.
- nnTrainer – Package contains supporting structures that are used while training the neural network for positioning the basketball referees.
- DoubleBufferPB – Package contains clsDoubleBufferPB class, whose instance is the Display element which is used for displaying the court, drawing actions and animating and rendering solutions. The functionality of the Display element will be described in detail later in the text.

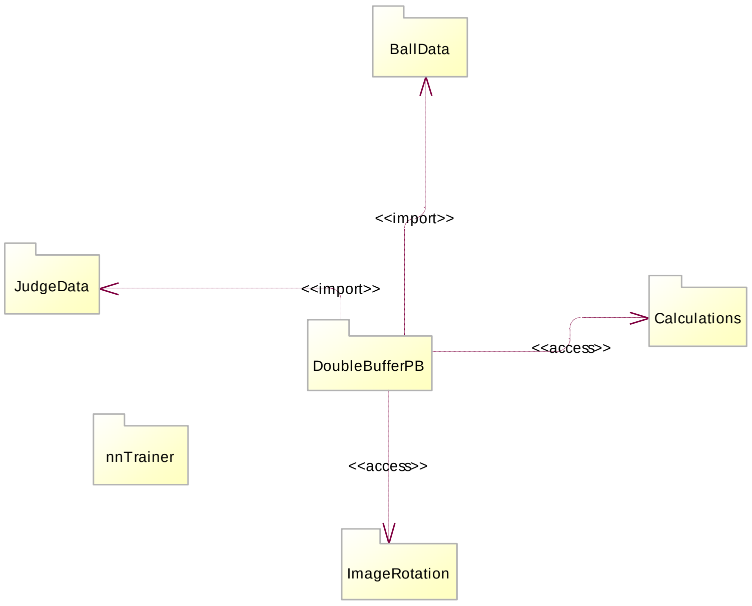


Figure 2
Main diagram of the nnUtility package

2.1 BBFBR Program - Structure and the Contents of the BallData Package

The package BallData, shown in Figure 3, contains the structure and logic classes that are needed for the description of the ball movement on a basketball court. The content of the BallData packages is the following:

- clsGridStructure - Class represents grid model which is applied on a basketball court, and divides it into quadrants and sub-quadrants. [8]
- clsBallPosition - Class defines the position of the ball on a basketball court
- clsBallPath - Class defines movement of the ball on a basketball court
- clsBallMovement - Class defines smooth ball movement on a basketball court for the purpose of animation
- clsAction - Class defines one basketball action
- clsActionDataFile - Class defines a binary file in which actions are stored
- clsActionFileWriter - Class contains a static method for reading and writing binary files in which basketball actions are stored

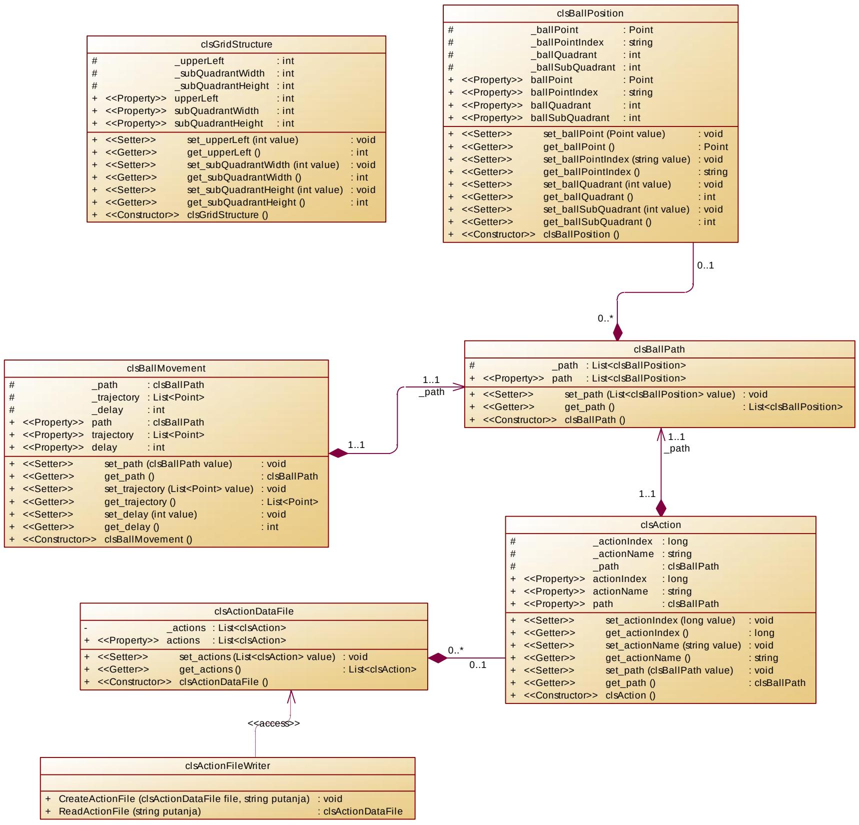

clsGridStructure
# _upperLeft : int
# _subQuadrantWidth : int
# _subQuadrantHeight : int
+ <<Property>> upperLeft : int
+ <<Property>> subQuadrantWidth : int
+ <<Property>> subQuadrantHeight : int
+ <<Setter>> set_upperLeft (int value) : void
+ <<Getter>> get_upperLeft () : int
+ <<Setter>> set_subQuadrantWidth (int value) : void
+ <<Getter>> get_subQuadrantWidth () : int
+ <<Setter>> set_subQuadrantHeight (int value) : void
+ <<Getter>> get_subQuadrantHeight () : int
+ <<Constructor>> clsGridStructure ()
clsBallPosition
# _ballPoint : Point
# _ballPointIndex : string
# _ballQuadrant : int
# _ballSubQuadrant : int
+ <<Property>> ballPoint : Point
+ <<Property>> ballPointIndex : string
+ <<Property>> ballQuadrant : int
+ <<Property>> ballSubQuadrant : int
+ <<Setter>> set_ballPoint (Point value) : void
+ <<Getter>> get_ballPoint () : Point
+ <<Setter>> set_ballPointIndex (string value) : void
+ <<Getter>> get_ballPointIndex () : string
+ <<Setter>> set_ballQuadrant (int value) : void
+ <<Getter>> get_ballQuadrant () : int
+ <<Setter>> set_ballSubQuadrant (int value) : void
+ <<Getter>> get_ballSubQuadrant () : int
+ <<Constructor>> clsBallPosition ()
0..1
0..*
clsBallPath
# _path : List<clsBallPosition>
+ <<Property>> path : List<clsBallPosition>
+ <<Setter>> set_path (List<clsBallPosition> value) : void
+ <<Getter>> get_path () : List<clsBallPosition>
+ <<Constructor>> clsBallPath ()
1..1
_path
1..1
clsBallMovement
# _path : clsBallPath
# _trajectory : List<Point>
# _delay : int
+ <<Property>> path : clsBallPath
+ <<Property>> trajectory : List<Point>
+ <<Property>> delay : int
+ <<Setter>> set_path (clsBallPath value) : void
+ <<Getter>> get_path () : clsBallPath
+ <<Setter>> set_trajectory (List<Point> value) : void
+ <<Getter>> get_trajectory () : List<Point>
+ <<Setter>> set_delay (int value) : void
+ <<Getter>> get_delay () : int
+ <<Constructor>> clsBallMovement ()
1..1
_path
1..1
clsAction
# _actionIndex : long
# _actionName : string
# _path : clsBallPath
+ <<Property>> actionIndex : long
+ <<Property>> actionName : string
+ <<Property>> path : clsBallPath
+ <<Setter>> set_actionIndex (long value) : void
+ <<Getter>> get_actionIndex () : long
+ <<Setter>> set_actionName (string value) : void
+ <<Getter>> get_actionName () : string
+ <<Setter>> set_path (clsBallPath value) : void
+ <<Getter>> get_path () : clsBallPath
+ <<Constructor>> clsAction ()
0..*
0..1
clsActionDataFile
- _actions : List<clsAction>
+ <<Property>> actions : List<clsAction>
+ <<Setter>> set_actions (List<clsAction> value) : void
+ <<Getter>> get_actions () : List<clsAction>
+ <<Constructor>> clsActionDataFile ()
<<access>>
clsActionFileWriter
+ CreateActionFile (clsActionDataFile file, string putanja) : void
+ ReadActionFile (string putanja) : clsActionDataFile

Figure 3
Structure and content of BallData Package

3 BBFBR - Appearance and Usage

In Figure 4, the appearance of the main form of the BBFBR program is presented. The mentioned form consists of three parts:

- Main menu
- Area for displaying, animating and rendering the results of calculated actions (Display Area)
- Area for action management

The main menu contains four submenus: File, Neural Network, Display and Actions.

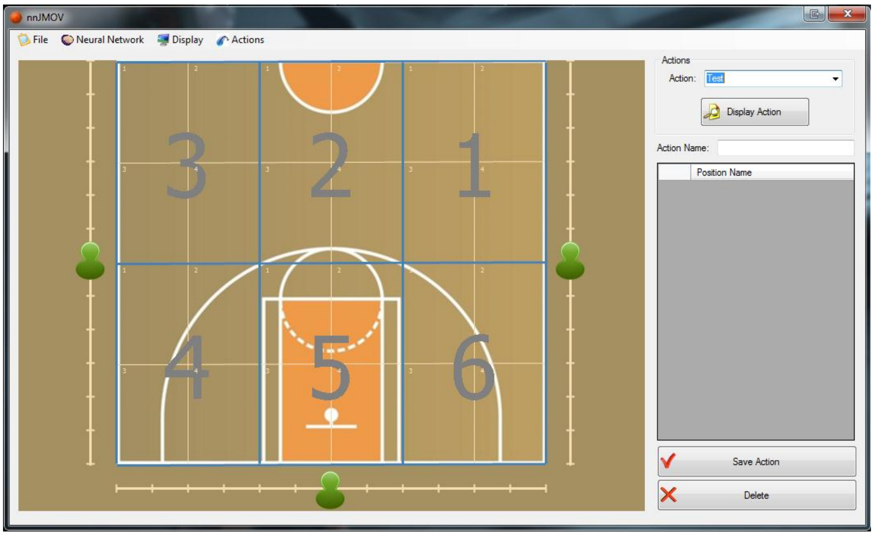


Figure 3
Appearance of the main form of the BBFBR program

The file submenu contains the following three options:

- *Open Actions File* – This option opens the binary file that contains the defined basketball actions. The file owns *.act extension.
- *Save Actions File* – This option records all changes to previously read *.act file that contains defined actions, or a new one is created if there is no existing *.act file.
- *Exit* – This option closes the program BBFBR.

The Neural Network submenu contains following three options:

- *Load Neural Network* – This option reads the binary file that contains the trained neural network. The extension of the file with the trained neural network is *.ann (Artificial Neural Network)
- *Train Neural Network* – This option calls NN Trainer form, which serves for training and adapting the neural network, for input, which is the movement of the ball in the court during a basketball action, giving as an output the optimal movement of the referees, in order to have the best observance of the current action.
- *Calculate Action* – This option uses the read or just draw new action to determine the optimal movement of the referees by the neural network, in order to gain the best observance of the action.

The Display submenu serves for modifying and managing the Display area of the BBFBR program. The Display area will be explained in detail later in the text. The Display submenu contains the following seven options:

- *Show Grid Lines* – This option displays the scheme that shows how the basketball terrain is divided into quadrants and sub-quadrants.
- *Show Quadrant Numbers* – This option displays the numbers that identify the quadrants of the basketball court.
- *Show SubQuadrant Numbers* – This option displays the numbers that identify the sub-quadrants of the basketball court inside a certain quadrant.
- *Show JMOV Paths* – This option displays the line of movement of the referees around the court.
- *Show Static Referees* – This option restricts the movement of the referees and places them in the predicted positions.
- *Animate* – This option animates the movement of the ball during the action and the referees who observe it.
- *Render* – This option exports the animation into a *.wmv file using the abilities offered by the option Animate.

The Actions submenu, contains following two options:

New Action – This option initialises the creation of a new action.

Reset Court – This option resets the currently chosen action, thereby deleting the path of the ball in the current action. If the path of the ball of a certain action is not recorded after deleting, it will not be removed from the set of actions, regardless of whether it is in the memory, in the stage of creation or in *.act file.

3.1 The BBFBR - Display Area - Area for Displaying, Animating and Rendering the Results of Calculated Actions

The Display area is used for defining an action by drawing the path of the ball, and for displaying and animating the movement of the ball and referees in the court. The appearance of the Display area is shown in Figure 5.

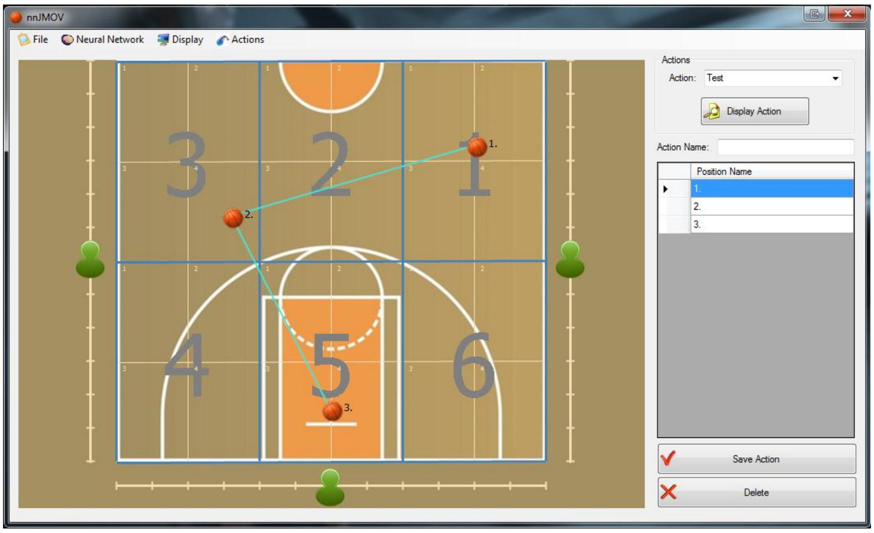


Figure 5
Appearance of the Display area

From Figure 5, it can be concluded:

- The Display area displays a half of the basketball court from the bird's-eye-view perspective.
- The basketball court is divided into six quadrants, and each of these six quadrants has four sub-quadrants [5] [6] [10] [11] [12]
- The quadrants and sub-quadrants mentioned can be marked and mapped differently, leaning on the structure provided by the clsGridStructure class. Depending on the values mentioned, the neural network is trained differently, but the input and the output vectors stay in the same format, which means that the results should simply be interpreted differently.
- On the side of the basketball court are three referees, who move according to their defined paths along the court, which is divided in precisely twelve equal distances between positions.
- Clicking on the basketball court, the ball is placed at a point in the action. The point receives the default name depending on the sequence of the number of the positioning. For example, if a point is the first point of the action, it is marked as “1.”; the second is marked with “2.”; etc. In order to achieve a better and clearer perceptive, these points can change their names into any common noun, for instance: dribbling, shooting, etc. An array of the action points represents the path of the ball movement during the action. Those points are matched using a blue line that automatically matches the last, currently added point, with the previous one.

- The appearance of the Display area does not affect the functionality of the Display component, but helps and makes drawing an action easier for a user.
- After loading or drawing an action, it is required to call the option Calculate Action from the menu Neural Network, so that the loaded neural network can calculate the optimal path and positions for the movement of the referees for that action. After executing the option mentioned, using the option Animate from the Display menu, it is enabled to animate the solution for the action drawing. The animation of the solution is presented in Figure 6.

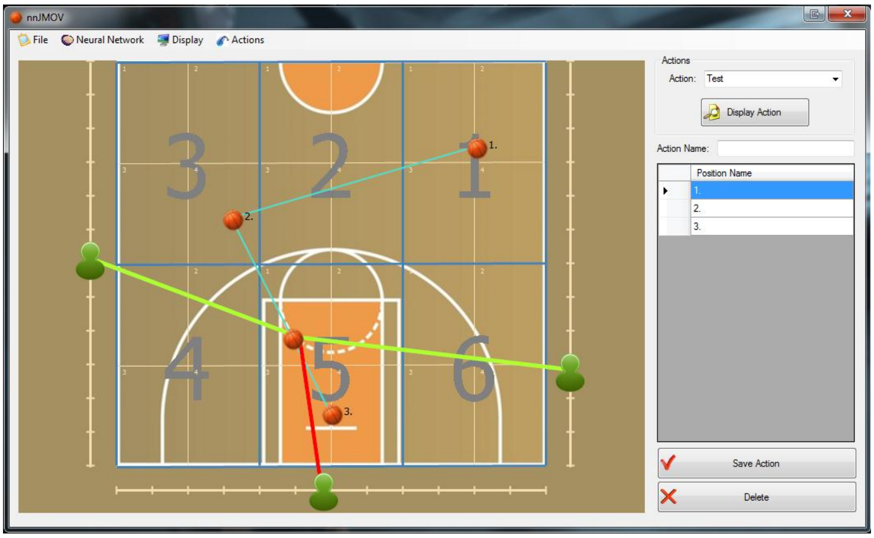


Figure 6
Animation of the solution

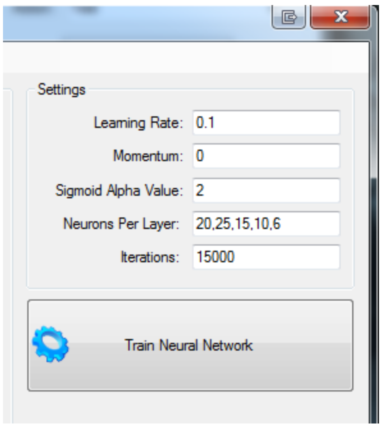
By analyzing Figure 6, it can be concluded that:

- During animation, the ball moves following the defined blue path meeting the points that define a basketball action.
- The referees move following the paths that are the result of the neural network calculations and they observe the ball movement.
- Depending on the values of the clsGridStructure and the schedule of the quadrants they are responsible for, between the ball and the referees two kinds of lines are drawn. If the ball is in a quadrant that the referee is responsible for, the line between him / her and the ball will be red. Otherwise, it will be green.

The area for the action management is used for displaying, modifying and deleting actions. For the reason the actions are drawn by clicking on the Display area, the area for the action management is basically used for naming the drawn action, renaming the dots of the action and saving in the temporary binary file. The section for the action management is on the right side of the Display area and is shown in Figure 6.

4 Neural Network Training for Analysis and Determination of the Ideal Path for Referees during a Basketball Action

In this paper, the trained neural network is the multilayer neural network, trained by the Back Propagation training algorithm that belongs to the class of controlled training algorithms. The structure of the hidden layers, regarding the number of neurons per layer, is shown in Figure 7, and the number of neurons is 30 in the first, input layer and 45 at the last, output layer. The trained neural network consists of seven layers with 30, 20, 25, 15, 10, 6 and 45 neurons in each layer, respectively. This neural network in hidden layers gradually narrows towards the last layer. All the neurons from a certain layer are connected to all neurons from the adjacent layer.


Settings

Learning Rate:	0.1
Momentum:	0
Sigmoid Alpha Value:	2
Neurons Per Layer:	20,25,15,10,6
Iterations:	15000

Figure 7
Settings of the neural network

From these facts it is visible that the input vector of neural network has 30 elements, and the output one 45. These elements' values are discrete ones. Depending on the length of a basketball action (no more than 15 key points), the appearance of the input and output vectors may change.

Every key point is determined by the ordered pair: [quadrant, sub-quadrant], while the ideal positions for the basketball referees for such point are determined by ordered three [referee1pos, referee2pos, referee3pos]. The values for the quadrant entity range between 1 and 6, and for the sub-quadrant between 1 and 4. The values for entities referee1pos, referee2pos and referee3pos are between 1 and 12, representing the fixed position from which the certain key point of basketball action may be seen best.

Let us suppose that a basketball action consists of four key points, and that we know the ideal positions for the basketball referees for these four points. Since the maximum length of the input vector of the neural network is 30 elements, or 15 points, noted as $t_1, t_2 \ldots t_n$, where max(n) = 15, defined by an ordered pair [quadrant, sub-quadrant], all values of the input vector are set to 0, and are filled, starting from the left-hand side, by ordered pairs, taking care to follow the structure [quadrant-t_1, subquadrant-t_1, quadrant-t_2, subquadrant-t_2 ....] etc. The output vector is formed in a similar way. Since the maximum length of the output vector of the neural network is 45 elements, or 15 points, noted as $t_1, t_2 \ldots t_n$, where max(n) = 15, defined by ordered three [referee1pos, referee2pos, referee3pos], all values of the input vector are set to 0 and are filled, starting from the left-hand side, by ordered threes, taking care to follow the structure [referee1pos -t_1, referee2pos -t_1, referee3pos -t_1, referee1pos -t_2, referee2pos -t_2, referee3pos -t_2,....] etc. Tables 1 and 2 show the basketball action consisting of four key points, described by ordered pairs [quadrant, sub-quadrant] and the ideal positions of the basketball referees for such action, described by ordered threes [referee1pos, referee2pos, referee3pos]. Tables 3 and 4 show the form of the input and output vectors of the neural network for the basketball actions given in Tables 1 and 2.

Table 1

Point coordinates

Point	Quadrant	Subquadrant
t_1	1	4
t_2	6	2
t_3	4	3
t_4	5	1

Table 2

Ideal positions of basketball referees

Point	referee1pos	referee2pos	referee3pos
t_1	4	6	5
t_2	1	8	9
t_3	5	3	1
t_4	10	6	7

Table 3
Input vector of neural network

I.V.E.V.	1	4	6	2	4	3	5	1	0	0	0	...	0
Index	0	1	2	3	4	5	6	7	8	9	10	...	29
Point	t_1		t_2		t_3		t_4						

Table 4
Output vector of the neural network

O.V.E.V.	4	6	5	1	8	9	5	3	1	10	6	7	0	...	0
Index	0	1	2	3	4	5	6	7	8	9	10	11	12	...	44
Point	t_1			t_2			t_3			t_4					

I.V.E.V. - Input Vector Element Values O.V.E.V. - Output Vector Element Values

The total set of the training data for this neural network consists of 43 differently defined actions and ideal paths for the referees for each one; for one action, there is only one ideal path. These actions vary in length, based on the previously established rule that they may not be longer than 15 key points. Table 5 shows the relation between the actions' lengths and the number of their instances in the total set of training data.

Table 5
Relation between actions length and number of their instances in a training data set

Action length	No. of instances
3	5
4	4
5	4
6	6
7	4
8	3
9	3
10	6
11	5
12	3
Total	43

The neural network was trained in two ways:

- By sequential repetition
- By sequential repetition with progressive action development

Neural network training by sequential repetition understands the passing of pairs of input and output vectors, one after another, until all 43 pairs are passed as intended to train the neural network. When all pairs (patterns) are passed, this cycle continues as many times as defined in the Iterations configuration of the neural network. Specifically, regarding settings from Figure 7, 43 pairs of input and output vectors will be passed through neural network 15,000 times, and thus the training will be done.

Neural network training by sequential repetition with progressive action development understands the passing of pairs of input and output vectors, one after another, but this action is treated progressively, i.e. from time aspect. Suppose that an action used for neural network training has four key points. First, the first key point of an action will be passed into the network, namely the ordered pair [quadrant, sub-quadrant] with corresponding output ordered three [referee1pos, referee2pos, referee3pos] as the input and output vector as shown in Tables 3 and 4. After that, in a similar way, the first and second key point will be passed through the neural network; then first, second and third one and so on until the full action length is reached. If this action is four key points long, the whole progressive sequence must be used for this input/output pattern, $[t_1]$, $[t_1,t_2]$, $[t_1,t_2,t_3]$, $[t_1,t_2,t_3,t_4]$, where t_n is the action key point, until there are no new patterns for training. When all pairs (patterns) are passed for training of the neural network, this cycle is repeated as many times as defined in the Iterations settings of the neural network.

Neural network training by the method of Sequential repetition with progressive action development is formed with the aim of minimizing the influence of input nodes on output nodes if those cannot occur within the given task. It has been mentioned that a structure of a formed neural network includes all the neurons from a layer being connected to all the neurons from an adjacent one. By bringing certain input vector into the neural network, all the neurons from the first layer are activated and the further propagation of the signal through the neural network activates all the output neurons of the network. In this way, any value in the output vector sequence depends on the previous values in the input vector sequence. During neural network training by the method of Sequential repetition, with direct training using the input vector comprised of all the points of action, from t_1 to t_n, and the output vector comprised of the complete positions list for the given key points of action, correlation is attained so that the values of the referee positions for point t_k, $k \subseteq n$, are strongly dependent on all previous points, i.e. on point set $S=\{t_1, \dots ,t_{k-1}\}$.

The advantages of neural network training using the method of Sequential repetition with progressive action development, in comparison to method of Sequential repetition, are summarized in two points:

- Decreased influence of input node sequence values on certain values of output node sequence values, reflected as solutions produced by thus trained neural network. These solutions are closer to expected, common-sense logical solutions for particular situation than solutions given by the neural network trained by the Sequential repetition method.

- Application of the Sequential repetition method with progressive action development over the same set of patterns for neural network training, quantitatively multiplied number of patterns used for its training. New patterns, formed during neural network training, were formed as subsets of existing patterns.

Now we will consider these theses in more detail, with explanations using data used in tests.

Neural network trained by the method of Sequential repetition gives satisfactory results for actions between 3 and 12 key points long, as defined in training set. Nevertheless, such a neural network could not manage with actions shorter than 3 and longer than 12 key points, which is quite logical. However, if a neural network knows how to find ideal paths for actions longer than 3 key points, why could it not, without any particular training for such cases, find ideal paths for referees in shorter actions?

The answer is given by the method of Sequential repetition with progressive action development. Since for every action defined and its ideal path, through defined points, the neural network is progressively trained, it will be passively trained by subset of actions shorter than the particular action being taught to the neural network. By this method, for action 3 key points long, the neural network will be, in a single pass for this action, also trained for actions that are 1, 2 and 3 key points long. Theoretically, this means that by defining a large number of actions (the currently maximal supported length is 15 key points) such a neural network may be trained in a very short time. Nevertheless, quality training of neural networks is based on the quality, representativeness, variety and validity of the examples used for the neural network training. In this way, neural network training based solely on 15 key points length actions decreases the quality of the neural network, especially bearing in mind that most basketball actions consists of 3 to 8 key points, while longer ones are rarely longer than 12 key points. Tables 6 and 7 show relations between action length and number of their instances in the training data set by using the method of Sequential repetition with progressive action development.

Based on the data from Tables 5 and 7, we may conclude that the method of Sequential repetition with progressive action development considerably increases the data set for training a neural network, in this case about 8 times, from 43 to 317 specimens of training data.

A larger training data set contributes to greater precision of the neural network, which improves its conclusions. In comparative test for actions similar to those used in the training, the output vectors were formed similar to those in similar action from training set. Specifically, the neural network that was trained by the method of Sequential repetition sometimes gave output vectors of position values deviating for +/- 2 to 3 notches (on a fixed scale of referee movement lines) from the common-sense logical expected value.

Table 6

Relation between action length and number of instances in data set for training using method of Sequential repetition with progressive action development

Table 7

Relation between action length and number of their instances in data set for training using method of Sequential repetition with progressive action development – final

Action length	No. of instances	Progressively
3	5	15
4	4	16
5	4	20
6	6	36
7	4	28
8	3	24
9	3	27
10	6	60
11	5	55
12	3	36
Total	43	317

Table 6

Action length	No, of instance
1	43
2	43
3	43
4	38
5	34
6	30
7	24
8	20
9	17
10	14
11	8
12	3
Total	317

Table 7

For actions for which the neural network was trained, independently of the method of training, it gave identical output vectors as attached to such action during the training. An increase or decrease of iteration number, before neural network training by the method of Sequential repetition, influenced the solution quality produced by such a trained neural network. By increasing the iteration number to 30000, such a trained neural network gave results close to results given by the neural network trained by the method of Sequential repetition with progressive action development. Nevertheless, the neural network trained by this method still gave better results. The assumption is that, during progressive action development for the neural network training, paths were shaped for action subsets formed in the course of this development, with strong difficulty coefficients that steered the neural network to a better and more optimal (expected) solution.

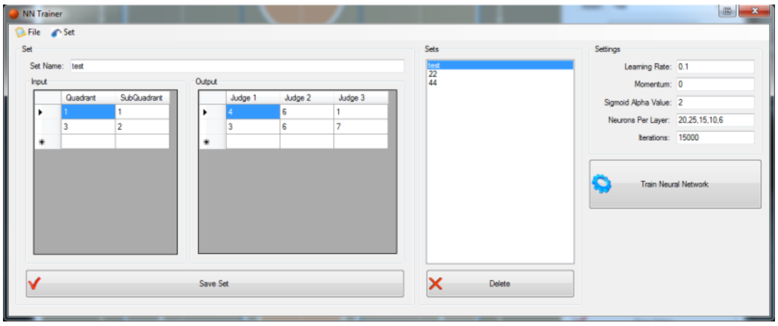
The neural networks, trained by these methods, had in all cases correctly established correlation between number of input and output points as input and output vectors. If input action has 4 points, input vector is represented by a sequence of 30 elements, from which the first 8 elements are values for the quadrant and sub-quadrant of points for the key actions formed by previously described rules, while all the other elements of input vector are equal to zero.

Based on this input vector, the output vector is formed as a sequence of 45 elements, from which the first 12 elements are values for referee positions based on the previously described rules, while the other elements of the output vector are equal to zero. For the calculation of ideal paths for animation purpose, zero values of the output vectors are neglected, while calculated values are read and formed based on the previously described rule.

5 NN Trainer - Appearance and Usage

The NN Trainer is called by activating the option Train Neural Network, in the Neural Network submenu of the main form of the BBFBR program. The NN Trainer is used, as the name implies, for training and configuring the neural network. The tool mentioned, which is shown in Figure 8, consists of three segments:

- Main menu
- Area for configuring the input and output data of the neural network
- Area for configuring the neural network



Set Name: test

Input

	Quadrant	SubQuadrant
▸	1	1
	3	2
*		

Output

	Judge 1	Judge 2	Judge 3
▸	4	6	1
	3	6	7
*			

Sets: test, 22, 44

Learning Rate:	0.1
Momentum:	0
Sigmoid Alpha Value:	2
Neurons Per Layer:	20,25,15,10,6
Iterations:	15000

Figure 8
Appearance of the NN Trainer tool

The main menu consists of two submenus: *File* and *Set*.

The file submenu contains four options:

- *Load Neural Network Training Data* – This option reads the *.nnet binary file that contains the actions which will serve for training the neural network
- *Save Neural Network Training Data* – This option creates the *.nnet binary file that contains the actions which will train or have already trained the neural network

- *Train Neural Network* – This option starts the training of the neural network by defined or read input and output data and clearly determined configuration of the neural network, which will be described in more detail later in the text.
- *Save Neural Network As* – This option saves the trained neural network in the binary file with the extension *.ann

The Set submenu has only one option - *New Set.* Activating the New Set option, the control for defining a new set of data for training the neural network, is initialised.

The area for configuring the input and output data of the neural network defines the data for training the neural network leaning on the quadrants and sub-quadrants of the basketball court and twelve fixed positions for each referee.

The section mentioned consists of two parts: The Set and the Sets section. The Set section consists of an input and output table. Rows of the input table are pairs quadrant and sub-quadrant, thereby showing the location of the ball in the court. The rows of the output table are arranged in threes, which, for one position, define the positions of the referees on the fixed scale from 1 to 12. Analysing picture 16, it can be concluded that, for a position of the ball in quadrant 1, sub-quadrant 1, referee 1 should be in position 4, referee 2 in position 6, and referee 3 in position 1. One set for training is one action, which is defined by the path of the ball which consists of the rows of the input table, and the trio from the rows of the output table. Each set of data for training has its own name.

The neural network chosen for solving this problem is a multi-layer neural network with a Sigmoid Bipolar activation function and is trained by the Back Propagation algorithm. In the settings section, all crucial parameters for training the neural network can be found. By analyzing Figure 7, the following parameters can be found:

- *Learning Rate* – This parameter shows the speed of the neural network's learning. The default value is 0.1
- *Momentum* – The momentum of the neural network. The default value is 0.
- *Sigmoid Alpha Value* – The value of the alpha parameter of the Sigmoid bipolar activating function of belonging. The default value is 2. The value of the output values of mentioned function are from -1 to 1.
- *Neurons Per Layer* – The number of neurons per a hidden layer of the neural network. The layers are separated with comma, whilst the numbers show how many neurons there are in each layer. In Figure 7, the value of the box Neurons Per Layer is: 20,25,15,10,6, which means that the neural network has 5 hidden layers. The first hidden layer has 20 neurons, second hidden layer has 25 neurons, etc.
- *Iterations* – The number of iterations required for training the neural network.

The input vector of the neural network has 30 elements, while the output vector has 45 elements, so, in conclusion, the trained neural network in this example has 7 layers in total. This enables the calculation of the ideal position of the referees for actions up to 15 key points, actually the key positions of the ball, although it is statistically shown that the action is usually finished within 6 to 10 key ball points.

Conclusions

Solution BBFBR has shown good results and the solution itself is still in the stage of development. The current realization of the BBFBR solution has implemented only a simple computation of the optimal path for the basketball referees in some actions. Therefore, it currently has only an educational purpose, which means that it can be used for training young basketball referees. During further development of the BBFBR solution, the paths of the basketball players will be implemented, so the percentage of visibility of a certain part of an action from the aspect of a certain referee can be determined. As a next step in the development, the implementation of the retroactive analysis with the research of the percentage of visibility of an action from the aspect of the field of visibility of a referee is planned, and if the results are satisfactory, it will be followed by the use of the adaptive neural networks, so the movement of the referees does not simply depend on the movement of the ball in the court, but on the position and the ability of observing on the part of the other referees as well. Calculating the percentage of visibility will be based on the rules of human visual field, fuzzy controllers and percentage coverage of the visual field. Based on the results mentioned, micro corrections of the referees’ positions will be possible, which means that fuzzy controllers will add to the outputs of the neural network. [7]

In addition, further plans for the BBFBR solution include the integration with data mining techniques based on video recording, animations and other techniques. [9]

Acknowledgements

This work was partially supported by the Serbian Ministry of Education and Sciences (Grant No: 171039).

References

[1] Zdravko Ivankovic, Milos Rackovic, Branko Markoski, Dragica Radosav, Miodrag Ivkovic, "Appliance of Neural Networks in Basketball Scouting", Acta Polytechnica Hungarica, Vol. 7, Issue 4, 2010

[2] José Manuel Sánchez Santos, Ana Belen Porto Pazos, Alejandro Pazos Sierra, "Team Performance in Professional Basketball: an Approach Based on Neural Networks and Genetic Programming ", XIII IASE and III ESEA Conference of Sports, Prague, May 2011

[3] Michael E. Young, "Nonlinear Judgment Analysis: Comparing Policy Use by Those Who Draft and Those Who Coach", Psychology of Sport and Exercise, Volume 9, Number 6, November 2008

[4] Bernard Loeffelholz, Earl Bednar, Kenneth W. Bauer "Predicting NBA Games Using Neural Networks", Journal of Quantitative Analysis in Sports: Vol. 5, Issue 1, Article 7, 2009

[5] Dean Oliver, "Basketball on Paper – Rules and Tools for Performance Analysis", Brassey's, Washington DC, 2004

[6] Ratgeber, L. Play from a Game: (Head Coach). Mizo Pecs 2010. 2007/2008. Mizo Pecs 2010 vs. Euroleasing Sopron

[7] Rita Lovassy, László T. Kóczy, László Gál, "Function Approximation Performance of Fuzzy Neural Networks" Acta Polytechnica Hungarica, Vol. 7, Issue 4, 2010

[8] Martin Sarnovský, Peter Butka, Ján Paralič, "Grid-based Support for Different Text Mining Tasks", Acta Polytechnica Hungarica, Vol. 6, Issue 4, 2009

[9] Bojan Kuljić, János Simon, Tibor Szakáll "Pathfinding Based on Edge Detection and Infrared Distance Measuring Sensor" Acta Polytechnica Hungarica, Vol. 6, Issue 1, 2009

[10] Vasiljevic P. Markoski B., Ivankovic Z., Setrajcic J., Milosevic Z., "Basket Supervizor- Collectiom Statistiacal Data in Basketball and Net Casting" Technics Technologies Education Management-TTEM, 169-178, 2011

[11] FIBA – Basketbal Statisticians Manual (2008)

[12] FIBA – Basketbal Statisticians Manual (2011)