Forward Control of Robotic Arm Using the Information from Stereo-vision Tracking System

Michal Puheim[1], Marek Bundzel[2], Ladislav Madarász[3]
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Košice, Slovak Republic
[1]michal.puheim@tuke.sk, [2]marek.bundzel@tuke.sk, [3]ladislav.madarasz@tuke.sk

***Abstract*—In this paper we present the feed-forward neural network controller of robotic arm, which makes use of tracking method applied to stereo-vision cameras mounted on the head of the humanoid robot Nao, in order to touch the tracked object. The Tracking-Learning-Detection (TLD) method, which we use to detect and track the object, is known for its state-of-art performance and high robustness. This method was adjusted to be usable with a stereo-vision camera system, in order to provide 3D spatial coordinates of the object. These coordinates are used as the input for the feed-forward controller, which controls the arm of a humanoid robot. The goal of the controller is to move the hand of the robot to the object by setting arm joints into position corresponding to the object location. The controller is implemented as an artificial neural network and trained using the error back-propagation algorithm. The experiment, which demonstrates the proof of the concept, is also denoted in this paper.**

Keywords—arm controller, neural network, TLD, tracking, learning, detection, humanoid robot, Nao, stereo-vision

I. INTRODUCTION

This paper proposes a possible application of Tracking-Learning-Detection (TLD) [1]–[3] algorithm in stereo-vision on the Nao robot [4]. TLD is able to detect and track an object in a continuous series of video frames. Position of the object is defined as coordinates of its bounding box (x, y, width, height). This gives us the idea about the position of the object in the frame, but no further information about its location in the environment. This problem occurs due to the fact that TLD is only processing two-dimensional picture and the information about the third dimension is not available.

In order to obtain the additional information, it is possible to employ another detector using the camera with different viewpoint. With the information about the object position in the frames from both cameras and the information about mutual position of these cameras, we are able to determine the position of the object in three-dimensional space quite precisely.

Then we can use this information as the input for the forward arm controller, based on the inverse control model with goal to enable the robot's interaction with the tracked object in three-dimensional environment.

II. TRACKING-LEARNING-DETECTION

Tracking-Learning-Detection (TLD) [1]-[3] is an object tracking method usable for long-term tracking of arbitrary objects in unconstrained environments [2]. TLD system is made of three independent components:

- Tracker – a short term tracker based on Lucas-Kanade method [5], which is used to generate examples for training of the detector.
- Detector – has a randomized forest form [6], enables incremental update of its decision boundary and real-time sequential evaluation during run-time [2]. Runs independently from the tracker.
- Learning algorithm – so called "P-N Learning" [1] uses the tracker to generate positive (P) and also negative (N) examples that are further used in order to improve the model of the detector.

Before the tracking begins, the bounding box of the object is manually selected by the supervisor. After that, during TLD runtime, the object is tracked by the tracker component. During the tracking, the model of the object is built for use with the detector component. It is built upon the information from the first frame, as well as the information provided by the tracker. If the detector has successfully finished the training process, it enables re-detection of the object, whenever the tracker fails. Both detector and tracker make errors, the stability of the whole system is achieved by mutual cancellation of these errors [2].

A. Tracking

The tracker used with TLD, as proposed by [3], is Median-shift (or Median-flow) tracker. It is a rectangular shaped tracker based on the Lucas-Kanade tracker [5], which is robust to partial occlusions and also estimates translation and scale. It is designed to track number of points between consecutive frames and estimates a rectangle displacement and scale change using median of tracked points. Tracked points are selected as 50 % of the most reliable points within tracked rectangle, using forward-backward error, as proposed by [3]. In each frame a completely new set of feature-points is tracked, which makes the tracker very adaptive [2].

The forward-backward error is defined as a difference between two feature-point trajectories, where the first one is defined by forward motion of tracked point and the second one by its backward motion. The next paragraph defines this error according to [3].

Let $S = (I_t, I_{t+1}, \ldots, I_{t+k})$ be an image sequence and x_t be a point location in time t. Using an arbitrary tracker, the point x_t is tracked forward for k steps. The resulting trajectory is $T_k^f = (x_t, x_{t+1}, \ldots, x_{t+k})$, where f stands for forward and k indicates the length. Our goal is to estimate the error of trajectory T_k^f within the image sequence S. For this purpose, the validation trajectory is constructed first. Point x_{t+k} is tracked backwards, up to the first frame and produces $T_k^b = (x'_t, x'_{t+1}, \ldots, x'_{t+k})$, where $x'_{t+k} = x_{t+k}$. The final forward-backward error for given point is then given as the Euclidean distance between x_t and x'_t.

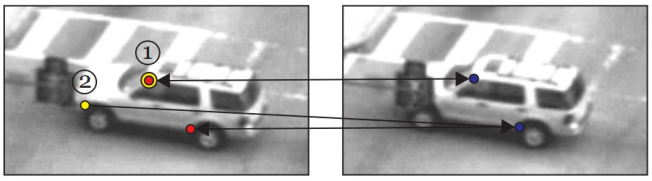

Fig. 1. Tracking of the point trajectory during forward-backward error calculation. Main idea is that some points cannot be re-tracked to their initial position [3].

This kind of recursive tracking is possible as long as the tracked object is visible in the image. The tracker will eventually fail in case of occlusion or other dynamic change in the appearance of the object. In this case the median shift is greater than the accepted threshold and the recursive tracking is forced to stop. When this happens, tracker is unable to reinitialize itself, because it lacks mechanism for object detection [7]. In this situation the TLD relies on the detector component.

B. Detection

A detector of the TLD system is based on scanning window [8] and the randomized fern forest classifier [6]. The training and testing is on-line [1]. Using the scanning window approach, the input image is scanned across possible positions and scales and at each sub-window a binary classifier decides about presence of the object [1]. The following paragraphs describe detector design as explained in [2].

The model of the given object is defined by a set of image patches, which represent possible appearances of the object. Each patch is described by a number of local 2bit Binary Patterns (described in Fig. 2), which position, scale and aspect ratio were generated at random. These features are randomly partitioned into several groups of the same size. Each group represents a different view of the patch appearance. The response of each group is represented by a discrete vector that is called a "branch" [2].

The classifier used for detection has the randomized forest form [6]. It enables online update and sequential evaluation. The forest consists of several trees, each of them is build from one group of features. Every feature in the group represents a measurement taken at a certain level of the tree [2].

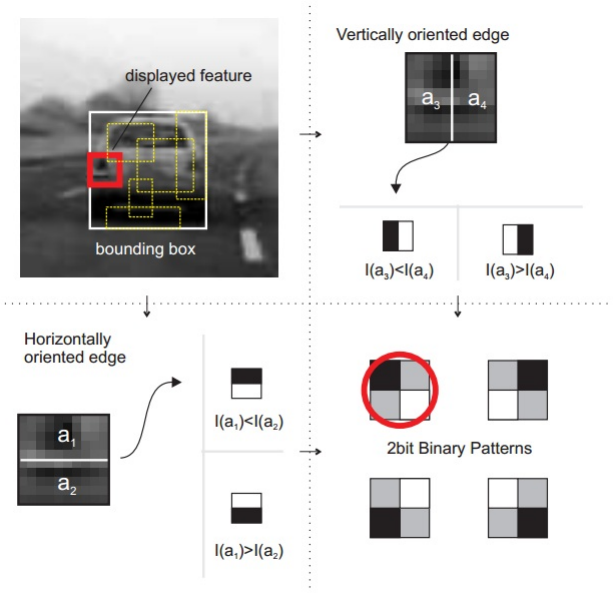

Fig. 2. Local 2bit Binary Patterns used in object detector. Features encode local gradient orientation within the object bounding box. Their position within image patch, scale and aspect ratio are generated at random [2].

At the beginning, every tree contains one single branch defined by the selected patch. With every positive example a new set of branches is added to the forest. Growing and pruning of the forest is therefore incremental – one example is processed at a time [2].

Evaluation of an unknown patch by the tree is very efficient. The features within each tree are measured sequentially. If the patch reaches the end of the tree, it is considered positive. Otherwise, if it differs from the defined branches, the measurement is terminated and the patch is considered negative. The sequential nature of the randomized tree enables real-time evaluation on a large number of positions. The input image is scanned with a sliding window. At each position, every tree makes a decision, whether the underlying patch is in the model or not. The final decision is obtained by the majority vote [2].

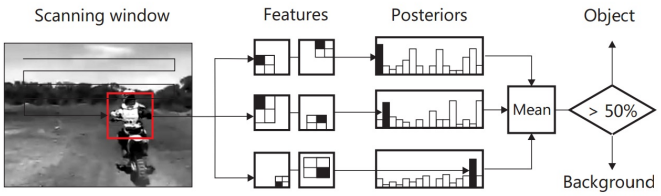

Fig. 3. Detector based on scanning window and randomised fern forest classifier [1].

C. Learning

Learning algorithm, employed in the TLD system, is called P-N Learning [1]. It uses positive (P) and negative (N) examples that are used in order to improve the model of the

detector. Main idea of this approach is to make use of inevitable errors of both tracker and detector. The stability of the system is achieved by error cancellation of both components. The rest of this section gives a brief description of P N Learning according to [1].

Training of the initial classifier is performed in the first frame. The posteriors from all ferns in the classification tree are initialized to zero and are updated by positive examples generated by affine warping of the selected patch. The classifier is then evaluated on all patches. The detections far from the target represent the negative examples and further updates the posteriors [1].

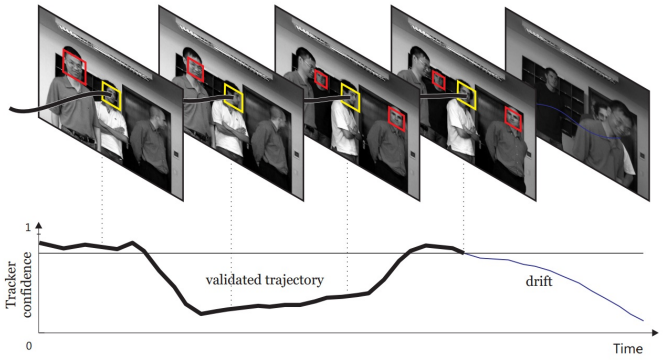

Fig. 4. A trajectory of an object represents a structure in video..Patches close to the trajectory are positive (YELLOW), patches far from the trajectory are negative, only the difficult negative examples are shown (RED). In training, the real trajectory is unknown,but parts of it are discovered by a validated adaptive tracker [1].

When the P-N learning is initialized in the first frame by training the Initial Detector, the initial position of the Lucas-Kanade tracker is set using the selected patch. Afterwards, for each of the consecutive frames, the detector and the tracker find the location(s) of the object. As depicted in Fig. 4 and Fig. 5, the patches close to the trajectory, given by the tracker and detections far away from this trajectory are used as positive and negative examples respectively. If the trajectory is validated (i.e. forward-backward consistent), these examples are used to update the detector. However, if there is a strong detection far away from the track, the tracker is reinitialized and the collected examples are discarded [1].

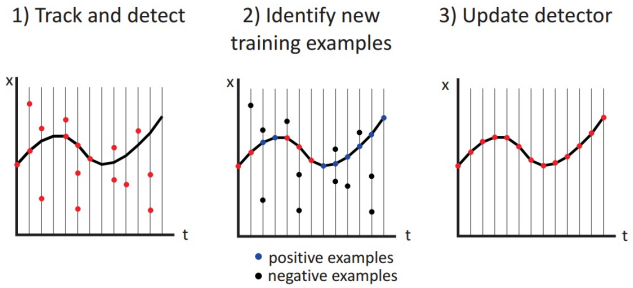

Fig. 5. Representation of the P-N Learning. Object is tracked by the tracker and simultaneously detected by the detector. Patches close to the trajectory update the detector with positive label, other patches update the detector with negative label [9].

The trajectory of the tracker is then denoted as P-N Tracker, since it is reinitialized by a detector trained by online P-N learning. The detector that is obtained after processing all frames of the sequence is called Final Detector [1].

III. Stereo-vision Tracking System

An advantage of stereo-vision systems is their ability to extract more information from an image than typical single camera systems [10]. Goal of the proposed stereo-vision tracking system [15] is to determine the spatial coordinates of the tracked object in the three-dimensional environment and provide it to the arm controller. The arm controller is then supposed to use this information in order to move hand of the robot towards the object.

A. Calculation of Depth in Stereo-vision Systems

Let us have two identical cameras with parallel optical axes (i.e. in rectified configuration), as shown in Fig. 6. If f is the focal distance of the cameras, c is the distance between the cameras, $a = c/2$ is the distance of cameras from the central optical axis, X_R is the x-coordinate of the object in the image from the right camera and X_L is the x-coordinate of the object in the image from the left camera, then by using the similarity of triangles we get the following equations:

$$\frac{X_L}{f} = \frac{x-a}{z}, \quad \frac{X_R}{f} = \frac{x+a}{z} \tag{1}$$

By merging these equations and elimination of x we get the formula, which can be used to calculate the z coordinate:

$$z = \frac{-2af}{X_L - X_R} \tag{2}$$

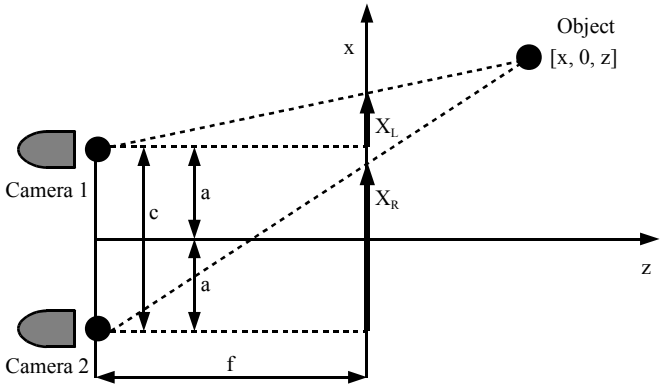

Fig. 6. Estimation of object z coordinate using the stereo cameras with parallel optical axis (i.e. in rectified configuration). Distance between the cameras is $c = 2a$, where a is distance of the camera from the optical axis. X_R and X_L are coordinates of the object as seen by right and left camera respectively. If f is focal length then z coordinate can be calculated using X_R and X_L.

The equation (2) basically expresses that with known camera parameters f and d, we can use the difference in vision of the same object by both cameras in order to determine the distance of the object from the image plane z (and also the distance of the object from the cameras). The difference in vision between two cameras is called disparity [10].

B. *Integration of TLD in the Stereo-vision Tracking System*

In spite of the simplicity of the formula used to calculate depth, stereo-vision is not a trivial task, particularly because of the correspondence problem [11]. The root of this problem is that search of the corresponding points (and objects) in the images from both cameras is difficult and usually requires sophisticated algorithms based on the edge detection (e.g. RANSAC [12]). In order to solve this problem for the purpose of this work, we used the combination of two TLD systems tracking the same object simultaneously, one on each camera.

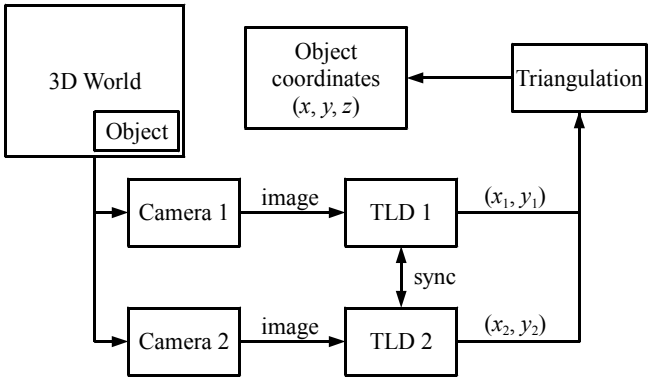

Fig. 7. Simplified block diagram of the stereo-vision tracking system which employs the TLD method for object tracking. Tracked object is captured by two cameras at the same time. Images from the cameras are used as the input for two synced TLD systems. Each of them produces the 2D coordinates of the target object in respective image. The couple of 2D coordinates is then further used to calculate 3D coordinates of the target by means of the triangulation method.

The proposed stereo-vision tracking system uses two TLD subsystems (see Fig. 7), which are based on the implementation by George Nebehay [7]. Tracked object is selected manually by the human operator from the image of single camera. At this point, the initial object model for the first TLD subsystem is created. This model is then copied to the second TLD subsystem. Both models are regularly synchronized in order to avoid divergence in object detection. Outputs of each subsystem are the coordinates of the object bounding box (x, y, w, h) in respective camera image (see Fig. 8). Coordinates of the object itself within the image are defined as the center (x_c, x_c) of the respective bounding box.

Let us assume that w_i is the image width (in pixels), h_i is the image height (in pixels), f is the focal distance of the cameras (in pixels) and c is the distance between cameras (in meters). If (x_L, y_L) are the center coordinates of the object on the left image and (x_R, y_R) are the center coordinates of the object on the right image (given in pixels), then the 3D coordinates of the target object (x, y, z) can be calculated as:

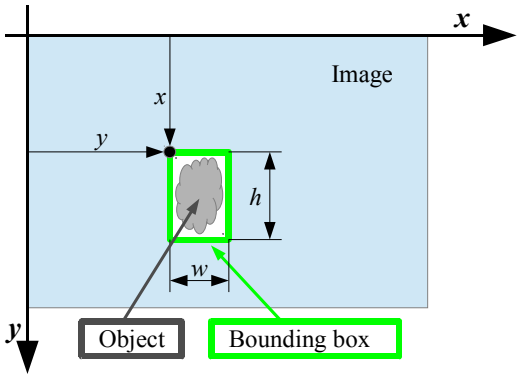

Fig. 8. TLD tracks the kernel of the object which is delimited by bounding box. It is defined by horizontal and vertical coordinate of the upper left corner, width and height. These four values (x, y, w, h) represent the output of the TLD system.

$$z = \frac{-cf}{X_L - X_R}, \tag{3}$$

$$x = \frac{c}{2} + z\frac{\left(X_L - \frac{w_i}{2}\right)}{f}, \quad y = -z\frac{\left(X_R - \frac{h_i}{2}\right)}{f}. \tag{4}$$

In application to humanoid robots, it is more convenient to use polar coordinates of the object instead of the Cartesian coordinates. These are given as (d, α, β), where d is the distance of the object and α, β are directional angles. The distance d of the target object from the cameras is the length of the vector (x, y, z):

$$d = \sqrt{x^2 + y^2 + z^2}\ . \tag{5}$$

If λ_v is the vertical field of view of the camera and λ_h is the horizontal field of view of the camera, then the directional angles α, β can be approximated as:

$$\alpha \approx \lambda_v\left(\frac{y_L}{h_i} - 0{,}5\right), \quad \beta \approx \lambda_h\left(\frac{x_L + x_R}{-2w_i} - 0{,}5\right). \tag{6}$$

IV. Robotic Arm Controller

Goal of the controller is to move the hand of the robot to the position of the tracked object using the information about the object, provided by the tracking system. In order to achieve this goal, we used feed-forward neural network with three inputs and three outputs.

In order to train the neural network, it is necessary to generate a sufficient set of training data. It can be created by exploiting the inverse model of the system.

A. Topology of the Artificial Neural Network

Inputs of the artificial neural network are polar coordinates of the object position related to the robot body (which means that head pitch and yaw are also taken into account when we determine the object directional angles). Outputs are the values of joint positions of the robotic arm corresponding to the current object position in 3D space. There are two hidden layers within the neural network, each including four neurons.

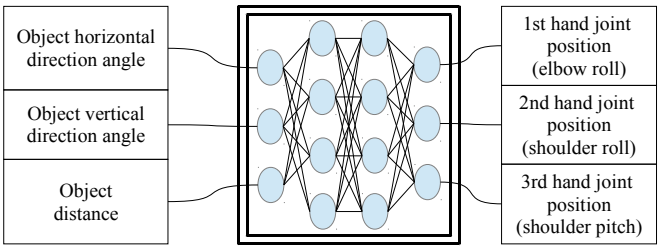

Fig. 9. Arm controller of the humanoid robot is based on the feed-forward neural network. Inputs are the information about the object position related to the robot's body. Outputs are the positions of the arm joints. Setting arm joints to these positions will effectively move robot's hand to the position of the tracked object.

B. Training of the Neural Network

We set the robotic arm to use three degrees of freedom, i.e. the movement is enabled on three joints only. Now let us assume, that these joints are "shoulder roll", "shoulder pitch" and "elbow roll" or (s_1, s_2, s_3). Let us assume, that the tracked object is some random position around the robot. The task is to determine the joint position (s_1, s_2, s_3) corresponding to the polar spatial coordinates of the object (d, α, β). In other words, the task is to perform the transformation:

$$\mathrm{T}: (d, \alpha, \beta) \rightarrow (s_1, s_2, s_3), \tag{7}$$

which is not possible without an adequate mathematical model or feedback control. Other option to carry out the transformation T is to use the trained neural network.

In order to generate the training data, we put the tracked object into the robot's hand. If we put the arm joints in one certain position (s_1, s_2, s_3), we can use the stereo-vision tracking system in order to get the polar spatial coordinates (d, α, β) of the object in robot's hand. In other words, it is possible to carry out the inverse transformation:

$$\mathrm{T}': (s_1, s_2, s_3) \rightarrow (d, \alpha, \beta). \tag{8}$$

We can use it to generate corresponding pairs of input and output data $[(d, \alpha, \beta); (s_1, s_2, s_3)]$. This data can be used in training of the neural network arm controller using the error back-propagation algorithm.

In order to achieve satisfactory results of the trained neural network it is necessary to normalize the input and output data on the interval between 0 and 1. This fact may become a problem, for example if the domain of the given input (or output) is infinitely large. In this case, it is necessary to limit its range to appropriate values. For example maximum distance range should be limited to the length of robot's arm. Also the minimum distance range should be limited according to the limits of the stereo-vision tracking system. Similar considerations should be made for other input/output domains [16].

V. Experiment

In order to test the applicability of the proposed arm controller, we performed an experiment, which tested the ability to touch the tracked object. We put the object 20 times in several different positions within reach of the robot's hand. In each attempt to touch the object, we determined its success. Results are shown in Table I.

The experiment has proven that the proposed combination of stereo-vision tracking system and neural network arm controller is feasible. Precision of the hand movement is sufficient for the feed-forward controller. It can be further improved by implementation of feed-back control.

In the current stage of development, the proposed object tracking system is not able to provide more information about the target object except for its location. This can be a problem if it is necessary to manipulate with the objects of different sizes or shapes. This problem can be solved by employing a different object tracking method (see [13]) or by tracking various parts of the tracked objects separately in order to determine its orientation additionally.

VI. Conclusion

In this work we made a brief overview of TLD system architecture, as well as its individual components. Illustration of its possible application in stereo-vision tracking system has also been mentioned. We applied this system to the Nao humanoid robot and used its output information about the tracked object as the input for the arm controller in order to enable the robot to interact with the tracked object. We carried out the experiment in which we demonstrated the viability of the proposed system.

At present the proposed system is running on the PC and connecting to the Nao robot via proxy connection. For future work it would be interesting to employ the proposed system on the cloud, which would enable multiple robots to take advantage of it [17][18].

Acknowledgement

The work presented in this paper was supported by VEGA, Grant Agency of Ministry of Education and Academy Science of Slovak Republic under Grant No. 1/0298/12 – "Digital control of complex systems with two degrees of freedom." The work presented in this paper was also supported by KEGA under Grant No. 018TUKE-4/2012 – "Progressive methods of education in the area of control and modeling of complex systems object oriented on aircraft turbo-compressor engines."

TABLE I. RESULTS OF THE HAND CONTROLLER EXPERIMENT

	NN Inputs (object coordinates)			NN Outputs (hand joint positions)			OK?
	d **(m)**	***v*** **(rad)**	***h*** **(rad)**	***sp*** **(rad)**	***sr*** **(rad)**	***er*** **(rad)**	
1.	0.134146	0.027178	0.290937	0.057461	0.357181	-0.788319	Y
2.	0.190570	-0.068388	0.589369	-0.172792	0.477867	-0.169463	Y
3.	0.198999	-0.073607	0.765424	-0.184782	0.740034	-0.160568	Y
4.	0.164649	0.103553	0.424750	0.061034	0.431430	-0.588861	Y
5.	0.221017	-0.173743	0.923930	-0.495389	0.778589	-0.039320	Y
6.	0.223769	0.021826	0.735878	-0.166607	0.515333	-0.049318	Y
7.	0.162134	-0.435731	0.842525	-0.614742	1.071152	-0.477502	N
8.	0.194244	-0.387002	1.168614	-0.553836	1.345710	-0.174646	N
9.	0.191741	-0.055399	0.831764	-0.122697	0.975315	-0.333835	Y
10.	0.185708	-0.190491	0.733462	-0.329463	0.733202	-0.207705	Y
11.	0.153799	-0.011105	0.150542	-0.038457	0.190594	-0.393123	Y
12.	0.163303	0.320239	0.386022	0.225971	0.462984	-0.762712	Y
13.	0.185293	0.017649	0.357128	-0.072228	0.261859	-0.185558	Y
14.	0.217610	-0.034607	0.627409	-0.253171	0.383026	-0.042686	Y
15.	0.201207	-0.011139	0.518828	-0.137416	0.347757	-0.096682	Y
16.	0.209144	-0.268592	1.175758	-0.458409	1.302464	-0.116085	Y
17.	0.129244	-0.330636	0.627409	-0.385313	0.940373	-1.098963	N
18.	0.141164	-0.175151	0.122507	-0.166845	0.178159	-0.397670	N
19.	0.193073	-0.401333	0.510929	-0.773710	0.276209	-0.030392	Y
20.	0.193977	0.237270	0.403652	0.181841	0.320561	-0.257261	Y
	Successful touch: 16			**Failed touch: 4**		**Percentage: 80 %**	

Each row represents one measurement, distance values d are given in meters, all other values are given in radians. Inputs of the neural network hand controller: d – object distance, v – vertical directional angle of the object, h – horizontal directional angle of the object. Outputs of the neural network hand controller: sp – shoulder pitch joint, sr – shoulder roll joint, er – elbow roll joint. Last column determines if the touch was successful or not.

REFERENCES

[1] Z. Kalal, J. Matas, K. Mikolajczyk: "P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints." In Conference on Computer Vision and Pattern Recognition. 2010.

[2] Z. Kalal, J. Matas, K. Mikolajczyk: "Online learning of robust object detectors during unstable tracking." In 3rd On-line Learning for Computer Vision Workshop 2009, Kyoto, Japan, IEEE CS, 2009.

[3] Z. Kalal, J. Matas, K. Mikolajczyk: "Forward-Backward Error: Automatic Detection of Tracking Failures." In International Conference on Pattern Recognition, Istambul, Turkey, 23-26 August, 2010.

[4] Aldebaran Robotics: Nao Website, [Online]. 2013. Available: http://www.aldebaran-robotics.com/en/

[5] B. D. Lucas and T. Kanade: "An iterative image registration technique with an applica-tion to stereo vision." In Proceedings of the International Joint Conference on Artificial Intelligence, pages 674–679, 1981.

[6] L. Breiman: "Random forests." ML, 45(1):5–32, 2001.

[7] G. Nebehay: "Robust Object Tracking Based on Tracking-Learning-Detection." Diplomarbeit. Fakultät für Informatik, Technische Universität Wien. 2012.

[8] P. Viola, M. Jones: "Rapid object detection using a boosted cascade of simple features." CVPR, 2001

[9] Z. Kalal, J. Matas, K. Mikolajczyk: Poster – Components of TLD [Online]. Available: http://kahlan.eps.surrey.ac.uk/featurespace/tld/Publications/2010_cvpr_demo_poster.pdf

[10] R. Hartley and A. Zisserman: "Multiple View Geometry in Computer Vision", Cambridge University Press, 2004.

[11] Abhijit S. Ogale and Yiannis Aloimonos: "Shape and the stereo correspondence problem". International Journal of Computer Vision 65 (1). October 2005.

[12] Martin A. Fischler and Robert C. Bolles: "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". Comm. of the ACM 24 (6): 381–395. June 1981.

[13] A. Yilmaz, O. Javed, M. Shah.: "Object Tracking: A Survey." ACM Comput. Surv. 38, 4, Article 13, 45 pages. Dec. 2006.

[14] M. Puheim.: "Application of TLD for object tracking in stereoscopic images". Diploma thesis (in slovak). Technical University of Košice. Faculty of Electrical Engineering and Informatics. Košice. 2013. 68 pages.

[15] M. Puheim , M. Bundzel, P. Sinčák, L. Madarász: "Application of Tracking-Learning-Detection for object tracking in stereoscopic images". Symposium on Emergent Trends in Artificial Intelligence and Robotics. Košice. Sep. 2013.

[16] Nagy, I., Várkonyi-Kóczy, A.R., Dineva, A.: GA Optimization and Parameter tuning at the Mobile Robot Map Building Process. IEEE 9th International conference on Computational Cybernetics, July 8-10, 2013. Tihany, Hungary. pp. 333-338. ISBN 978-1-4799-0060-2.

[17] Jordán, S., Haidegger, T., Kovács, L., Felde, I., Rudas, I.: "The Rising Prospects of Cloud Robotic Applications.", IEEE 9th International conference on Computational Cybernetics, July 8-10, 2013. Tihany, Hungary. pp. 327-332. ISBN 978-1-4799-0060-2.

[18] Tar, J.K., Rudas, I., Kósi, K., Csapó Á., Baranyi, P.: "Cognitive Control Initiative". 3rd IEEE International Conference on Cognitive Infocommunications, December 2-5, 2012, Košice, Slovakia. pp. 579-584. ISBN 978-1-4673-5188-1.