Proceedings of the Post-Graduate Conference on Robotics and Development of Cognition
10-12 September 2012, Lausanne, Switzerland

Towards Spatial Perception: Learning to Locate Objects From Vision

Jürgen Leitner, Simon Harding, Mikhail Frank, Alexander Förster, Jürgen Schmidhuber

Dalle Molle Institute for Artificial Intelligence (IDSIA),
Scuola universitaria professionale della Svizzera Italiana (SUPSI), and
Universitá della Svizzera Italiana (USI), Lugano, Switzerland
juxi@idsia.ch

Abstract

Our humanoid robot learns to provide position estimates of objects placed on a table, even while the robot is moving its torso, head and eyes (cm range accuracy). These estimates are provided by trained artificial neural networks (ANN) and a genetic programming (GP) method, based solely on the inputs from the two cameras and the joint encoder positions. No prior camera calibration and kinematic model is used. We find that ANN and GP are both able to localise objects robustly regardless of the robot's pose and without an explicit kinematic model or camera calibration. These approaches yield an accuracy comparable to current techniques used on the *iCub*.

Index Terms: spatial understanding, object localisation, humanoid robot, neural network, genetic programming

1. Introduction

The majority of robotic systems used nowadays are still mainly performing pre-programmed automation tasks. In recent years progress has been made in enabling these robotic systems to perform more autonomous behaviours. Increasing these capabilities is necessary for future use of robots in interesting settings of daily living, such as, household tasks, grocery shopping and elderly care. An important step to perform autonomous decisions and actions is to perceive the state of the environment. Perception though is still a hard problem in robotics.

We are interested in robust approaches to visual perception, with applications to object localisation while the robot is controlling its torso, head and gaze. The localisation will be used in combination with on-line motion planning for object manipulation tasks on a real humanoid robot. In this work, we focus on a machine learning setup that provides the robot with a method to estimate the location of objects relative to itself in 3D Cartesian space. Our research platform is the *iCub* humanoid robot [1], an open robotic system, providing a 41 degree-of-freedom (DOF) upper-body, comprising two arms, a head and a torso. Its visual system is a pair of cameras mounted in the head in a human-like fashion (see Fig. 1), providing passive, binocular images.

The problem of localising objects in 3D Cartesian space given two images from cameras in different locations is widely known in the computer vision literature as 'Stereo Vision'. In the following discussion, *CSL* and *CSR* refer to the local reference frames of the left and right cameras respectively, the reference frame of the body is *CSBody*, but as it is mounted at a fixed point this is also the reference frame chosen for the environment. Therefore *CSWorld* denotes the common environmental reference frame, in which we seek to express object locations. Cameras that photograph the same scene from two different locations provide different 2D projections of the 3D scene. If the 'intrinsic parameters' that specify each camera's projection from 3D to 2D, as well as the 'fundamental matrix' that is the rigid-body transformation between *CSL* and *CSR* are known, and if there are some features of the scene that can be identified in both images, then the 3D locations of those features can be triangulated. For a thorough review of approaches based on this principle, we refer the interested reader to [2]. While traditional stereo vision approaches, based on projective geometry, have been proven effective under carefully controlled experimental circumstances, they are not ideally suited to most robotics applications. Intrinsic camera parameters and the fundamental matrix may be unknown or time varying, and this requires the frequent repetition of lengthly calibration procedures, wherein known, structured objects are viewed by the stereo vision system, and the required parameters are estimated by numerical algorithms. Assuming a solution to the standard stereo vision problem, applying it to a real physical robot to facilitate object manipulation remains a challenge. In many robotics applications, it is somewhat inconvenient to express the environment with respect to a camera. For example, from a planning and control standpoint, the most logical choice of coordinate system is *CSWorld*, the reference frame at the base of the manipulator, which does not move with respect to the environment. In order to transform coordinates from *CSL* or *CSR* to *CSWorld*, such that we can model objects and control the robot in the same frame of reference, an accurate kinematic model of the robot is required. If such a model is available, it must be carefully calibrated against the actual hardware, and even then its accuracy may be limited by un-modelled nonlinearities.

We show that localising can be learned without explicit knowledge of the camera parameters and the kinematic model.

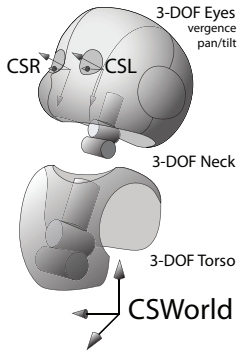

Figure 1: The coordinate frames relevant for object localisation on the *iCub*. Cameras located at the origin of *CSL*/*CSR* are used to express the position of objects with respect to the *CSWorld*.

Table 1: A typical entry from the dataset and the limits used to scale the features and locations for the neural network.

	ImageL		**ImageR**		**Neck**			**Eyes**			**Torso**			**Location**		
	X	Y	X	Y	0	1	2	3	4	5	0	1	2	X	Y	Z
Vector	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	p_0	p_1	p_2
Example	479	411	503	437	-10.0	0.0	0.0	-19.9	-19.9	0.0	-0.1	-9.9	10.1	0.42	0.27	-0.12
max	640	480	640	480	25	25	10	20	15	5	20	20	50	0.66	0.5	0.55
min	0	0	0	0	-25	-25	-10	-20	-15	0	-20	-20	0	0.00	-0.5	-0.15

2. Previous Work

Several different localisation systems have previously been developed for the *iCub*. A popular representation for (stereo) vision research is based on log-polar transformed images. This biologically inspired approach first applies a transformation to the camera images before typical stereo vision algorithms are used. The available module currently supports only a static head, ie. it puts the object position in the *CSL/R* coordinate frame. The 'Cartesian controller module' provides another basic 3D position estimation functionality [3]. This module works well on the simulated robot, however its performance on the hardware platform is weak, this is because of inaccuracies in the robot model and camera parameters. The most accurate, currently available localisation module for the *iCub* exists in the 'stereoVision' module providing centimeter accuracy. Unlike the presented log-polar approach, this module employs the entire *iCub* kinematic model, providing a position estimate in the *CSWorld* coordinate frame. The module requires the previously mentioned 'Cartesian controller' and uses tracking of features to improve the kinematic model of the camera pair by estimating a new fundamental matrix continuously. The precision of all of these approaches depends upon an accurate kinematic model of the iCub. A very accurate model, or estimation of the model, is therefore necessary.

There exists currently no module estimating the kinematics of the iCub, for other robotic systems this has been done: Gloye et al. used visual feedback to learn the model of a holonomic wheeled robot [4] and Bongard et al. used sensory feedback to learn the model of a legged robot [5], but their method uses no high-dimensional sensory information (such as images).

In robot learning, especially imitation learning, various approaches have been investigated to tackle these problems. Sauser & Billard have investigated the problem of reference frame transformations from a neuroscience perspective [6]. They were able to imitate gestures from a teacher on a Hoap-2 humanoid robot with external fixed cameras. Though promising their approach has so far not been extended to systems with non-stationary cameras.

3. Machine Learning Approach

In this paper we investigate two biologically inspired machine learning approaches: a feed-forward artificial neural network (ANN) and genetic programming (GP) approach. These techniques use supervised learning, requiring a dataset including both inputs and outputs (ground truth). More formally, the task is to estimate the position of an object $p \in \mathbb{R}^3$ in the robot's reference frame (*CSWorld*) given an input, also called feature vector, v. Here we defined $v \in \mathbb{R}^{13}$ containing the state of the robot as described by 9 joint encoder values (ie. the 9 controlled DOF) and the observed position in both camera images.

A dataset of reference points (RPs) was collected on the real hardware. A YARP [7] module registering the robot state and storing the camera images was implemented. To obtain the position of an object in the images, an object detection algorithm [8] was used to filter the raw stream from the camera. The hand-measured position of the object in 3D space was then added as the correlating output. The dataset contains 32 RPs on the table, with more than 30 robot poses per point. They lie in a region where the iCub is able to reach with its arms and were distributed in a grid with a spacing of 6 cm.

3.1. Artificial Neural Network (ANN)

An ANN, more precisely a multi-layer perceptron [9] was trained applying a standard error back-propagation [9] method on the dataset collected. The neural network approach requires a pre-processing step, in which the dataset (input vector) is scaled using the limits given in Table 1 to get values in the range $[-1, +1]$. The limits are based on to the maximum image size for the first 4 values, and the joint limits (range of motion in the stochastic controller) of the robot, for the encoder values. The output of the neural network is in the same limited range and needs to be un-scaled.

For training the network the (scaled) dataset was first randomly split into a training (80% of the data) and test set (20%). The test set allows to verify that the results obtained via learning are not over-fitting. Separate networks were trained for the estimation in the X and Y direction. Each network consists of one input layer with dimension 13, a hidden layer, and an output layer. The network uses bias terms and is fully connected. The hidden layer consists of 10 neurons, which use a sigmoidal activation function of the form $\sigma(u) = \frac{1}{1+e^{-u}}$. Finally the output layer is a single neuron representing the estimated position along one axis. The ANNs were trained using PyBrain [10] with a learning rate of 0.35 and a moment of 0.1. The errors reported are the average of 10 runs.

3.2. Genetic Programming

Genetic Programming (GP) is a search technique, most commonly used for symbolic regression and classification tasks. It is inspired by concepts from Darwinian evolution [11]. Herein we use GP to find expressions mapping the inputs to the outputs (3D coordinates). The basic algorithm works as follows: a population is initialised randomly. Each individual represents a tree, encoding a mathematical expression. The nodes encode a function, with the leaf nodes either being an available input or a constant value. For a given set of input values, the output of the expression can be found by recursing from the root node through to the terminal nodes. The individuals are then tested to calculate their 'fitness' (in our case the sum of the mean error). The lower this error, the better the individual is at performing the mapping. In the next step a new population is generated out of the old, by taking pairs of the best performing individuals and performing functions analogous to recombination and mutation. The process of test and generate is repeated until a solution is

Table 2: The mathematical functions available for the genetic programming (GP) method to select from.

add	subtract	multiply	devide
power	sqrt	exp	log
sin	sinh	cos	cosh
tan	tanh	asin	acos
atan2	min	max	abs

found or a certain number of individuals have been evaluated. A comprehensive introduction to genetic programming and its applications can be found in [12].

Herein we use a freely available software 'Eureqa' [13]. It produces compact, human readable expressions from datasets employing the above mentioned techniques. The input values do not have to be scaled in this approach and can remain in the original form. As with the neural network regression, data was shuffled and then split into training and validation sets. The standard settings were used. These including a population of 64 individuals, a crossover rate of 0.5 and a mutation rate of 1.5% and the mean square error of the predication was used as a fitness metric. The generated solution can contain any of mathematical functions in in Table 2.

4. Experiments and Results

To learn the ability to generalise, the techniques need a dataset representing the robot in various configurations and object locations on the table. Our first approach was to place a single object at different known positions on the table and collect data. To simplify the image processing, a red LED was used. The LED was placed at a known position in the grid to mark the reference point, while the *iCub* moved into different poses. For each pose the joint angles and camera images were collected. After collecting data for a number of poses, the LED was moved to another position and the process repeated.

For the table case the problem is simplified as we can assume a constant height (Z axis) estimation. Table 3 compares the position prediction errors of the ANN and GP techniques. It shows that the neural network is performing better during learning, which can also be seen in Fig. 2. Both approaches perform similarly when generalising to unseen data (test set). The ANN training necessitates a longer runtime, as the back-propagation

Table 3: Estimation accuracy on the dataset for both techniques.

	ANN	**GP**
Average Error 2D (cm)	**0.846**	**3.325**
Standard Deviation 2D (cm)	0.504	2.210
Average Error X (cm)	0.540	2.028
Standard Deviation X (cm)	0.445	1.760
Average Error Y (cm)	0.5433	2.210
Standard Deviation Y (cm)	0.4304	1.716

algorithm repeats to update the neural network until the network performance is satisfactory. As described above, two separate networks were trained to predict the coordinates on the X and Y axes independently. This approach was chosen as it allowed for faster learning (i.e. less generations needed to yield the results) and the ability to run the learning in parallel. On average about 1700 epochs were needed per network for its prediction error to converge. After training the network produces estimates with an average accuracy of 0.8 cm, with lower separate errors on the axes (see Table 3). This makes the ANN approach the best performing approach on the dataset.

The GP method, while converging faster than the neural network, performs with a lower average accuracy of 3.3 cm. Although this performance is worse than the ANN, it is still sufficiently accurate to allow for simple reaching and grasping tasks on the *iCub*. However, there are a number of advantages to be considered. The output is in a human-readable form, which can easily and quickly be transferred and tested on the robot. Table 4 shows the evolved equations. An interesting observation is that only one of the camera images is used (features v_0 and v_1). This allows to reduce the (complete) runtime of the estimation as only one images needs to be processed with object detection algorithms before the expression can be evaluated.

During off-line training it appeared that both the ANN and GP approaches provide sufficient accuracy for object manipulation. Both approaches were implemented on the *iCub* to perform real time distance estimation of objects and to allow for further verification. The object position in the images (provided by an object detection filter from a separate iCub vision system [8]) and joint encoder values were provided to both the trained neural network and the GP evolved formulae, to allow easy comparison of the position predictions.

The validation results were obtained using locations on the

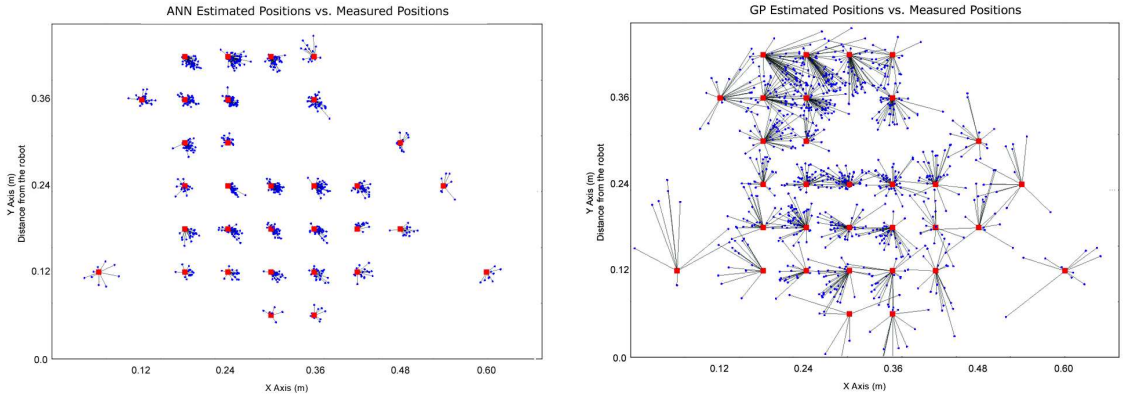

Figure 2: The estimated object position (blue dots) vs. the measured object position (red blocks) for the machine learning approaches: on the left the result obtained from artificial neural networks (ANN), on the right the results using genetic programming (GP).

Table 4: The equations generated using Genetic Programming.

$$x = 17.81 - 0.01906\ v_1 + 0.1527\ v_4 + 0.1378\ v_7 + 0.01108\ v_{10} - 0.0296\ v_{11} - 0.1207\ v_{12}$$
$$y = 1.124224045 + 0.1295920897\ v_{10} + 0.1156011386\ v_8 + 0.01695234993\ v_0$$

Table 5: The relative estimation errors (in cm) when estimating the position using fixed poses of the robot and object locations not in the training nor test set.

		ANN		**GP**		current *iCub*	
dX	dY	estX	estY	estX	estY	estX	estY
0	**+2**	0.10	**1.93**	0.51	**2.28**	0.0	**2.17**
0	**+1**	0.10	**0.78**	0.30	**0.91**	0.05	**1.0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**-1**	0.03	**1.14**	0.31	**1.35**	0.03	**1.07**
0	**-2**	0.11	**2.08**	0.61	**2.40**	0.03	**2.07**
+2	0	**1.70**	0.01	**1.93**	0.57	**2.01**	0.17
+1	0	**0.71**	0.10	**0.81**	0.34	**0.92**	0.11
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
-1	0	**0.99**	0.21	**1.12**	0.11	**1.17**	0.06
-2	0	**1.98**	0.30	**2.24**	0.34	**2.33**	0.1

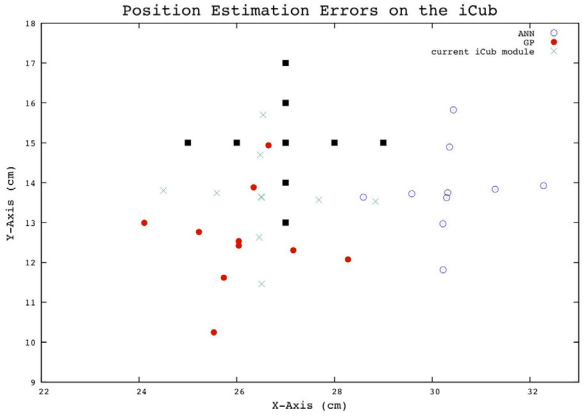

Figure 3: The relative localisation errors on the real hardware. The ground truth is shown in black, the circles represent the learning approaches, ANN (empty circle) and GP (filled). Results from the *iCub* 'stereoVision' module is plotted in green.

table and poses that were not in the original training not test set. It was found that the GP out-performed (average error of 2.7 cm) the ANN (average error of 3.5 cm) on localisation. Both techniques performed slightly worse than a fully calibrated *iCub*'s 'stereoVision' module (1.8 cm accuracy). The performance on the relative error (where the target object was moved by small increments away from a central point) was very high for both implementation with the ANN yielding better results, as can be seen by the values in Table 5 and Fig. 3. The results of the current *iCub* localisation module are added for comparison.

To test these approaches under moving conditions, we scripted the robot to move a given trajectory and recorded the position estimates for an object at a fixed location. The errors were tested for using only head/neck joints, for only using torso and for a combination of both. These all ranged in 2-4 centimetres. The faster the movement the higher the error was, this lead us to believe that it might mainly be an issue of getting the images from both cameras synchronised as much as possible.

We also performed this test with a moving test object, the error though is harder to measure when both objects are moving. In visual verification no big errors were found[1].

5. Conclusions

To estimate the positions of objects placed on a table in front of an iCub robot we compared artificial neural networks (ANN) and genetic programming (GP). No explicit robot model nor a time-consuming stereo camera calibration procedure is needed to learn. Results of locating objects on the table (2D) are sufficient for real world reaching scenarios, with the GP approach performing worse than the ANN method on the training set but generalising better when used on the hardware. The results on the first 3D dataset show that the method can be scaled to perform full 3D estimation. That said a more thorough experimental testing on the iCub will need to be conducted.

The results show that the *iCub* can learn simpler ways to perceive the location of objects than the human engineered methods. Both approaches provide simple and fast methods that can be used in real time on the robot. As the learnt models are 'light weight' they could easily be incorporated into embedded systems and other robotic platforms.

6. References

[1] N. G. Tsagarakis *et al.*, "iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research," *Advanced Robotics*, vol. 21, pp. 1151–1175, Jan. 2007.

[2] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2000.

[3] U. Pattacini, "Modular Cartesian Controllers for Humanoid Robots: Design and Implementation on the iCub," Ph.D. dissertation, RBCS, Italian Institute of Technology, Genova, 2011.

[4] A. Gloye, F. Wiesel, O. Tenchio, and M. Simon, "Reinforcing the Driving Quality of Soccer Playing Robots by Anticipation," *IT - Information Technology*, vol. 47, no. 5, 2005.

[5] J. Bongard and V. Zykov, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[6] E. Sauser and A. Billard, "View sensitive cells as a neural basis for the representation of others in a self-centered frame of reference," in *Int'l. Symposium on Imitation in Animals and Artifacts*, 2005.

[7] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *Advanced Robotic Systems*, vol. 3, 2006.

[8] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, "icVision: A Modular Vision System for Cognitive Robotics Research," in *International Conference on Cognitive Systems*, 2012.

[9] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[10] T. Schaul *et al.*, "PyBrain," *Journal of Machine Learning Research*, 2010.

[11] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[12] R. Poli, W. B. Langdon, and N. F. McPhee, *A field guide to genetic programming*. Published at `http://lulu.com`; Freely available at `http://www.gp-field-guide.org.uk`, 2008.

[13] M. Schmidt and H. Lipson, "Distilling Free-Form Natural Laws from Experimental Data," *Science*, pp. 1–5, Apr. 2009.

[1] A video showing localisation while the iCub and the object is moving can be found at `http://Juxi.net/projects/icVision/`.