Transferring Spatial Perception Between Robots Operating In A Shared Workspace

Jürgen Leitner, Simon Harding, Mikhail Frank, Alexander Förster, and Jürgen Schmidhuber[1]

***Abstract*— We use a Katana robotic arm to teach an iCub humanoid robot how to perceive the location of the objects it sees. To do this, the Katana positions an object within the shared workspace, and tells the iCub where it has placed it. While the iCub moves it observes the object, and a neural network then learns how to relate its pose and visual inputs to the object location. We show that satisfactory results can be obtained for localisation even in scenarios where the kinematic model is imprecise or not available. Furthermore, we demonstrate that this task can be accomplished safely. For this task we extend our collision avoidance software for the iCub to prevent collisions between multiple, independently controlled, heterogeneous robots in the same workspace.**

I. INTRODUCTION

Currently the vast majority of robotic systems are used in industrial applications. In these settings robots have mainly been used as programmable machines, solving automation tasks with with pre-defined, pre-programmed actions in static environments. In recent years however the field has been moving towards extending the use of robots in other areas. A main hurdle is that a predefined, static environment can not be assumed in almost all interesting settings in daily life coexisting and helping humans. Proposed applications range from household tasks, helping in a hospital, to elderly care, grocery shopping, etc.

For a robot to be able to work in these 'unstructured' environments, and extend its applications from industrial to domestic settings, it needs to be able to perceive and understand its surroundings, as the state of the workspace and the objects in it can not be known a priori. The robot therefore has to rely on its sensory feedback to build a model of the scenery. To do so it needs to identify and localise objects autonomously and robustly. This spatial understanding is crucial for motion planning, obstacle avoidance and finally interacting with these environments and the objects therein.

We aim to provide the (low precision) humanoid robot with a technique to estimate positions of objects relative to itself in 3D Cartesian space. Our humanoid platform is the *iCub* robot [1], an open-system robotic platform, providing a 41 degree-of-freedom (DOF) upper-body, comprising two arms, a head and a torso (see Fig. 1). The *iCub* is generally considered an interesting experimental platform for cognitive and sensorimotor development and embodied Artificial Intelligence (AI) [2], and is particularly well suited for learning object manipulation experiments. A high precision robotic arm, in our case a 5 DOF Katana arm by Neuronics [3], is used to position an object in the shared workspace to provide the humanoid with the information to learn from.

[1]The authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), the Università della Svizzera italiana (USI) & the Scuola universitaria professionale della Svizzera italiana (SUPSI) `juxi@idsia.ch`

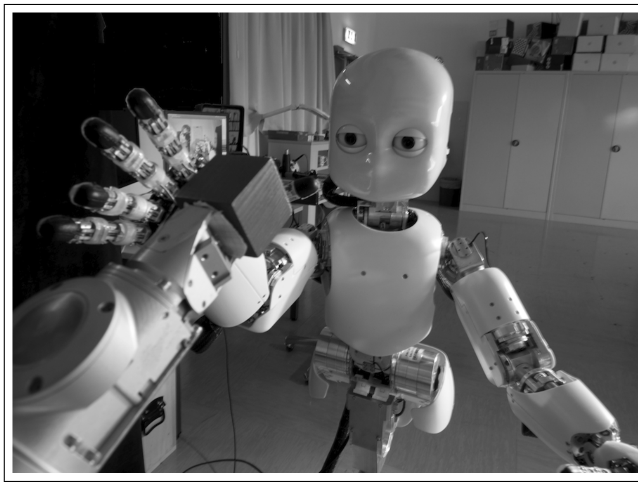

Fig. 1. The *iCub* humanoid robot reaching for a block mounted on the Katana manipulator (foreground) in a shared workspace.

II. SHARING THE WORKSPACE

Multiple robots sharing the same workspace, may it be in cooperation or competition, have been investigated previously. The research is usually focused on mobile robotics for exploration scenarios [4], as area coverage is seen as one of the canonical problems. Cooperating mobile robots have been researched for diverse applications, such as, cleaning [5], indoor surveillance [6], and space exploration [7].

Recently cooperation has also become of interest for humanoid robots and robotic helpers at home. A dual-robot setup was shown by TUM using their PR2 *James* and humanoid robot *Rosie* (similar to DLR's Justin [8]) making pancakes [9]. In their setup the workspace of the two robots is only overlapping during a very little part of the demonstration and therefore collision avoidance between robots can be ignored.

Our aim is for the *iCub* to learn to interact in this environment, act out pre-defined tasks, and adapt to changes in the environment, introduced by the Katana manipulator arm. The setup, in which the robots are facing each other can be seen in Fig. 3.

An obvious challenge in the multi-robot scenario is to prevent the robots from colliding with each other, or indeed, with themselves or the environment. Collisions are likely to lead to damage to either the robot or the environment, leading to time-consuming maintenance.

One approach to tackle this problem of multiple robots interfering and colliding while in the same workspace is to plan ahead of time. Algorithms that take this approach are generally called 'Path Planning' or 'Motion Planning' algorithms, as they plan and validate feasible motions, which can later be passed to the robot as reference trajectories. A vast literature exists on these topics, the interested reader is referred to the recent text book by LaValle [10]. For multi-robot settings a good and thorough introduction to collision avoidance and detection problems has been published by Gill [11].

Previous work [12] investigated collision-free trajectory coordination, in industrial applications, where the trajectories of the (homogenous) robots were predefined and known and coordinating these was the main issue. Multiple robot arms and generation of non-colliding paths while, e.g. passing it from one arm to another were explored by Koga et al. [13].

Alternatively the robot can react to impending collisions as they are predicted. Previous work by Frank et al. [14] introduced *Virtual Skin*, an open-source framework, allowing the monitoring of the state of physical robots in real-time and providing easy to adapt 'reflex' behaviours. These reflex behaviours are invoked when a possible collision is detected and return the robot to a safe pose. We extend this framework to a multi-robot configuration.

Surprisingly little work has been done on multi-robot setups with humanoids. The main focus is towards shared workspaces with humans. This field of human robot interaction (HRI) is growing, the interested reader is referred to a recently published book [15]. In the last years the work on humanoids, while controlling both arms to, e.g. perform bimanual grasping, has become more prominent. Gharbi et al. [16] presented a roadmap approach for path planning using both arms of the DLR Justin robot [8]. It allows to plan object manipulation motions, based on decomposing the system into kinematically independent parts, without the two arms colliding or interfering with each other.

III. OBJECT LOCALISATION

Developing an approach to object localisation that is robust enough to be deployed on a real humanoid robot is necessary to provide the necessary inputs for on-line motion planning and object manipulation tasks. The current state-of-the-art approach to solving the object localisation is using a variety of different time-of-flight sensors, such as, LASER range finders. Recently the use of active vision increased due to the availability of cheap and robust sensors, such as, the Microsoft Kinect. The *iCub* has no such sensors, and therefore the localisation has to rely, similarly to human perception, on stereo vision. As the cameras are mounted in the head of the robot the method for localisation must be able to cope with motion to work **while** the robot is controlling its gaze and upper body for reaching. More formally, the fundamental matrix will vary as a function of pan and vergence of the eyes, and the position and orientation of the stereo camera unit (the head) will vary as a function of the state of the torso and neck.

Stereo Vision describes the extraction of 3D information out of digital images and is similar to the biological process of stereopsis in humans. Its basic principle is the comparison of images taken of the same scene from different viewpoints. To obtain a distance measure the relative displacement of a pixel between the two images is used. In the following discussion, *CSL* and *CSR* refer to the local reference frames of the left and right cameras respectively, while *CSK* is the local reference frame of the Katana manipulator, and *CSWorld* denotes the common reference frame for the workspace, in which we seek to express object locations (see Fig. 2).

The cameras provide two different 2D projections of the same 3D scene. Both the 'intrinsic parameters', specifying each camera's projection from 3D to 2D, as well as, the 'fundamental matrix', that is the rigid-body transformation between *CSL* and *CSR*, need to be known to triangulate the 3D position from the two images. For a thorough review of approaches in stereo vision, we refer the interested reader to the textbook by Hartley & Zisserman [17].

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 inconvenient to express the environment with respect to a camera. From a planning and control standpoint, for example, the most logical choice of coordinate system is *CSWorld*, the reference frame at the base of the *iCub*, which is stationary 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

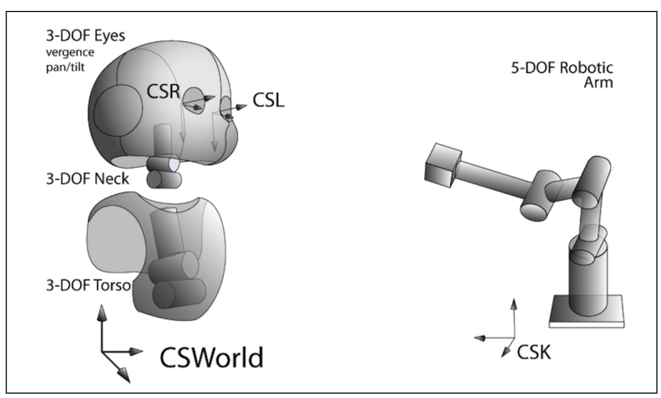

Fig. 2. The object localisation problem, illustrated according to the kinematic model of the iCub humanoid robot, is to process images from cameras located at the origin of *CSL* and *CSR* to express the position of objects with respect to *CSWorld*. *CSK* denotes the reference frame for the Katana manipulator.

frame of reference, an accurate kinematic model of the robot is necessary. In the case of the Katana arm this model is available and therefore the location of the end effector, given in *CSK*, is known with high precision (in millimetre range).

We present a learning technique which does not require, nor tries to *explicitly* build a model of the robot or the cameras. Only one calibration, learning on the collected dataset, is needed, to estimate the location of objects placed in-front of the robot. The herein presented localisation techniques enables the *iCub* to successfully estimate object positions in cartesian space, based on a training set collected with the help of a Katana robotic arm. This is a prerequisite for reaching for an object (and eventually manipulating it).

For the *iCub* platform several different localisation systems have previously been developed, One of these methods is a biologically inspired approach that mimics the retina of the human eye. Camera images are projected by a log-polar transform before typical stereo vision depth estimation algorithms are used to analyse this view. The currently available implementation on the humanoid only supports a static *iCub* head, putting the object position in the *CSR* or *CSL* coordinate frame. A full review of log-polar techniques for robotics applications can be found in [18].

The 'Cartesian controller module', available in the *iCub* software repositories, also provides basic 3D position estimation functionality [19]. This module works well on the simulated iCub, however it is not yet supported on the hardware platform, and therefor does not perform well. One reason for this is its need for an accurate robot model and camera parameters, which necessitates a thorough configuration before using this module on the hardware.

The most accurate currently available localisation module for the iCub exists in the 'stereoVision' module. It provides accuracy in the range of a few centimeters, but with high variance depending on where the object is placed in the camera frame. Unlike the presented log-polar approach, this current, state-of-the art module for 3D localisation[1] works with 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 SIFT [20] and SURF [21] features to improve the kinematic model of the camera pair by estimating a new fundamental matrix, for moving eyes, head and torso. SIFT and SURF analysis is however quite computationally expensive and therefore is not suitable for some embodied applications.

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. To our knowledge no module currently exists to estimate the kinematics of the *iCub*, this is partly due to the openly available CAD models and thorough calibration procedures that need to/should be applied regularly. For other robotic platforms machine learning has been used to estimate the kinematic model, for example, Gloye et al. used visual feedback to learn the model of a holonomic wheeled robot [22] and Bongard et al. used sensory feedback to learn the model of a legged robot [23]. Their method uses no high-dimensional sensory information, as provided by camera images. A genetic programming approach has previously been shown to evolve basic hand-eye coordination on a simple humanoid robot [24].

IV. IMPLEMENTATION OF COLLISION AVOIDANCE

Over the last years we developed *Virtual Skin*, a module for YARP [25] providing collision detection and avoidance behaviours for the *iCub* robot, with an approximate complexity of the computation of $O(n^2m)$, where n is the number of objects in the robot model and m the number of objects in the environment. For this detection the Software Library for Interference Detection (SOLID) [26] is used. It provides highly optimised code for geometric computations (supporting primitives, Minkowski sums, and polyhedra).

YARP is a popular open source robotics middleware, comparable to ROS [27], which was the middleware used for the previously mentioned TUM work on their cooperating robots. It allows to create distributed systems of loosely coupled modules and provides standardised interfaces. As a YARP module, *Virtual Skin* can easily be used with any robot, as long as YARP drivers have been implemented. The iCub drivers are included in the standard version of YARP but for the Katana, we had to add this functionality. The

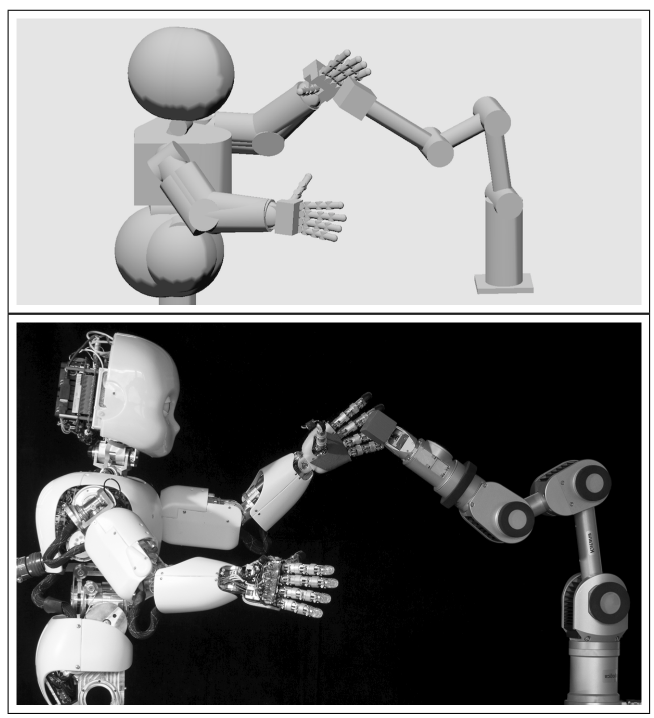

Fig. 3. *iCub* and Katana arm models loaded into the *VirtualSkin* [14] (top) to perform collision detection for both robots while working in a shared workspace. The lower pictures shows the real scene.

[1]The documentation for this code which can be found in the *iCub* source code repository hosted at SourceForge is available at `http://eris.liralab.it/iCub/contrib/dox/html/group__icub__stereoVision.html`

basic driver was developed previously [28] and was adapted to work with the current YARP version. Due to the open-source design this could be done rather modular building on various wrappers for the Katana API. We are confident that our drivers for the Katana manipulator will be added to future YARP releases.

Virtual Skin is intended to enable machine learning research on real robotic systems. It has a similar design philosophy to YARP and also aims for transparency and modularity in its subsystems. This allowed us to extend the behaviour to our multi-robot setup.

Our *Virtual Skin* module consists of three primary components [14]:

1) A kinematic model of the robot and workspace system.
2) A port filter that allows *Virtual Skin* to act as a proxy between an arbitrary control module and the robot.
3) A collision response behaviour.

The system works as follows: A controller, connects to the proxy created at runtime (instead of the direct YARP interfaces of the robot) and can then start controlling the robot. *Virtual Skin* uses the state messages arriving from the real hardware to update the kinematic model and performs real-time collision detection computations based on these. When an impeding collision is detected the robot is stopped and the proxy is closed. Then the defined reflexive collision behaviour is triggered and the controller module is notified. Once the the system is recovered from the dangerous configuration the reflex stops and the controller can continue to use the robot.

To allow for our setup to be using the *Virtual Skin* software, we needed to adapt two out of these three building blocks, namely we needed to:

- add a kinematic model of the Katana arm
- allow to load two models side by side and
- add a collision response for the second robot

In *Virtual Skin* the robot model can be specified via an XML configuration file, using the "Zero Position Displacement Notation" [29], which is significantly less complex and more intuitive than the popular Denavit-Hartenberg convention [30].

Due to the lower complexity of the Katana arm the XML file is short and easy to read. The loaded file together with the loaded iCub model is shown in Fig. 3. The reflex behaviour was a bit tricker to modify, mainly because the parameters used for the two robots had to be tuned to be synchronous. The parameters are defining the length of the history of stored poses is and how much delay is needed between firing consecutive position move commands to the hardware.

V. TRANSFERRING SPATIAL PERCEPTION USING MACHINE LEARNING

To transfer the spatial information we are using a machine learning approach. This supervised learning requires a dataset which includes the inputs and the outputs to be learned (ground truth), i.e. the measured positions in 3D Cartesian space as provided by the Katana.

More formally, the task here is to estimate the position of an object $p \in \mathbb{R}^3$ in the robot's reference frame (*CSWorld* in Fig. 2) given an input, also called feature vector, x. This could, in theory, be solved numerically, in practice though, no precise enough kinematic model is available for the iCub. For other robots, such as Holland's fully anthropomorphic CRONOS [31], no useful kinematic model exists. Therefore we focus on solving the spatial perception problem on the real hardware. In fact a precise learning of spatial perception could in turn be used to build an accurate kinematic model.

The inputs used are the state of the robot, described by 9 encoder values representing the 9 DOF of the iCub's kinematic chain between *CSWorld* and the eyes (see Fig. 2), followed by the observed position of the segmented object in the image plane, defined by the centre of the blob in XY coordinates in both the left and the right image. Additional inputs, based on the *bounding box* of the segmented object, which we specify by the XY coordinates of the upper-left corner, width and height, were used to facilitate learning.

To extract this visual information from the stream of both camera images simultaneously a vision module [32] using the OpenCV [33] library was used to precisely detect the object. The images are segmented and a bounding box over the segmentation is added to provide more inputs to learn from. A typical segmentation is shown in Fig. 4, with the bottom showing the object and its bounding box, defining the values to be added for both the left and right camera image. We can then define $x \in \mathbb{R}^{21}$ as input vector.

The output vector p is taken from the Katana arm, which provides its end effector position with high precision (in mm accuracy) in Katana's reference frame (*CSK* in Fig. 2). These are easily translated into *CSWorld* as these are aligned and only have an offset in one axis.

Three feed-forward artificial neural networks (ANN), one

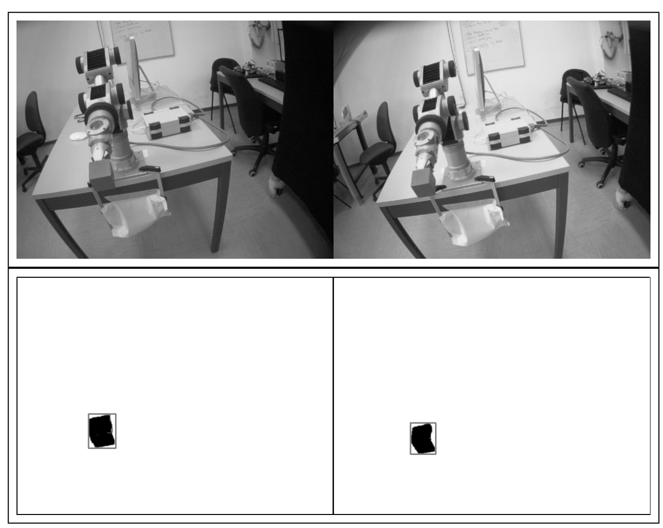

Fig. 4. The upper row shows both camera images (left and right) perceived at the *iCub* (Note: while the camera images are shown in grayscale here the technique presented uses the raw RGB images provided by the iCub YARP interfaces). The lower one shows the results after processing. The object is segmented out and a bounding box is calculated (shown in bright grey).

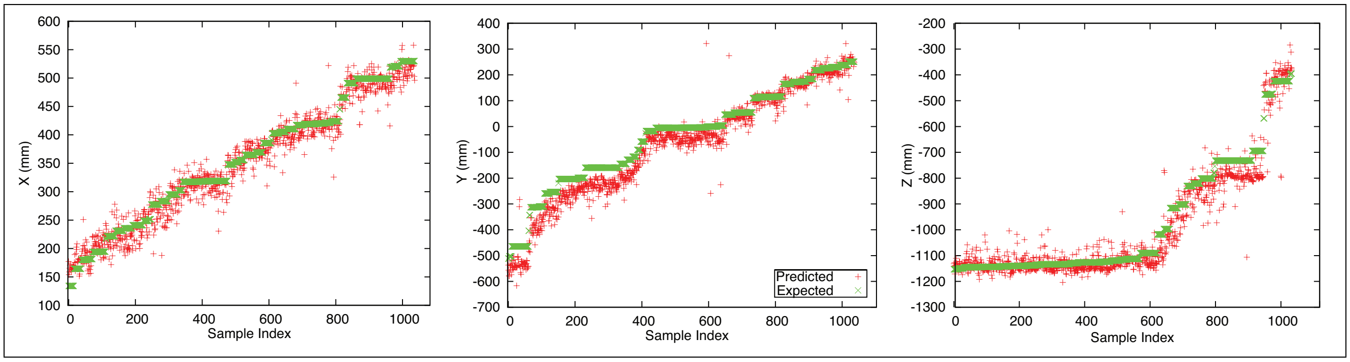

Fig. 5. The prediction errors by the neural networks for the X, Y, and Z axis.

per axis, are used to estimate the position p given the input x. The ANNs are using a multi-layer perceptron architecture and were trained applying a standard error back-propagation [34] method on the dataset collected. The neural network approach requires a pre-processing step, in which the dataset (input vector) is scaled down to provide values in the range $[-1, +1]$. The limits are based on to the maximum image size for the first 12 values, and the joint limits (and range of motion used in the stochastic controller) of the robot, for the robot state (encoder) values.

Each network consists of one input layer (with dimension 21), a hidden layer, and an output layer. The network uses bias terms and is fully connected. The values presented to the input neurons of the neural network are the 21 elements of the feature vector x scaled using the limits. 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 and needs to be rescaled.

Separate networks were trained offline for the estimation in the X, Y and Z direction, each using using a standard error back-propagation algorithm implemented in PyBrain [35]. The errors stem from the difference between the estimation (output neuron) and ground truth (the measured outputs provided by the Katana arm). The learning rate is set to 0.35 and a momentum of 0.1 is used. A training set (80% of the data) was used to train the network, while a test set (the remaining 20%) was applied to verify that the results obtained via learning are not over-fitting.

Fig. 6. This image shows the placement of a cup within the world model of the iCub. The evolved formula is used to calculate the position on the table based on the input images and the current encoder positions. *Note: The cup is placed directly under the arm, due to the parameters of the camera and the different perspective this is hard to see.*

A dataset of reference points was collected in order to learn the 3D positions of objects as a function of the camera images and encoder positions. To collect the dataset both robots moved to randomly selected poses allowing for a random sampling of the configuration space. Once the robots reach their poses, camera images and the encoder positions of the *iCub* are read out and stored together with the position information from the Katana arm to complete the raw entry in the dataset. The *iCub* then continued to another random pose to collect the next datapoint. After iterating through some poses also the Katana was moved to another randomly selected pose. In this first experiment we collected a dataset with 1036 points.

VI. RESULTS

The trained neural networks allow to estimate the position of the object in 3D space, with a high enough accuracy to allow for grasping experiments. The average error on the dataset is for the X-axis 15.9 mm, for the Y-axis 43.1 mm and for the Z-axis 37.3 mm. This is also in the error range of current localisation methods provided for the *iCub*.

Fig. 5 shows three plots, one per axis, visualising the location and prediction error per sample in the dataset. To better depict that multiple samples per location were collected the dataset was sorted by location in this plot. A few prediction outliers can be seen which might result from errors during the collection of data points (e.g. when a data point was collected while a collision was prevented and the robots were reset to a safe pose). Although the system was trained with only one specific object, arbitrary objects can be localised by the deployed system. A video can be found on the authors' website.

We tested the learnt localisation by reaching for the red block held up by the Katana manipulator (Fig. 3), as well as, trying to reach for a cup placed on the table, as shown in Fig. 6.

VII. CONCLUSIONS

Allowing our *iCub* to interact with other robots opens up a range of potential research avenues. In this paper we have demonstrated how our *iCub* can learn from a teaching robot. The accuracy of the learnt localisation is sufficient for our applications. We put emphasise on learning positions in the (limited) area our robot is able to reach. There the location is precise enough to allow the *iCub* to grasp and manipulate objects.

A key component in the success of this work was the development and extension of the *Virtual Skin* software. It allows our two robots to safely work in the same workspace. Without this, the *iCub* would only be able to learn about objects that were far away, or in a constrained region – such as on the surface of a table [36].

In future work we will investigate improving the training by teaching the *iCub* with different object types. The accuracy of the localisation may be improved by having object-specific localisation approaches. Another goal is to extend this work to detection of the object's orientation. In fact we already collect 6 inputs from vision per image (centre of blob, as well as, the location and size of the bounding box), which allow an estimation of the objects orientation, if the detected object's geometry is known.

REFERENCES

[1] N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell, "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] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, "The iCub humanoid robot: An open-systems platform for research in cognitive development," *Neural Networks*, vol. 23, no. 8-9, pp. 1125–1134, Oct. 2010.

[3] Neuronics AG, "Katana user manual and technical description."

[4] W. Burgard, M. Moors, C. Stachniss, and F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[5] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *ACM SIGCHI/SIGART Conference on Human-Robot-Interaction*, 2006, pp. 258–265.

[6] M. Moors, T. Rohling, and D. Schulz, "A probabilistic approach to coordinated multi-robot indoor surveillance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 3447–3452.

[7] J. Leitner, "Multi-robot formations for area coverage in space applications," Master's thesis, Helsinki University of Technology (TKK), Helsinki, Finland, 2009.

[8] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, *et al.*, "A humanoid two-arm system for dexterous manipulation," in *IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 276–283.

[9] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, and M. Tenorth, "Robotic roommates making pancakes," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, October, 26–28 2011.

[10] S. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[11] M. Gill and A. Zomaya, *Obstacle Avoidance in Multi-Robot Systems: Experiments in Parallel Genetic Algorithms*. World Scientific Pub Co Inc, 1998, vol. 20.

[12] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 2002, pp. 624–631.

[13] Y. Koga and J. Latombe, "On multi-arm manipulation planning," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 945–952.

[14] M. Frank, A. Förster, and J. Schmidhuber, "Reflexive Collision Response with Virtual Skin - Roadmap Planning Meets Reinforcement Learning," in *International Conference on Agents and Artificial Intelligence*, 2012, pp. 642–651.

[15] K. Dautenhahn and J. Saunders, *New Frontiers in Human-Robot Interaction*. John Benjamins Publishing Company, 2011.

[16] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multiarm systems path planning," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 2471–2476.

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

[18] V. J. Traver and A. Bernardino, "A review of log-polar imaging for visual perception in robotics," *Robotics and Autonomous Systems*, vol. 58, pp. 378–398, 2010.

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

[20] D. Lowe, "Object Recognition from Local Scale-Invariant Features," in *Proceedings of the International Conference on Computer Vision*. IEEE Computer Society, Sept. 1999.

[21] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision – ECCV 2006*, vol. 3951, pp. 404–417, 2006.

[22] 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.

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

[24] W. B. Langdon and P. Nordin, "Evolving Hand-Eye Coordination for a Humanoid Robot with Machine Code Genetic Programming," in *European Conference on Genetic Programming (EuroGP)*. Springer-Verlag, 2001.

[25] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *International Journal of Advanced Robotics Systems*, vol. 3, no. 1, 2006.

[26] G. van den Bergen, *Collision detection in interactive 3D environments*. Morgan Kaufmann, 2004.

[27] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[28] G. Kaufmann, "A flexible and safe environment for robotic experiments : a sandbox and testbed for experiments intended for the humanoid robot icub," Master's thesis, Università della Svizzera italiana (USI), 2010.

[29] K. Gupta, "Kinematic analysis of manipulators using the zero reference position description," *The International Journal of Robotics Research*, vol. 5, no. 2, p. 5, 1986.

[30] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices." *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[31] O. Holland and R. Knight, "The anthropomimetic principle," in *AISB Symposium on Biologically Inspired Robotics*, 2006.

[32] 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 (CogSys)*, 2012.

[33] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

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

[35] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," *Journal of Machine Learning Research*, 2010.

[36] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, "Towards spatial perception: Learning to locate objects from vision," in *Postgraduate Conference on Robotics and Development of Cognition (RobotDoC)*, 2012.