Autonomous Learning Of Robust Visual Object Detection And Identification On A Humanoid

Jürgen Leitner*†, Pramod Chandrashekhariah*‡, Simon Harding*§,
Mikhail Frank†, Gabriele Spina††, Alexander Förster†, Jochen Triesch‡, Jürgen Schmidhuber†
†Dalle Molle Institute for Artificial Intelligence (IDSIA)/SUPSI/USI, Switzerland
‡Frankfurt Institute for Advanced Studies (FIAS), Germany
§Machine Intelligence Ltd, United Kingdom ††ACTLab, TU Eindhoven, The Netherlands
*Joint first authorship.

***Abstract*—In this work we introduce a technique for a humanoid robot to autonomously learn the representations of objects within its visual environment. Our approach involves an attention mechanism in association with feature based segmentation that explores the environment and provides object samples for training. These samples are learned for further object identification using Cartesian Genetic Programming (CGP). The learned identification is able to provide robust and fast segmentation of the objects, without using features. We showcase our system and its performance on the iCub humanoid robot.**

I. Introduction

Animals and humans are exposed to vast amounts of visual inputs in their daily lives. From this complex visual environment they seem to effortlessly extract relevant knowledge about the world, and learn about the objects without any supervision and recognise them later on.

Robots so far have not been able to perform as well with large amount of visual input data. Most of the popular algorithms used for learning about objects involve an explicit training phase where a supervised learning algorithm is applied to a large set of hand-labeled training samples [1]. In recent years the field of robotics is moving towards application areas such as cleaning tasks, grocery shopping and elderly care. These involve the robots working together, helping and coexisting with humans in daily life. To decide and act in these unstructured environments, the robot needs to perceive, as it is not possible to provide the robot with all the information it needs a priori. Therefore it is of great importance to devise autonomous vision systems, and in particular active robot vision systems that can perform the learning task effectively. Only recently have researchers started addressing how a robot can learn to recognise objects in a largely autonomous fashion [2], how learning can be made fully online [3], and how the need for a human teacher can be minimised [4].

In this work, we show a technique for a humanoid robot to learn robust object identification based on image segmentation. We developed a visual system that actively explores an unfamiliar environment and autonomously learn a visual representation for objects in the scene. We combine different techniques to perform this robot learning task. An attention system drives the scene exploration using salient points provided by a neuromorphic model (Section II). This is coupled with a feature-based stereo segmentation scheme to perform a rough segmentation of objects (see Section III). Then these images are used to train robust, efficient and unique object identifiers using a machine learning approach called Cartesian Genetic Programming (CGP) (Section IV). After learning, these identifiers function *without* the need of feature detection, yet providing quick and robust object detection capabilities. While the main focus of the paper is to learn these unique identifiers using CGP, the autonomous scene exploration is a novel integration of components in itself. We then demonstrate the integration of these approaches on a humanoid robot.

II. Scene exploration for autonomous learning

Our aim is to let the robot explore the scene and learn to detect and identify objects in an autonomous fashion. Our experimental platform is the *iCub* humanoid robot [5], 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 [6] research, with a focus on object manipulation. To allow for manipulation, the object of interest has to first be detected and located using the available sensors. The robot has to rely, on a visual system consisting of two cameras in the head, analogous to human visual perception.

A. Bottom-up attention mechanism

To identify elements of the visual input that are interesting a neuromorphic model developed by Itti *et al.* [7] is employed. It aims to detect those points in the scene that are likely to attract the attention of a human observer.

Different visual features, such as, colour, intensity, motion etc., contribute to finding these points in the visual input, representing a stimulus worth paying attention to. In the model used the conspicuity of each image are integrated into one single 2D map of intensities, called the saliency map.

1) Attention selection: We make use of stereo information to select the most salient point in the scene. Saliency maps, calculated from images recorded at *both* eyes, are used by the decision module to select the single most salient point in

the scene (Fig. 2A). The most salient point, and therefore the dominant eye, can be in either the left or the right eye.

Its corresponding location in the other eye is derived using feature matching (explained in detail in Section III). As our robot can not only control its gaze, but also its neck (3 DOF) and hip (3 DOF), the interest point locations in 2D image plane coordinates are not unique, therefore 3D location is computed based on the robot's current pose. This interest point in the scene is stored and further used to control the robot's posture, gaze and vergence to focus on the salient object before segmentation.

2) Inhibition of return: When there are no other variations in the visual characteristics of the scene it is very likely that the attention mechanism continues to select the same location as the most salient point. To avoid this the intensity of the saliency map is suppressed at the current winner location by subtracting a Gaussian kernel. This is only a temporary inhabitation and the intensity can cumulatively over time raise up again. This allows the system to shift attention to the next most salient location, if one exists.

B. Top-down modification of attention

In order to avoid constant switching between the two most salient locations, we also introduce a top-down inhibition of recently visited locations. In our experiments, a list of the 5 most recently visited locations is maintained and close-by points are inhibited for the next gaze shift (Fig. 2B), to guide the attention towards unseen locations.

III. Image Segmentation Using Features

Image segmentation is the most common approach to perform object detection in computer vision. It is the process of separating foreground from background, or one object class from another. There is a vast body of work on all aspects of image processing and segmenting, using both classical and machine learning approaches [8]. However, image segmentation is especially challenging when the objects are static, occluded and the background model is not known. In this work, we use a feature based approach to overcome some of these challenges.

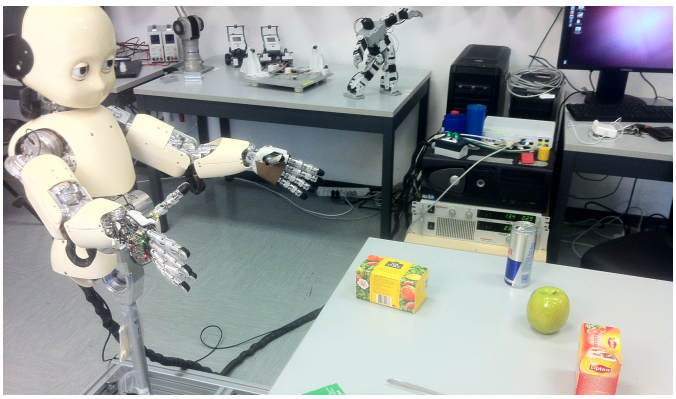

Fig. 1. The *iCub* robot and the objects to be learned placed on the table.

Features are a brief yet comprehensive representation of the image or scene that possess the property of being interesting and repeatable. Local image features, also called local descriptors, are entities that capture the information of region around identified points which are called 'interest points'. Local image features have proven to be resistant to occlusions, local deformations of the objects, variation in illumination conditions, and background clutter [9], [10]. Features are calculated at interest points in the image that are stable under local and global perturbations in the image domain, and perspective transformations such as scale. We use the high-quality and quick FAST corner detection algorithm [11] herein, although other approaches can be substituted (e.g., SIFT [12] and SURF [13] lead to comparable results).

A. Feature Extraction

We use Gabor wavelets for calculating the features which are convolution kernels having the shape of plane waves restricted by a Gaussian envelope function [14]. The choice is also motivated by the fact that they have a similar shape as the receptive field of simple cells found in the visual cortex of vertebrate animals [15], [16]. At each interest point we extract a 40-dimensional feature vector, which we refer to as a Gabor-jet, resulting from filtering the image with Gabor wavelets of 5 scales and 8 orientations. A Gabor-jet is obtained by convolving a small patch of an image at a given location $\mathbf{x}$ with Gabor wavelet with a specific orientation and scale.

B. Feature Matching

To segment a potential object from the background we make use of stereo information provided by the robot's two cameras. We find correspondences between features detected in the left and right images by exhaustively comparing Gabor-jets extracted at these points. We use normalised Euclidean inner product to find the similarity between two jets $\mathcal{J}_1$ and $\mathcal{J}_2$:

$$S(\mathcal{J}_1, \mathcal{J}_2) = \frac{\mathcal{J}_1^T \mathcal{J}_2}{||\mathcal{J}_1|| \; ||\mathcal{J}_2||}$$

Each interest point in the left image is associated with the best matching interest point in the right image if the similarity S between the two jets is above a preset threshold (0.95 herein). Fig. 3 shows the result of such a matching.

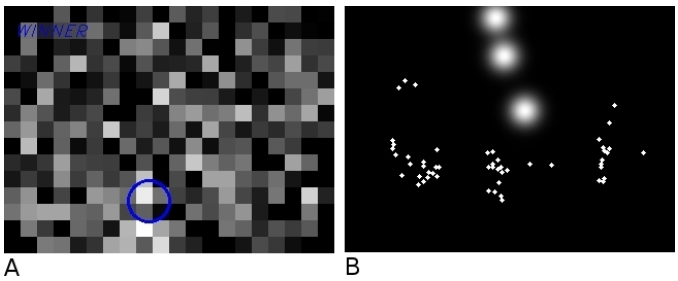

Fig. 2. (A) Saliency map with the winning location highlighted (blue circle). (B) Salient and visited locations (white blobs) projected into 2D.

C. Stereo segmentation

We cluster the matched interest points from the left image (that is used for learning) into different groups according to their image location and disparity. We use a greedy clustering scheme that starts with a single interest point and adds new ones if their x-position, y-position, and disparity are all within a 5 pixel threshold of any existing cluster member. Fig. 4 shows, in two examples, how the object at the centre of gaze is segmented (white dots) from other objects which are at a similar depth but different spatial location or at a close-by spatial location but different depth. Cluster of dots of different colours belong to different objects.

To refine the segmentation, the locations of features are segregated into on-object and outside-object categories (white vs other colours). A minimum bounding rectangle enclosing on-object locations with an additional 20 pixel margin on every side is considered for initialising the segmentation. The outside-object locations on the image as well as all the image pixels that are lying outside the bounding area are considered as outside-object pixels (black colour in Fig. 4B). Next, on-object locations are connected by straight lines (to form the convex hull, shown in white) representing the initial segmentation of the object. All other pixels inside the bounding box that are marked grey indicate that their status is unknown. This pre-segmented area is sent to a non-parametric, recursive watershed segmentation algorithm [17]. Fig. 4C shows the segmentation result for two examples.

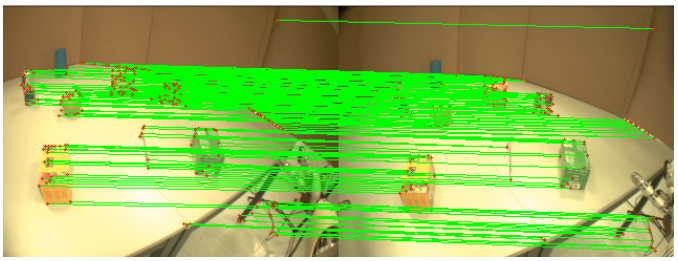

Fig. 3. Feature matching between left and right camera images.

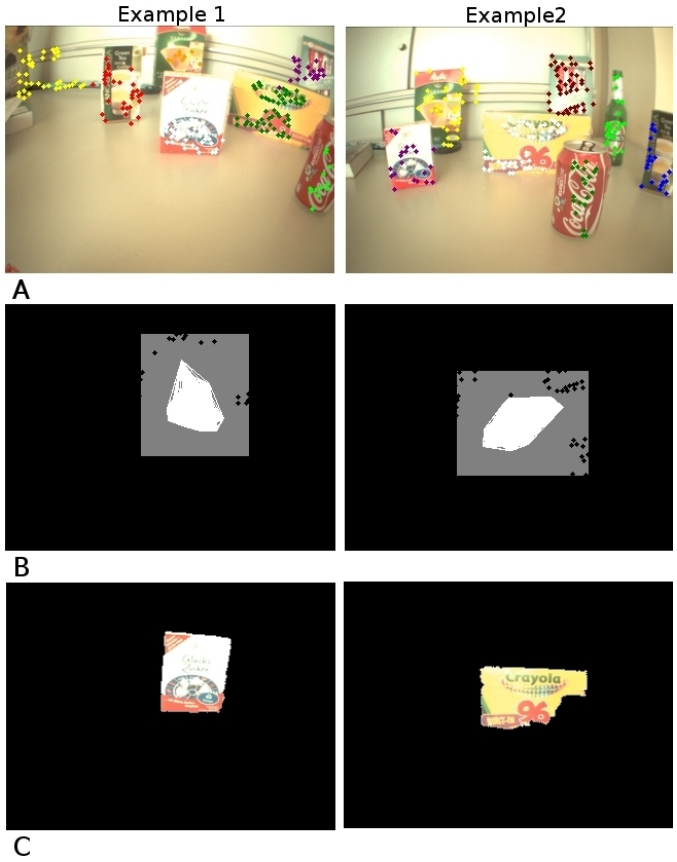

Fig. 4. (A) The features, for which a matching exists, are clustered. (B) Segmentation Initialisation. (C) Segmented Object.

IV. Learning Object Identification Using Cartesian Genetic Programming

Genetic Programming (GP) is a widely used machine learning and search technique inspired by concepts from Darwinian evolution [18], [19].

We use a version of GP called Cartesian Genetic Programming (CGP) to learn programs that can identify objects robustly and execute in real-time. With the evolved programs performing simultaneous segmentation and object recognition. A comprehensive review of CGP can be found in [20]. In this work, we use a version called 'CGP for Image Processing' (CGP-IP) [21]. Previously CGP-IP has been applied to problems such as medical imaging, robot vision, terrain classification [22] and image filters.

In CGP-IP genotypes encode programs in partially connected feed forward graphs [23], [20]. For each node in the genome there is a vertex, represented by a function, and a description of the nodes from where the incoming edges are attached. Other parameters are also encoded at the genotype of each node, and these are described below. Fig. 5 shows an exemplary graph. Executing the CGP genotype is a straightforward process. First, the active nodes are identified to perform the genotype-phenotype mapping. This is done recursively, starting at the output node, and following the connections used to provide the inputs for that node. The phenotype is then executed to obtain a fitness score, which is related to the segmentation difference between the output of the CGP and the segmentation provided by the feature-based segmentation from the previous section.

As previously noted, CGP-IP needs a number of additional parameters encoded in *each node*, compared to classical CGP.

TABLE I
The additional parameters encoded at each node in CGP-IP.

Parameter	**Type**	**Range**
Function	Int	# of functions
Connection 0	Int	# of nodes and inputs
Connection 1	Int	# of nodes and inputs
Parameter 0	Real	no limitation
Parameter 1	Int	$[-16, +16]$
Parameter 2	Int	$[-16, +16]$
Gabor Filter Frequency	Int	$[0, 16]$
Gabor Filter Orientation	Int	$[-8, +8]$

These (listed in Table I) are needed because often the functions used require additional parameters, with specific requirements as to their type and range.

The function set for CGP-IP consists of high level image processing operations, such as dilate, edge detection etc. The primitive operators work on entire images, i.e. `add` will produce a new image adding the values of corresponding pixels from two input images. In this work, the function set comprises a large subset of the OpenCV image processing library. Over 60 functions are available, and a complete list can be found in [21]. This method enables inserting of domain knowledge into evolved programs, and also improves evolvability as operations that are useful can be used directly instead of re-evolving the same functionality. By using OpenCV we can also be confident about using high quality, high speed software. In CGP-IP, individuals are evaluated at the rate of 100s per second on a single core. This makes it both efficient to evolve with, but also means that the evolved programs will run quickly when deployed.

The output of a CGP-IP individual is based on the image at the final node in the genotype. In addition to the graph representing the program, the genotype also includes a real number value that is used for thresholding the output. This binary image is then evaluated by the fitness function to measure the performance of the individual. The fitness function is describe in Section IV-A.

CGP-IP does not require many *global* parameters to be set, with the main parameters being:

- Graph length (i.e. nodes in the genotype), is set to 50.
- Genes in the graph are mutated with a rate of 10% when an offspring is generated.[1]
- Number of islands depend on the available computing resources. We used 8 islands here.[2]
- The number of individuals per island is 5 for a typical 1+4 evolutionary strategy.
- Synchronisation between islands is every 10 generations.

All parameters are kept constant throughout our experiments. It is important to note that these parameters have not been optimised other than by casual experimentation. It may be possible to improve the performance by careful selection.

[1]The threshold parameter has a lower mutation rate of 1%.
[2]CGP-IP has been tested successfully with 1 to 24 islands.

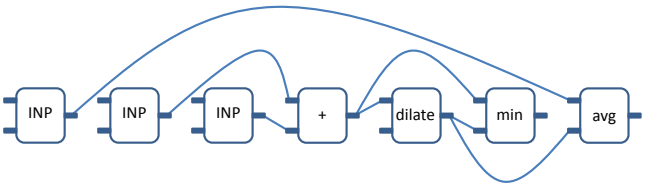

Fig. 5. In this example illustration of a CGP-IP genotype, the first three nodes obtain the components from the current test image (e.g. grey scale version, red and green channels). The fourth node adds the green and red images together. This is then dilated by the fifth node. The sixth node is not referenced by any node connected to the output (i.e. it is neutral), and is therefore ignored. The final node takes the average of the fifth node and the grey scale input.

A. Fitness Function

For this application the thresholded output image of an individual is compared to a target image using the Matthews Correlation Coefficient (MCC) [24], which has previously been observed to be useful when solving classification problems using CGP [25]. The MCC is calculated based on the count of the true positives, false positives, true negatives, and false negatives. A coefficient of 0 indicates that the classifier is working no better than chance. Perfect classification yields a score of 1, -1 indicates the classifier, though perfect, has inverted the output classes. Therefore, an individual's fitness

$$f = 1 - |MCC|$$

with values closer to 0 being more fit.

The MCC is insensitive to differences in class size, making it a nice choice for many binary classification tasks. The confusion matrix also allows for easy calculation of the classification rate as a percentage of correctly classified pixels.

V. Experimental Results

A. Training For Robust Image Segmentation

In our experiments the training set consists of between 2 and 10 images, in which the object of interest has been correctly segmented. Fig. 6 shows some examples of these pre-segemented images provided by the feature-based approach. The selection of images for this training set is of importance for the capabilities of the found solution. If chosen correctly, the resulting programs are very robust to changing light conditions, as shown in Fig. 7.

On a single core of a modern desktop PC, the evolved programs for objects like tea boxes and kids toys, run within 2-30ms. As these are largely linear in form, the memory requirements are low. Speed and simplicity of processing is important in many embedded systems, making this approach suitable for implementation in constrained computing environments.

Fig. 6. A set of samples provided by the features-based segmentation as inputs to the CGP-IP learning. The convex-hull of the features detected has high variance and covers, in most cases, only parts of the object.

Fig. 7. The detection of a tea box in changing lighting condition performed by a learned filter. The binary output of the filter is used as red overlay.

```
icImage GreenTeaBoxDetector::runFilter() {
    icImage node0 = InputImages[6];
    icImage node1 = InputImages[1];
    icImage node2 = node0.absdiff(node1);
    icImage node5 = node2.SmoothBilataral(11);
    icImage node12 = InputImages[0];
    icImage node16 = node12.Sqrt();
    icImage node33 = node16.erode(6);
    icImage node34 = node33.log();
    icImage node36 = node34.min(node5);
    icImage node49 = node36.Normalize();
    icImage out = node49.threshold(230.7218f);
    return out;
}
```

Listing 1. The generated code from CGP-IP for detecting the tea box. `icImage` is a wrapper class to allow portability within our framework [26].

B. Combining Features and CGP Approach

To create a robust object identification based on image segmentation we combine the two techniques discussed in Sections III and IV. Extracting features that allow for robust object detection is not an easy task.

Fig. 8 shows the same tea box and the FAST features extracted in two frames taken roughly 5 seconds apart. To decide whether this is the same object is hard as the features detected are different. To overcome this we use the features detected to build a training set for our CGP-IP approach. CGP-IP quickly learns how to segment the green tea box (1200 evaluations). The same evolved program detects the object in both images (Fig. 9) allowing for unique identification.

CGP-IP directly generates human readable C++ programs. The resulting C++ code, shown in Listing 1, performs the detection on the hardware within 30ms, well below the image refresh rate of the cameras (10-20fps).

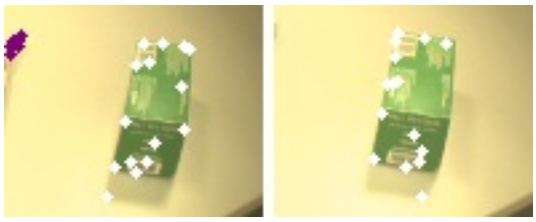

Fig. 8. The FAST features (in white) found of an object in two images.

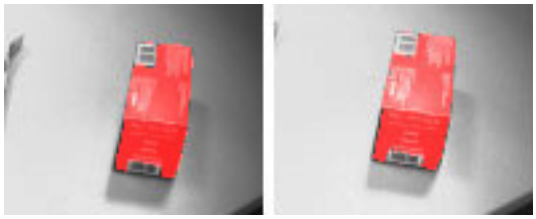

Fig. 9. The tea box as detected with a learned filter using CGP. The binary output of the filter is used as a red overlay.

C. Autonomously segmenting

We showed that the feature based segmentation can be used to train unique object detection programs. In a next step we want the robot to autonomously explore the scene and create one such program for each object detected. The robot uses the attention mechanism described in Section II to control the movements of the hip, neck and eyes to explore the scene. It looks at objects, one after the other, while the feature-based segmentation is run in the background. These segmented images are stored and used to train separate identifiers for each object using CGP-IP. Once the training is complete, the learned identifiers are tested by placing the objects in different locations on the table. The tests included different poses and varying lighting conditions. The learned programs can also be further used to build a model of the environment, as shown in Fig. 10, by integrating this system with existing software [26], [27], [28].

VI. Conclusion

We have developed and demonstrated an autonomous learning and robust detection mechanism of objects on a humanoid robot. Our mechanism has coupled a stereo based image segmentation mechanism with a machine learning approach called Cartesian Genetic Programming (CGP) for learning object representations.

A neuromorphic scene exploration mechanism is used to find the objects in the scene. Upon gazing at a certain object images are created using a feature-based approach, roughly separating the object from the background and other objects. These are then in turn used to train identifiers using CGP. The CGP programs, once trained, are shown to robustly perform object segmentation without the use of features. As they are trained for one specific object (type), they simultaneously segment and identify the objects uniquely in changing environments. These efficient filters perform the task in real-time on the real robot.

The combination of a stereo vision, feature-based segmentation with a robust, supervised filter learning using CGP creates a novel, autonomous and unsupervised learning framework for

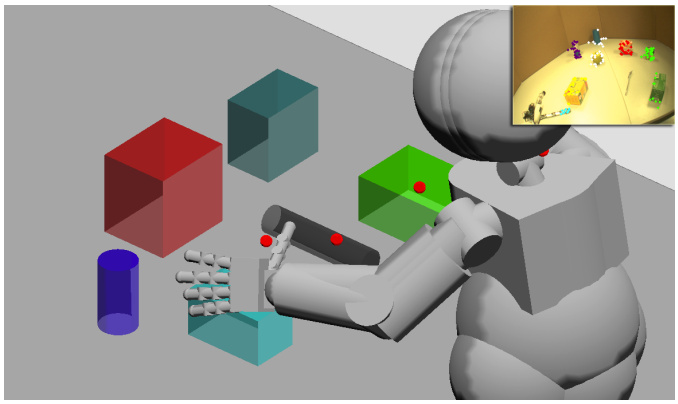

Fig. 10. World model generated in the MoBeE framework [27] using the output from the learned filters. The inset shows the left camera image.

robust object detection. This entire system can serve as an inspiration and guiding mechanism for humanoids that learn their environments and detect the objects later on with a high accuracy. In our work we use this system to autonomously build a model of the robot's surroundings.

VII. Acknowledgment

This research has partly been funded by the European Community's Seventh Framework Programme FP7/2007– 2013, under grant agreement No FP7-ICT-IP-231722 "IM-CLeVeR" and by the German Federal Ministry of Education and Research (BMBF) through project "Bernstein Fokus: Neurotechnologie Frankfurt, FKZ 01GQ0840".

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, p. 511, 2001.

[2] H. Kim, E. Murphy-Chutorian, and J. Triesch, "Semi-autonomous learning of objects," *Computer Vision & Pattern Recognition Workshop*, 2006.

[3] S. Kirstein, H. Wersing, and E. Korner, "A biologically motivated visual memory architecture for online learning of objects," *Neural Networks*, Nov. 2007.

[4] Y. Gatsoulis, C. Burbridge, and T. M. McGinnity, "Online unsupervised cumulative learning for life-long robot operation," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011.

[5] 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.

[6] G. Metta *et al.*, "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.

[7] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[9] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[10] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," in *ECCV*, 2002, pp. 113–130.

[11] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 1, pp. 105–119, 2010.

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

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

[14] L. Wiskott, J. Fellous, N. Krüger, and C. v.d. Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 775–779, 1997.

[15] J. P. Jones and L. A. Palmer, "An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex," *J. of Neurophysiology*, vol. 58, pp. 1233—1258, 1987.

[16] D. A. Pollen and S. F. Ronner, "Phase relationship between adjacent simple cells in the visual cortex," *Science*, vol. 212, p. 1409, 1981.

[17] S. Wegner, T. Harms, J. H. Builtjes, H. Oswald, and E. Fleck, "The watershed transformation for multiresolution image segmentation," in *International Conference on Image Analysis and Processing*, 1995.

[18] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *International Conference on Genetic Algorithms and Their Applications*, 1985.

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

[20] J. F. Miller, Ed., *Cartesian Genetic Programming*, ser. Natural Computing Series. Springer, 2011.

[21] S. Harding, J. Leitner, and J. Schmidhuber, "Cartesian genetic programming for image processing," in *Genetic Programming Theory and Practice X (in press)*. Springer, 2012.

[22] J. Leitner, S. Harding, A. Förster, and J. Schmidhuber, "Mars Terrain Image Classification using Cartesian Genetic Programming." in *International Symposium on AI, Robotics and Automation in Space (i-SAIRAS)*.

[23] J. F. Miller, "An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2, Orlando, FL, 1999, pp. 1135–1142.

[24] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme." *Biochimica et Biophysica Acta*, vol. 405, no. 2, pp. 442–451, 1975.

[25] S. Harding, V. Graziano, J. Leitner, and J. Schmidhuber, "Mt-cgp: Mixed type cartesian genetic programming," in *Genetic and Evolutionary Computation Conference*, Philadelphia, PA, 2012.

[26] 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.

[27] M. Frank, J. Leitner, M. Stollenga, G. Kaufmann, S. Harding, A. Förster, and J. Schmidhuber, "The modular behavioral environment for humanoids and other robots (mobee)." in *9th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*., July 2012.

[28] J. Leitner, S. Harding, M. Frank, A. Förster, and J. Schmidhuber, "Transferring spatial perception between robots operating in a shared workspace." IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.