FP6-IST-002020

COGNIRON

The Cognitive Robot Companion

Integrated Project

Information Society Technologies Priority

D5.2005-2
Joint RA5 deliverable:
Models of Objects

Due date of deliverable: 31/12/2005
Actual submission date: 25/01/2006

Start date of project: January 1st, 2004 **Duration**: 48 months

Contributing partners: IPA, UniKarl, LAAS, UvA
Revision: final
Dissemination level: PU

Executive Summary

A robot companion needs to acquire and maintain models of objects in order to recognize and to use them. These models need to be learned during operation since it is hard to predict what objects the robot will be confronted in its operation environment. In this deliverable we describe important research work that will finally contribute to cognitive behavior of robot companions in that algorithms allow for learning object knowledge at different layers of representation. We focus on interactive learning, autonomous learning and different kinds of models that share the property of high plasticity in order to cover a potentially unlimited range of objects. Contributions of this work relate to all three experiments (see below). Learning and applying object knowledge for e.g. recognition or object manipulation is one the most important key competences of a robot companion. There are a lot of unanswered questions to be solved since in the "pure" computer vision community the problem is usually studied in an isolated manner ignoring the potentials of a rich sensory-motor embodiment and interaction with a human supervisor. We believe that these aspects are necessary to the design of artificial cognition.

Table of Content

Executive Summary 1
Table of Content 2
Role of "Object Models " in Cogniron 3
Relation to the Key Experiments 3
1 Learning Object Models 4
1.1 Introduction 4
1.2 IPA: Trainable appearance-based detection and model-based pose estimation (appearance layer) 5
1.3: LAAS: Learning sensory-motor relationships from observation (appearance and interaction layer) 6
1.4: UKA, IPA & LAAS: Generation and refinement of object models (interaction layer) 7
2 Future work 8
3 References 9
Appendix 9

Role of "Object Models " in Cogniron

In Cogniron the work on object models emerges as being one important Cogniron contributions through various partner activities and interactions. This can be confirmed by the strong need for contributions of object modelling in the three key experiments (see below) and several links to other RAs. An obvious link refers to RA4, which contributes task learning. There is also a link to the field of visual navigation which is also part of RA5. Clearly, the existence of static objects in the near vicinity of the robot can help in deciding where the robot is. Another relationship links to RA1 and RA2 through the problem of resolving objects references as a part of the dialogue functions involve in human robot interaction. Another link to RA2 is currently in preparation in form of an object tracking function that utilizes the human tracker (CF-TBP). Additional knowledge about objects is also used for the activity recognition (CF-ACT). The main link to RA3 is through human friendly passing of objects. Furthermore, there is a relation to RA6 in terms of human friendly manipulation of object (CF-MHP) and the placement of object modelling contributions in the Cogniron architecture.

Relation to the Key Experiments

In KE1 there is a function for resolving object references in which parts of this work will be integrated. In KE2 there are two parts addressing the learning of object knowledge: learning object-action relationships and highly flexible mesh representation to model 3D data. In KE3 object representations rather than task descriptions are learned.

1 Learning Object Models

1.1 Introduction

A robot companion needs to acquire and maintain models of objects in order to recognize and use them. These models need to be learned during operation since it is hard to predict what objects the robot will be confronted with in its operation environment. Knowledge can be learned either through interaction with the human or autonomously by interacting with the object.
To structure research work in Cogniron related to object models we defined a *unified object model* and a general *learning cycle* [1] that are shown in Figure 1.

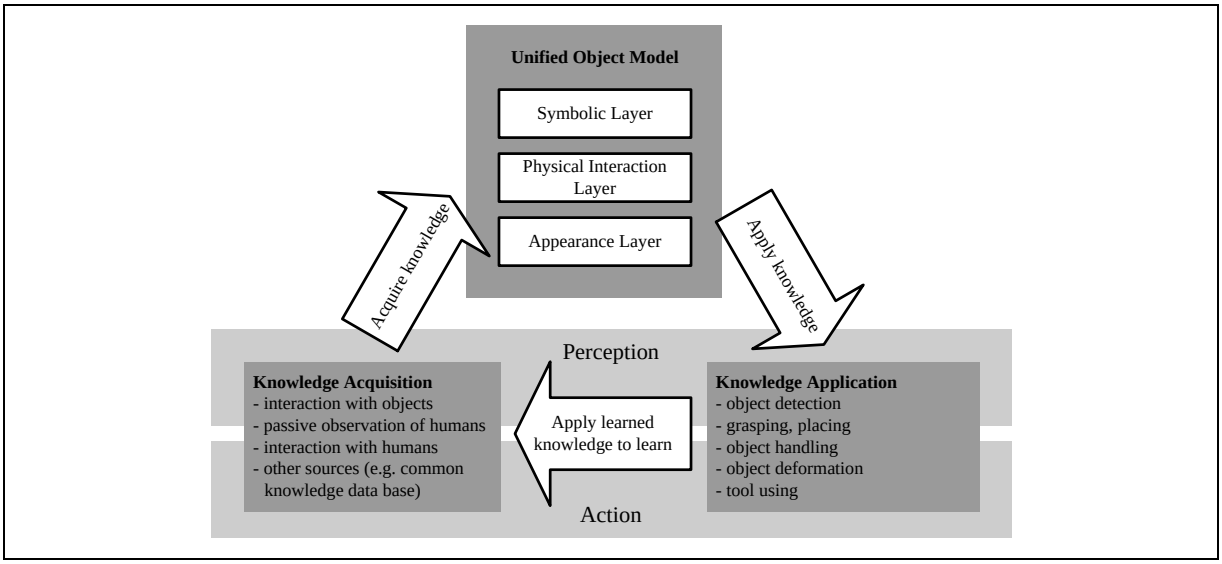


Figure 1: The unified object model and learning cycle as general concepts to enhance object models over operation time.

The following paragraph is an excerpt from [1] describing the concepts involved. The unified object model (Figure 1, top) consists of three main layers.
The lowest layer is the *appearance layer*. The work and results from classical computer vision related to object recognition with a strong focus on learning from example views fits into this level. However, also textured 3D models for object detection and recognition would be learned here.
The next layer is the *physical interaction* layer. This incorporates e.g. the association of grasping points to a geometric 3D model of the object. However, if the object has degrees of freedom (e.g. a pair of scissors or other tools) then descriptions of the object's mechanics would also belong into this layer.
The *symbolic layer* associates symbolic information to the object. Entries in this layer may contain class information such as part-of and kind-of relationships as well as object names or ownerships. If the object has 'functions' or associated actions then this information would also be included in this layer.
The object learning cycle is divided into two main groups of competences that include perception and action (Figure 1, bottom). The first group relates to the acquisition of knowledge i.e. to the extraction of information according to the object model. Within the second group, the aquired knowledge is applied for object detection and recognition. There is a link between both groups that describes the possibility that improved recognition and manipulation competences may also allow for more sophisticated learning strategies.

In the subsequent chapters we describe project work related to different object models and learning strategies carried out by IPA, UKA and LAAS.

1.2 IPA: Trainable appearance-based detection and model-based pose estimation (appearance layer)

An interactive learning scheme was developed at IP that enables the robot to train an appearance-based object detection system by showing the object. A range imaging and a colour imaging sensor are used. The goal is to find the coordinates and pose of the object relative to the sensor head with an appearance-based trainable detector and a pose estimator utilizing superquadrics as geometrical models. For the appearance-based detector SIFT feature key-points were used in this phase and Support Vector Machines were trained to label a key-point with object identifiers. This is called *key-point teaching*. For teaching the key-point distances are used for segmentation. This process is termed *range segmentation*. Furthermore, the superquadric modelling part of the system has been implemented with a particle swarm optimizer that is described in the 3d modelling part (below). In the following the parts of object learning are described in more detail. For further information on the techniques used (SIFT, SVM) the reader is referred to the references listed in [1].

Range segmentation with direct calibration

The term *direct calibration* refers to a calibration between the image coordinates of the range imaging sensor and the colour imaging sensor published in [2].

Figure 2: Range segmentation. Left: segmentation of colour pixels, right: segmentation of feature key-points (in magenta)

First pixels were extracted from the colour imaging sensor that correspond to a certain depth interval in the range image (Figure 2, left). Later the method was refined in that key-points were directly segmented based on their "distance" measured by the range imaging sensor.

Learning an appearance model for 3D shift and rotation invariant object detection

The goal of this part was to design an appearance-based object detection method that is trained ether by showing the object to the robot or when the object is grasped by the robot. In both cases we apply range segmentation. The idea is to combine the advantages of scale invariant key-points with the advantages of learning modern algorithms. The results of a first experiment were conducting and are published in [1]. We applied a SIFT filter to all three colour channels (RGB) of an image of objects of the so-called COIL data base. Key-point descriptors were used to train a one-class SVM. The trained SVMs were able to label key-points from the data base with high probability. We improved the method by applying the SIFT filter only to the V-layer of the HSV representation of the images and by appending H and S information to each key-point descriptor. With this it was possible to train SVMs

that have error rates less than 20% per feature key-point. In the future more than one key-point will be used per object to improve the results. The interesting point is that we used one-class SVMs. If n-class (batch learning) classifiers are used then the original views of all objects must be stored over operation time and the time for training one new object increases with the number of objects already known.
Good results have been achieved on the images of the COIL data base for the first ten objects. However, using a data base that was produced with the range segmentation method incorporating real-world objects have not yet led to acceptable results. Current and future work in this project phase focuses on this problem.

1.3: LAAS: Learning sensory-motor relationships from observation (appearance and interaction layer)

Existing learning approaches in the context of object modelling lack flexibility related to learning sensory-motor representations. At LAAS we investigate to use appearance-based representations and Pulsed Neural Networks for learning. As illustrated in figure 3, we use maps of neurons sharing the same sets of learning weights to obtain shift-invariance properties. These maps are connected in layers to extract complex features while avoiding extraction redundancy. The Hebbian-like rule used for learning. The neurons learn patterns in an unsupervised way, while respecting real-time computing constraints. The major contribution of this system is that neither the goal nor objects, actions or space representations are here known initially. This gives a chance to acquire knowledge from interaction, as every representation of each kind (objects, actions or space) stored in memory will be encoded in a same uniform language ready to learn associations. We believe that such a system provides for open-endedness. This robot-centric point of view and full-learning scheme will be validated on a simple two-wheeled robot system using standard color cameras for stereovision, but is not a priori constrained to one unique kind of platform or sensors.

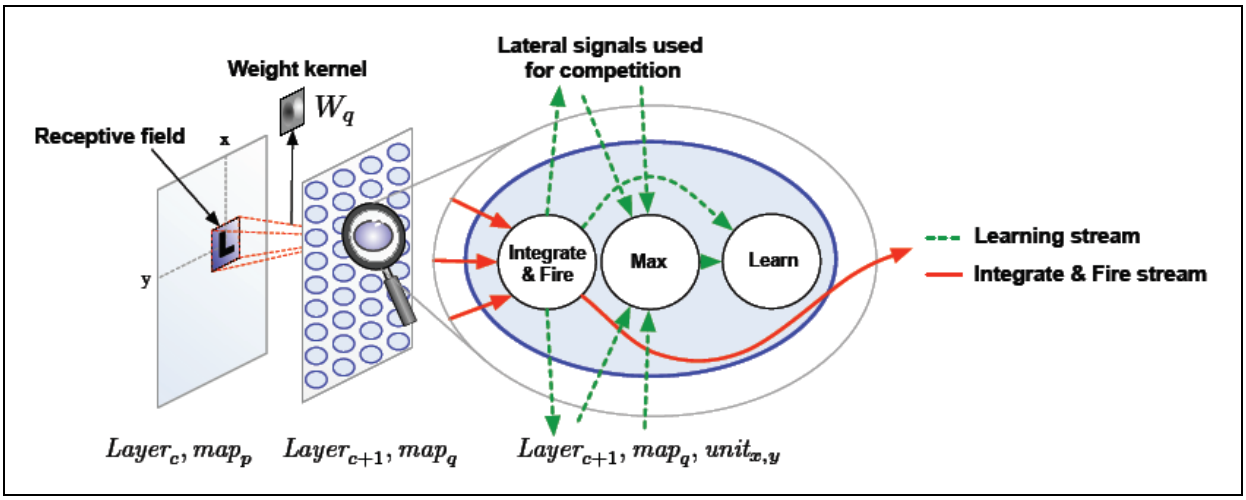


Figure 3: Neural structure of the learning method

In an experiment we analysed knowledge acquisition in terms of sensory-motor representations and with little *a priori* knowledge or representational choices to preserve generality and open-endedness. The purpose of the experiment is to be able to build representations by associating perception with action through an unsupervised learning process during which the robot interacts with its environment. The curiosity behaviour which drives the robot to explore newly perceived objects is related to increasing a value function. The underlying architecture we were experimenting here is a neural one as explained above. This architecture enables to learn the perceptual and motor representations, which are linked together. It also provides for distributed and hierarchical representations, starting from simple image processing, which we believe are a good basis for achieving open-endedness without being constrained by specific primitives. The Cogniron functions implemented in this

experiment is OR (Object recognition), embedded in the learning process. One important question that arises then is the possibility of integrating such an architecture with the symbolic-oriented ones. This issue will have to be investigated further in the next period. One approach would be to introduce a neuronal-symbolic interface that translates the neuronal representations into predefined data structures by means of a supervised categorization process in which the human tutor names the categories.
Detailed information on the neural network architecture is given in our paper [3]. Much more details on the experiment are given in the RA7 deliverable [4].

1.4: UKA, IPA & LAAS: Generation and refinement of object models (interaction layer)

UKA and IPA are working together on a combined representation, using a SwissRanger sensor coupled with a colour camera; UKA deals with the construction of a 3D superquadric model in order to make a "first grab" to feed the appearance-based object detection developed at IPA. At IPA also the matching part that gives a pose estimate of the object has been implemented in this phase. LAAS improves the incremental modelling method described below, and is integrating
a bayesian approach to cope with active shape-based recognition of an object.

Basic body extraction and model completion of Superquadric models (UKA)

The following work describes the work related to 3d superquadric models performed at UKA in collaboration with IPA. The aim is to allow for a "first grab" of an *a priory* unknown object and to refine the model in the subsequent steps. This is achieved through two processes: model completion and basic body extraction [1] and [5].
Model Completion. A single view does not contain enough information to construct a complete surface model of the object. Therefore, several views are taken from different angles, until the robot has exhausted the sphere segment centered on the object over which it can move its sensor. The resulting scans are registered using a fast variant of the ICP algorithm. Even if the sphere segment of possible sensor perspectives is exhausted, the resulting surface model will still not be complete in the general case. For instance the object might be located on a table in front of a wall. In this case the object's back and bottom are not observable. For real autonomous learning the object must be picked up by the robot. A partial model cannot supply manipulation information in the general case, however. This chicken-and-egg problem is solved using basic-body extraction.
Basic-Body Extraction. If only partial 3D information is available, for everyday objects it is often possible to approximate those using basic bodies sufficient for careful manipulation. This has been proved in literature using different shapes such as spheres, cylinders and cones, but it is more convenient to use a single versatile model such as the generalized cylinder or superquadric ellipsoids We chose superquadrics since they are capable of describing complex shapes ranging from cuboids over ellipses to spheres.
To adapt one or more superquadrics to a set of object points, a robust segmentation process first extracts those points from the background. In our initial experiments, we use an object standing on a table. The table plane is automatically extracted and defines background (everything outside the bounds of the table, projected upward and downward) and object (everything inside the bounds of the table but not part of the table plane itself). After this initial segmentation, clustering is performed to remove outliers and segmentation errors. Images of the related processing steps are shown in Figure 4.

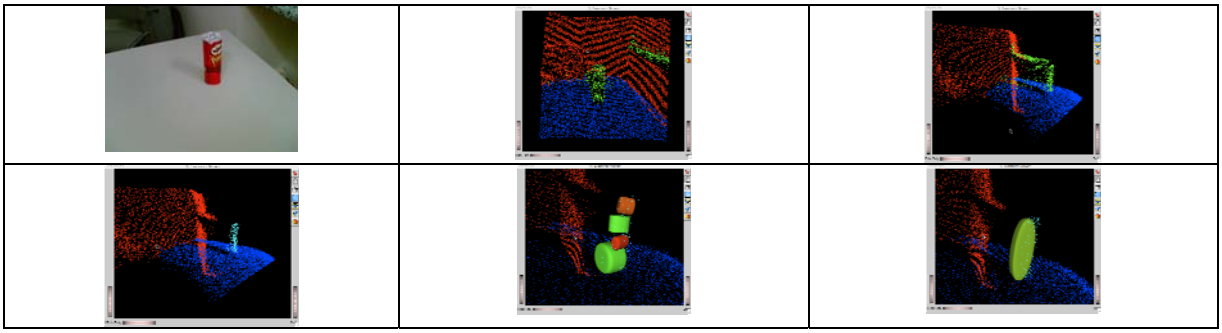


Figure 4: Top: Object on table, segmented range image from front and left. Bottom: Object points after clustering and outlier removal, Superquadric split, and Superquadric merge

Matching of Superquadric models (IPA&UKA)

At IPA we did first steps towards a trainable pose estimation method that copes with the imprecise data from the range imaging sensor. In the learning mode the parameters of a superqudric model are estimated based on views from the range image with an optimization algorithm (particle swarm optimization). This part is yet to be integrated with the basic body extraction and model completion methods from UKA. In the operation mode the optimizer regards the parameters as being fixed and optimizes the translation and rotation coordinates.

Mesh modelling of objects (LAAS)

LAAS in WP5.2 contributes on the incremental construction of a geometrical object model, as a triangular mesh, and on the automatic selection of the grasping positions from this model. Sensory data are acquired from a stereo sensor mounted on the wrist of a manipulator arm; 3D data are provided by a classical pixel-based stereo matching algorithm. The modelling method must cope with very noisy data and with classical stereo artefacts. An adapted version of the ICP algorithm, allows registering every view with the previous ones; ICP starts from estimated positions of the stereo sensor with respect to the robot reference frame, computed from rigid transforms provided by an offline hand-eye calibration, and from the effector position. In the current version, the triangular mesh is generated from the global cloud of 3D points obtained from all viewpoints, using the Marching Cubes algorithm, selected for his efficiency. Other methods (Ball Pivoting or Active surfaces) give better results, but require a longer computation time to build the 3D model [6]. The selection of the grasping positions consists in placing the gripper model with respect to the object reference frame, in order to handle it. At first, an approximate decomposition of the object model in rigidly coupled convex components is computed; then grasping positions are determined for every component [7].
The integration of these functions in KE2 is on the way. In the future, LAAS and UKA will compare the proposed methods to build an object model and to select the grasping positions.

2 Future work

Future work is described in detail in the new work plan RA5 (phase three of the project). Mainly, the work is a continuation of the activities here with a focus on further improvement and fusion. Furthermore the relationships between the Cogniron architecture and object models will be elaborated. Also we aim towards the symbolic layer of the unified object model. The following work areas are described in the next work plan:

- Autonomous and interactive learning schemes
- Fusion of appearance-based methods and geometrical models
- Open-ended learning algorithms for object models

3 References

3.1 Applicable documents

[1] J. Kubacki, B. Giesler, C. Parlitz: Active Autonomous Object Modelling for Recognition and Manipulation: Towards a Unified Object Model and Learning Cycle. Proceedings of "Fachgespräche für Autonome Mobile Systeme", 8th and 9th of December 2005, Stuttgart, Springer 2005,

[2] J. Kubacki & K. Pfeiffer: Using Range Imaging Sensors with Color Imaging Sensors in Cognitive Robot Companions: A New and Simple Calibration Technique Based on Particle Swarm Optimization, Proceedings of the First Range Imaging Research Day (RIM Day) 2005. September 2005.

[3] N. Dohuu , W. Paquier , R. Chatila: Combining structural description and image-based representation for image, object, and scene recognition, Report LAAS No05077,
19th International Joint Conference on Artificial Intelligence (IJCAI'05), Edinburgh (GB), 30th of July to 5th of August 2005, pp.1452-1457

[4] Cogniron Joint Deliverable D7.2, 2005.

[5] M. Walther, P. Steinhaus, R. Dillmann: A foveal 3D laser scanner integrating texture into range images, The 9th International Conference on Intelligent Autonomous Systems (IAS-9). Tokyo 2006.

[6] A. Restrepo Specht, M. Devy: Surface-segmenting modified Ball-Pivoting-Algorithm, Report LAAS No04543, 2004. International Conference on Image Processing (ICIP'2004), Singapore, 24-27 of October 2004, 4p.

[7] E. Lopez Damian, D. Sidobre, R. Alami: Grasp planning for non-convex objects
Rapport LAAS No05012, 36th International Symposium on Robotics (ISR'2005), Tokyo (Japan), 29th November to 1st, December 2005, 6p.

3.2 Related project papers

3.3 Other references

See the references in the applicable documents.

Appendix

The papers listed in the applicable documents section except the RA7 deliverable [4].

Active Autonomous Object Modeling for Recognition and Manipulation

Towards a Unified Object Model and Learning Cycle

Jens Kubacki[1], Björn Giesler[2] & Christopher Parlitz[1]

[1]: Fraunhofer IPA [2]: University of Karlsruhe (TH)

Abstract. In this paper the aim is combine the principle of active autonomous object modeling with results from the field of computer vision and 3D geometrical modeling for recognition purposes focusing on modern robotics. The goal is to make a first step towards a unified object model and learning cycle that allow for integration of inputs from different research activities related to recognition and modeling in order to enable a robot to actively develop models over operation time.

1 Introduction

In the European project COGNIRON [1] we study the perceptual, representational, reasoning and learning capabilities of embodied robots in human centered environments. The goal is that such robots will be able to learn new skills and tasks in an active and open-ended manner.

In order to recognize artifacts in their environment robots need internal representations or *models* that describe certain properties of objects so that they can be detected in the environment or so that a robot can manipulate them.

There are difficult problems involved in designing such powerful recognition systems. One is to find the right *type* of models and the other is the requirement for open-endedness. If virtually *all* objects are to be recognized then the robot should be equipped with capabilities that allow it to *learn* models either by interacting with the human or autonomously by interacting with the object.

In this paper the aim is combine the principle of active autonomous object modeling with results from the field of computer vision and 3D geometrical modeling. The goal is to make a first step towards a unified object model and learning cycle that will lead to continuous learning of object properties.

2 Related Research

Firstly, the work in this paper is influenced by the field of 3D object modeling of objects (see e.g. the related chapters in [2]). For complex physical manipulation it is e.g. required to grasp an object or to plan manipulator paths around it. 3D structure provides a bidirectional mapping from texture space to object coordinates, yielding accurate view synthesis and object pose reconstruction**.** Grasp planning can be done

using shape-primitive object approximation [3], which can be produced from 3D structure information.

Secondly, the work presented in this paper is influenced by results from appearance-based computer vision. There are approaches that rely on training an object recognition system with specific views from the object. Promising results have been achieved with scale-invariant key-points [4]. Also *pure* learning approaches seem to be suited for robust object labeling [5] on the basis of example views.

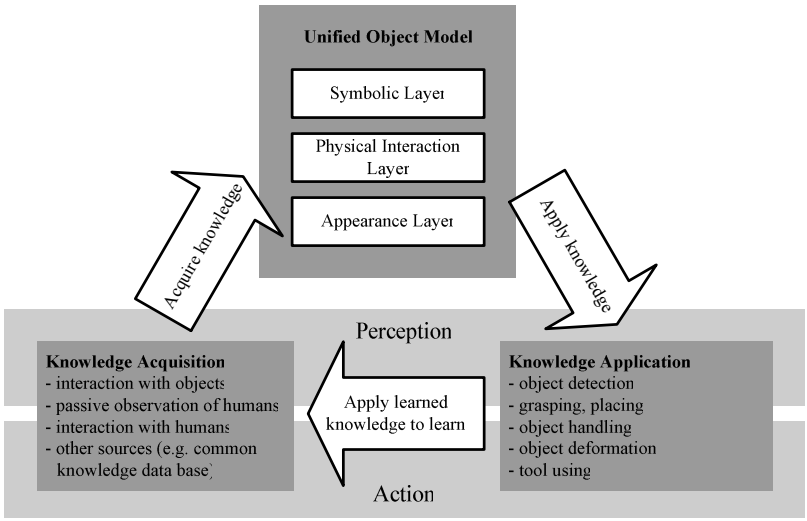
Since the robot's actuators are used for the training process the work can also be related to [6].

In contrast to other papers we take a rather pragmatic view on object modeling. The goal is to combine different methods that are already used in robotics with a focus on typical manipulation tasks that are investigated in the context of modern robotics. This view also includes the use of modern sensors (e.g. [7]) and typical three-layer control architectures.

3 Unified Object Model and Learning Cycle

In order to structure research work and to visualize basic problems and concepts involved in continuous object learning the unified object model and learning cycle shown in Figure 1 is presented. The aim of these concepts is to serve as a basic guide to structure inputs from different research activities related to object modeling.

Figure 1: The unified object model and object learning cycle as general concepts to enhance object models over operation time.



The unified object model (Figure 1, top) consists of three main layers. The lowest layer is the *appearance layer*. The work and results from classical computer vision related to object recognition with a strong focus on learning from example views fits

into this level. However, also shape-based matching would be located here if 3D information is available.

The next layer is the *physical interaction layer*. This incorporates e.g. the association of grasping points to a geometric 3D model of the object. However, if the object has degrees of freedom (e.g. a pair of scissors or other tools) then descriptions of the object' s mechanics would also belong into this layer.

The *symbolic layer* associates symbolic information to the object. Entries in this layer may contain class information such as part-of and kind-of relationships as well as object names or ownerships. If the object has ' functions' or actions associated with it then this information would also be included in this layer.

The object learning cycle is divided into two main groups of competences that include perception and action (Figure 1, bottom). The first group relates to the acquisition of knowledge i.e. to the *filling* of the of the object model's content. The second group deals with the application of the acquired knowledge to recognition tasks. There is a link between both groups that describes the possibility that improved recognition and manipulation competences may also allow for more sophisticated learning strategies.

4 Implementations and Results

For the unified object model and learning cycle are rather superficial concepts in this chapter a specific learning cycle related to a pick and place task is introduced to investigate object learning in more detail. Assumed are two sensors available: a range imaging sensor [7] and a conventional color imaging sensor mounted next to each other and looking into the same direction.

The specific object learning cycle assumes that the human supervisor places an object onto a *learning* table (Step 1). Then (Step 2) the robot fits a first but incomplete geometric model to the object that relates to the physical interaction layer of the unified object model. The object is separated from the background incorporation knowledge of the learning table and the background. The first model is used to grasp the object (Step 3) and to feed an appearance-based classifier model with example views by rotation the object in front of the sensors (Step 4). Here figure-background separation can be achieved using the measurements of the range imaging sensor and the known position of the object. The classifier model is related to the appearance layer of the unified object model. Then Step 2 to Step 4 are repeated until the geometric model and the classifier models are complete. In the following sub-chapters partial implementations of the two models are described.

4.1 Learning a Geometric Model for Grasping

Two methods are proposed for geometric object modeling: model completion and basic-body extraction.

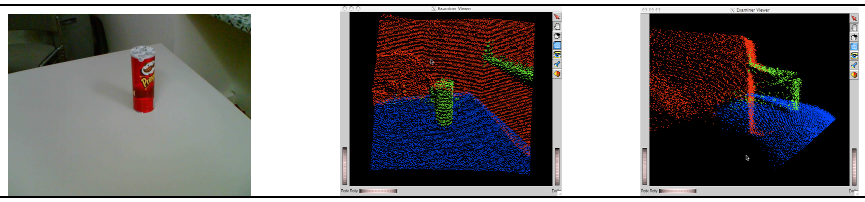
Model Completion. A single view does not contain enough information to construct a complete surface model of the object. Therefore, several views are taken from different angles, until the robot has exhausted the sphere segment centered around the object over which it can move its sensor. The resulting scans are registered using a

fast variant of the ICP algorithm. Even if the sphere segment of possible sensor perspectives is exhausted, the resulting surface model will still not be complete in the general case. For instance the object might be located on a table in front of a wall. In this case the object' s back and bottom are not observable. For real autonomous learning the object must be picked up by the robot. A partial model cannot supply manipulation information in the general case, however. This chicken-and-egg problem is solved using basic-body extraction.

Basic-Body Extraction. If only partial 3D information is available, for everyday objects it is often possible to approximate those using basic bodies sufficient for careful manipulation. This has been proved in literature using different shapes such as spheres, cylinders and cones [8], but it is more convenient to use a single versatile model such as the generalized cylinder [9] or, more recently, superquadric ellipsoids [10,11]. We choose superquadrics since they are capable of describing complex shapes ranging from cuboids over ellipses to spheres.

To adapt one or more superquadrics to a set of object points, a robust segmentation process first extracts those points from the background. In our initial experiments, we use an object standing on a table. The table plane is automatically extracted and defines background (everything outside the bounds of the table, projected upward and downward) and object (everything inside the bounds of the table but not part of the table plane itself). After this initial segmentation, clustering is performed to remove outliers and segmentation errors.

Figure 3. Left: Object on table, middle and right: segmented range image from front and left (red: background, blue: table plane, green: object including segmentation error)



A superquadric is parameterized by the 11-vector $\mathbf{a}=(\mathbf{a}_r|\mathbf{a}_t|\mathbf{a}_s)^T$, where $\mathbf{a}_r=(\varphi,\vartheta,\theta)$ is the vector of Euler rotation angles (below represented as rotation matrix $\mathbf{R}$), $\mathbf{a}_t=(x,y,z)$ is the translation and $\mathbf{a}_s=(a_1,a_2,a_3,\varepsilon_1,\varepsilon_2)$ is the 5-vector of shape parameters. With this parameterization, the superquadric is given by the equation

$$\mathbf{x}(\eta,\omega)=\mathbf{R}\begin{pmatrix} a_1\cos^{\epsilon_1}(\eta)\cos^{\epsilon_2}(\omega) \\ a_2\cos^{\epsilon_1}(\eta)\sin^{\epsilon_2}(\omega) \\ a_3\sin^{\epsilon_1}(\eta)\end{pmatrix}+\mathbf{a}_t,\ \text{for}\ -\frac{\pi}{2}\le\eta<\frac{\pi}{2}\ \text{and}\ -\pi\le\omega<\pi. \quad (1)$$

Fitting this parameterized equation into a set of N points $\mathbf{x}=(\mathbf{x}_1,...,\mathbf{x}_N)$ is a matter of minimizing the error function

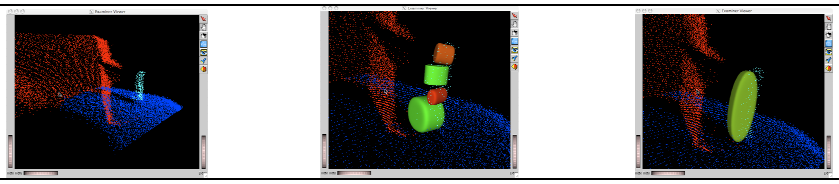
$$E(\mathbf{a})=\frac{1}{N}\sum_{i=1}^{N}\chi^2(\mathbf{a},\mathbf{x}_i) \quad (2)$$

where χ is a distance function relating the measurement $\mathbf{x}_i$ to the superquadric' s surface near it. The obvious choice for χ would be the Euclidean distance of $\mathbf{x}_i$ to its orthogonal projection $\mathbf{x}'_i$ onto the surface defined by $\mathbf{a}$. Since there exists no closed-form solution for determining $\mathbf{x}'_i$, we use the approximation suggested in [9], which uses the inside-outside function $F(\mathbf{a},\mathbf{x})$ (<1 if $\mathbf{x}$ is inside $\mathbf{a}$, =1 if $\mathbf{x}$ is on $\mathbf{a}$, >1 otherwise):

$$\chi(\mathbf{a},\mathbf{x}) = \sqrt{a_1 a_2 a_3}(F(\mathbf{a},\mathbf{x}) - 1). \tag{3}$$

Minimization of the error function is needed using least-squares approximation and refined using the Levenberg-Marquardt method. Since an object will not generally be well approximated by a single superquadric, a split-and-merge approach is used to first split the object point set into as few subsets as possible that can be approximated with superquadrics while keeping the error lower than a certain threshold T_s. A second threshold T_m is used as an upper error threshold for merging.

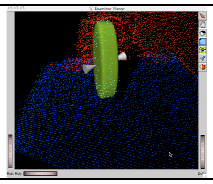
Figure 4. Left: Object points after clustering and outlier removal, middle: Superquadric split, right: superquadric merge



After split and merge, grasp points are determined from the resulting superquadric cluster. Automatic grasp point detection for arbitrary grippers has only been solved for objects made up of spheres, cuboids, cylinders and cones [12]. But since we use a simple two-finger parallel jaw gripper, we use a heuristic: A line is defined running parallel to the floor plane and at 45° angle to the robot' s front and side axes and passing through the cluster' s barycentre. This line' s outermost intersections with the cluster are used as initial grasping point estimates. They are then refined by gradient descent to minimize object rotation due to grasping.

After basic-body adaptation and grasping point analysis, the object can now be grasped and moved in front of the sensors to complete the surface model and to feed the appearance-based classifier part.

Figure 5. Resulting suggested grasping points.



4.2 Learning an Appearance Model for Invariant Object Detection

The goal of this part is to design an appearance-based object detection method that is trained when the object is grasped by the robot. The idea is to combine the advantages of scale stable key-points [4] with the advantages of learning algorithms [13].
The results of a first experiment are shown in Table 1. The SIFT filter was applied to all objects appearances of the first five objects of the COIL data base [14]. Each color layer (RGB) was processed separately. Then only the key point descriptors of the objects were extracted. Three numbers were appended to describe the color at the key-point' s location. The result is a large list of vectors with 131 dimensions (128 numbers for the key-point descriptors and 3 for color). The data was then reduced by a threshold applied to distances between learning samples (set to 160.0). The resulting numbers of key-point descriptors per object are shown in the second row of Table 1. Then one-class RBF-SVMs [15] were trained using the software from [16]. The parameter γ=0.00019 was used for the radial basis function. The parameter ν=0.001 was set to allow for errors on the training set. See [15] for an explanation of these parameters.

Table 1: Results of the SVM experiment. Key-point descriptors of an object can be learned by a single one-class RBF-SVM

COIL Object Number	1	2	3	4	5
No of overall key-points	9624	913	3889	1477	4359
No of support vectors	2250	572	1716	805	1777
Hit rate on object 1	0.89	0.12	0.52	0.16	0.57
Hit rate on object 2	0.58	0.68	0.48	0.24	0.51
Hit rate on object 3	0.47	0.16	0.80	0.15	0.43
Hit rate on object 4	0.60	0.24	0.58	0.74	0.43
Hit rate on object 5	0.12	0.08	0.34	0.57	0.81

This experiment shows that a single key-point descriptor can be associated correctly with high probability to the target object (see the gray-shaded recognition rates). The new and interesting point is that this is done with one-class SVMs. If n-class classifiers are used then the original views of all objects must be stored over operation time and the time for training one new object increases with the number of objects already known. Current work focuses on the reduction of the false hits on other objects then the one being searched for.

5 Summary and Conclusions

The aim of this paper is to combine different research activities related to object recognition in modern robots. For this purpose a unified object model with different layers related to recognition tasks is presented. A specific learning cycle for a pick and place task is introduced to investigate active modeling in more detail by describing current results in 3D modeling and appearance-based recognition. Further work is will be conducted to fully integrate of the two modeling parts into a real robot. Further conceptual work will focus on the unified object model and learning cycle.

References

1. Chatila R: The Cognitive Robot Companion and the European Beyond Robotics Initiative. 6th EAJ International Symposium "Living with Robots", 2004
2. Forsyth DA and Ponce J: Computer Vision: A Modern Approach. Prentice Hall 2002.
3. Miller AT, Knoop S, Allen PK, Christensen HI: Automatic grasp planning using shape primitives. In Proc. IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 1824-1829, 2003.
4. Lowe DG: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, Vol. 60, pp. 91-110, 2004.
5. Roobaert D, Zillich M, Eklundh JO: A Pure Learning Approach to Background-Invariant Object Recognition using Pedagogical Support Vector Learning. CVPR, Vol. 2, No. 2, pp. 351, 2001.
6. Fitzpatrick P, Metta G, Natale L, Rao S and Sandini G: Learning About Objects Through Action - Initial Steps Towards Artificial Cognition.
   Web page: citeseer.ist.psu.edu/fitzpatrick03learning.html
7. Oggier T, et al.: An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRangerTM). In: Proceedings of the SPIE, Vol. 5249, No. 65, 2003.
8. Lukacs G, Marshall AD, and Martin RR: Geometric least-squares fitting of spheres, cylinders, cones and tori. Report GML 1997/5, 1997.
9. Nevatia R and Binford TO: Description and recognition of curved objects. Artificial Intelligence, 8(1):77–98, 1977.
10. Leonardis A, Jaklic A, and Solina F: Superquadrics for segmenting and modeling range data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(11):1289–1295, 1997.
11. Jaklic A, Leonardis A, and Solina F: Segmentation and Recovery of Superquadrics. Kluwer, 2000.
12. F. Solina and R.K. Bajcsy: Recovery of parametric models from range images: The case for superquadrics with global deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(2):131–147, 1990.
13. Vapnik, VN: The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1998.
14. Nene SA, Nayar SK and Murase H: Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Columbia University, 1996.
15. Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, and Williamson RC: Estimating the support of a high-dimensional distribution. Neural Computation, 13(7):1443–1471, 2001.
16. Chang CC and Lin CJ: LIBSVM: a Library for Support Vector Machines (Version 2.31)
    Wep page: citeseer.ist.psu.edu/chang01libsvm.html

Acknowledgements

The work described in this paper was conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

Using Range Imaging Sensors with Color Imaging Sensors in Cognitive Robot Companions: A New and Simple Calibration Technique Based on Particle Swarm Optimization

Jens Kubacki & Kai Pfeiffer
Fraunhofer Institute for Manufacturing Engineering and Automation (IPA),
Nobelstrasse 12, 70569 Stuttgart, GERMANY

Abstract
A cognitive robot companion needs sensors in order to enhance its perceptual capabilities. Using different types of sensors in different mechanical settings there is a need to find suitable calibration models. Calibration can be performed between system components or relative to some global reference frame. In this paper we describe a method that can lead to useful calibration models in a generic manner. A particle swarm optimization algorithm is used to optimize a parameter set and suggest methods to reduce or to refine the calibration model iteratively. Relevant equations do not have to be reformulated as a system of equations and there is no principal restriction on the class of equations. The proposed method is applied to calibrate a new bi-modal sensor head in two different ways: directly between two sensors and to real-world coordinates.

Keywords
Calibration, particle swarm optimization, robot assistant, cognitive robot companion, perception, range imaging, automatic model selection.

1. INTRODUCTION

As a successor of the idea of robot assistants as published in [1] in the European integrated project COGNIRON we develop methods and technologies for the construction of so-called cognitive robot companions. These artificial creatures are expected to evolve and improve their capacities on-line i.e. during operation and in close interaction with humans [2]. In order to be able to enhance their perceptual, manipulation and reasoning capabilities the robots need to acquire representations about tasks, objects, locations and humans. One pre-requisite that enables the development of robust recognition and task execution is the physical coupling of the robot with the environment in terms of sensors and actuators. See figure 1 (top) for an example of a modern sensor-actuator coupling with the environment. Focusing on the sensor side, new technologies can serve as enablers for the acquisition of representations and hence for the development of robust recognition and task execution. Therefore, new developments in the field of range imaging sensors are highly relevant for the design of cognitive robot companions. To use modern range imaging sensors and other sensors in these robots it is necessary to calibrate the sensors either directly to each other or relative to some common reference. Depending on the type of device, peripheral components, and the mechanical setting there is a special set of equations needed for calibration. We call this set of equations also *calibration model*. In order to find such model it is useful to develop methods that can lead to this set systematically, either manually or possibly with the aid of a computer.

After describing the state-of-the-art in chapter 2 we introduce a new calibration technique in chapter 3. A particle swarm algorithm is used to optimize a general parameter set. Relevant equations do not have to be reformulated into an equation system and there is principally no restriction on the class of relevant equations. In chapter 4 the implementation of the method is sketched and in chapter 5 experiments are conducted on two different calibrations related to a bi-modal sensor head. In chapter 5 also necessary equations for a specific set-up are derived. Chapter 6 contains results and future work and chapter 7 conclusions.

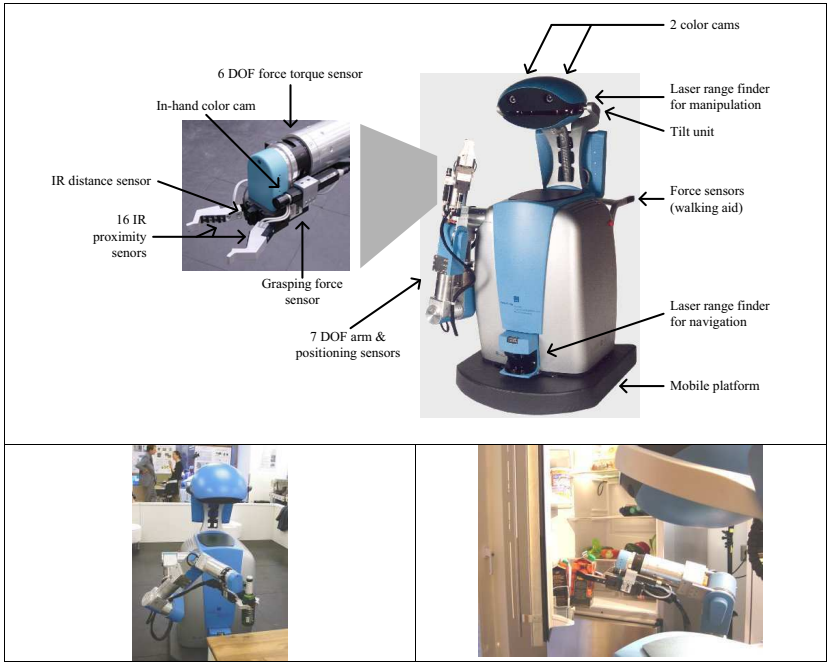


Figure 1: The Care-O-bot II (see [3]) as an example for a robot companion that needs to execute fetch-and-carry tasks. Top: The physical sensor-actuator coupling of the robot with the environment. Bottom left: The robot picks up a bottle of beer. Bottom right: the robot takes apple juice out of the fridge.

2. STATE-OF-THE-ART

2.1. Modern Range Sensors

One standard sensor device used in robotics is the color imaging sensor based on either CCD technology or CMOS technology. In addition to color information there are mainly three methods to produce range i.e. 'depth' images in robotics: (a) Using stereo-vision systems, (b) panning or rotating a 2d laser range finder, and (c) using 3d laser range finders.

Stereo-vision systems require for background structure in order to solve the correspondence problem. However, stereo-vision systems can work with high resolutions since they build on mature color imaging technology that has been driven by commercial applications.
2d laser range finders are standard components in the field of mobile robot navigation. They do not deliver a range image in one frame, but one 'line' of it reflecting the distances of a circular area in front of the sensor in one plane. If these sensors are panned or rotated it is possible to reconstruct a range image. Here a challenge is to synchronize the acquisition of single lines with the motion of the sensor. A drawback is the low effective frame rate as a result of this panning or rotation process. The Care-O-bot II shown in figure 1 utilizes a laser range finder which is integrated in the robot's head. Through panning of the head a 3d scene can be reconstructed. This scene is used for object localization and to plan and execute a collision-free trajectory for grasping.
3d laser range finders are also available on the market [4]. A drawback in mobile applications is the relative large volume of such sensors.
A promising new sensor technology is based on the time-of-flight measurement of light that returns on each photo element of an imaging chip after submission [5]. These sensors are very compact and deliver 3d range images in one frame. Already this technology has been shown to be suited for mobile robot navigation [6]. There are already a few sensors commercially available and others are in a prototypical state. See figure 2 for a rough overview.

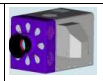
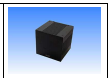
Product	 SwissRanger SR2	 1k-s (deve-lopment kit)	 19k PMD	 Canesta (deve-lopment kit)	 Zmini	 DMC 100	 Range Camera Prototype
Manufacturer	CSEM	PMD-Tec	PMD-Tec	Canesta	3DVSystems	3DVSytsems	Matsushita Electric Works
Reference	[7]	[8]	[9]	[10]	[11]	[12]	[13]
Pixel resolution	Up to 160x124	64x16	160x120	DP203, DP205, DP208: 64x64	752x582 (PAL) or 768x494 (NTSC)	510x492	128x123
Accuracy of one measurement	Down to 5 mm	> 6 mm	> 6 mm	> 6 mm	> 5 mm	> 3 – 5 mm	Not found
Frame Rate	Up to 30	Up to 50	Up to 10	Up to 15	Not found	Up to 60	Up to 15
Field of View	Range: 7.5m Angle: +/- 30° (diagonal)	Range: 7.5m Angle (HxV): 34° x 12°	Range: 7.5m Angle: 40°	Range: 5.0m Angle: 30°, 55°, 80°	Range: 3.5m Angle: 30°	Range: 2.5m Angle (HxV): 45° x 35°	Range: 7.5m Angle (HxV): 60° x 45°
Connection	USB 2.0	IEEE 1394a and Ethernet (IEEE 802.3u)	IEEE 1394a and Ethernet (IEEE 802.3u) and RS 232	USB 1.1	Not found	FireWire and RS 232	USB 2.0 and Parallel

Figure 2: Modern range imaging sensors that measure the time till re-arrival of a generated light pulse. These sensors are a considerable alternative to stereo-vision or mechanically animated laser range finders in robotics. Note that this is only a rough overview. The data printed in the table is drawn from the references given in the third row. Please refer to the manufacturers for details on a specific product.

We identify the following application areas in the context of cognitive robot companions relevant to modern range imaging sensors.

- Large scale geometrical modeling of space for navigation purposes (e.g. simultaneous localization and mapping)
- Reactive navigation competences: e.g. obstacle avoidance and person following
- Local scene reconstruction for object localization and collision-free grasping

- Geometrical human modeling for gesture and activity recognition
- Enhancement of color-based object recognition systems through range based segmentation

Note that this list is not meant to be complete. There may be more application areas in the context of robot companions for modern range imaging sensors. These applications we have extrapolated from our current activities in the field of constructing robot companions.

2.2. Calibration Techniques in Computer Vision

In computer vision the problem of calibration is usually split into the estimation of the *intrinsic* and the *extrinsic parameters* by solving a system of equations. The intrinsic parameters can include:

- Projection parameters of the pinhole model.
- Rotation and shift between the imaging chip and lens.
- Scaling parameter related to the imaging chip.
- Radial distortion of a real lens.

The extrinsic parameters include the 3d rotation and shift of the camera relative to some chosen world coordinate system.
To solve the calibration problem usually a system of equation is solved. A system of linear equations can be solved using standard methods. Nonlinear systems can be computed using the Gauss-Newton or Levenberg-Marquardt algorithms [16]. To cast complex models into a solvable system there is manual analysis work needed.

3. New Calibration Technique

3.1. Motivation

Developing robot companions we are often in a situation where we have to implement new sensors and/or actuators of various types in order to use them concurrently for recognition or task execution. Depending on the type of device and the mechanical setting of it relative to other components there is a special set of equations that is needed for calibration. Some devices need more complex models others may be modeled sufficiently with a reduced model (e.g. a camera equipped with some optics that does not lead to radial distortion). If e.g. different cameras are used in combination it is not clear which parameters are most relevant *between* them (e.g. equal lenses compensate for radial distortion when calibrating from one sensor into the other). Another issue is that for some tasks computationally cheap and thus fast methods are needed rather then very accurate ones. Furthermore, if the mechanical setting changes it should be possible to quickly adapt the calibration equations by either finding a new set of parameters or even by removing or adding terms. For these reasons, there is the aim to develop a method that allows finding a sufficient set of equations with minimal manual analysis effort possibly with the aid of a computer. Sufficient here means with acceptable (depending on the task) accuracy and generally low computational cost since the robot's on-board computers have limited capacity. Requirements for such a method are:

- Suited for learning from reference samples that can be produced by a simple set-up and procedure. In the best case autonomously by the robot itself.
- Independency of the class of equations that describe the problem.
- Simple combination of many equations possible.

- Easy editing/change of the model to be tested or even automatic selection of the equations needed.

We identify two types of calibration in the context of robot companions where such a method can help finding a sufficient calibration model:

- Devices are calibrated pair-wise directly to each other i.e. calibration is performed from the readings of one device into the readings of another. We call this *direct calibration*.
- All devices can deliver their measurements relative a common reference frame (either attached to the robot or to a world-coordinate frame attached to some point in the environment). In this paper this is referred to as *real-world calibration*.

The first allows for fast search algorithms the use e.g. two sensors in conjunction in the same recognition algorithm. The second is suited for comparisons, fusions and communications of measurement results in a common format.

3.2. Approach

In order to find a simple-to-use method that supports the search for a set of equations that suffices the requirements stated in the last chapter the following method is proposed.

1. Generate a list of reference points by either: (a) designing a reference object and specialized detection programs to detect a point attached to this object in different sensors that need to be calibrated or by (b) generating a sample point list manually. Augment the list with real-world coordinates if needed. The reference points can also be generated by the robot autonomously.
2. Implement all available transformation equations needed to fully describe the problem as a chain of programming steps in their explicit format. The equations are given in the manuals of the sensors or in the literature of the relevant scientific field. Some equations (e.g. coordinate transformation between two systems) are described in general mathematics, robotics or computer vision literature.
3. Cast the problem into an optimization problem by utilizing a stochastic search optimizer as parameter estimator that iteratively suggests and improves a set of parameters. For improvement a fitness function needs to be defined that measures an error criterion of the specific parameter set on the sample list. In this paper we use a particle swarm optimization technique [14]. This technique does not depend on the class of equations i.e. it assumes no special properties of the functions involved.
4. Reduce or refine the set of equations (a) manually or (b) using some automatic scheme and repeat step 2 and step 3 with the new calibration model until a sufficient set of equations is found.

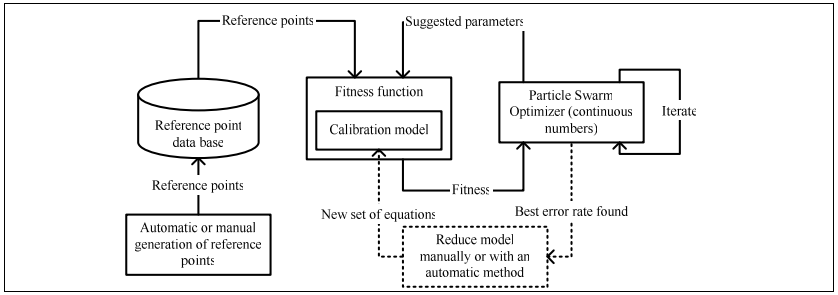


Figure 3: Sketch of the proposed method to derive sufficient calibration models. The model to be tested can be implemented in a simple explicit representation within a fitness function that is called by a particle swarm optimizer. The fitness function has access to the reference points to calculate an error measurement (fitness). The optimizer iteratively converges to a parameters set that gives a low error rate. The model can be refined or reduced manually or by some automatic method.

The particle swarm algorithm is used in this paper as a parameter estimator. The algorithm is described in detail in [14]. An advantage of this algorithm in comparison with other stochastic search algorithms, such as genetic algorithms is that it is simple to implement and that it is easy to use as an optimizer for problems in continuous numbers. In addition it can cope very well with multimodal maxima in the search space. Using this algorithm for our calibration procedure has the following advantages:

- It is still possible to get good results if parameters are not independent or even redundant.
- Not restricted to a special class of equations (e.g. the model equations can contain discontinuities)

In figure 3 is shown how we use the algorithm as parameter estimator with a data base of reference points in our current implementation.

3.3. Example: Calibration of a Bi-Modal Sensor Head

As an example application for the proposed general approach to finding sufficient calibration models we chose the calibration of a bi-modal sensor head. The next generation of our robot companion (Care-O-bot III) is planned to incorporate a time-of-flight range imaging sensor [7] mounted next to a CCD color imaging sensor [15]. The basic hardware setting is shown in figure 4.

We apply both concepts to the bi-modal sensor head:

1. *Direct* calibration from the three readings (two image coordinates and range) of the range imaging sensor to the two image coordinates of the color imaging sensor
2. *Real-world* calibration from the three readings (two image coordinates and range) of the range imaging sensor to the x-, y-, and z-components in a chosen sensor reference frame.

The first strategy is aimed to allow for recognition algorithms that work in the range and color image concurrently and the second method allows to communicate recognition results to other parts of the system (e.g. for grasping an object).

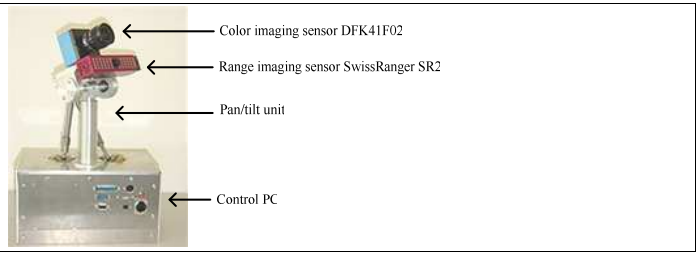


Figure 4: The experimental set-up of the sensor head includes a range imaging sensor [7], a color imaging sensor [14], a pan/tilt unit, and a control PC.

We describe the two types of calibrations formally as:

Direct Calibration:	$(u_R, v_R, d_R) \mapsto (u_C, v_C)$	(1)
Real-World Calibration:	$(u_R, v_R, d_R) \mapsto (x_R, y_R, z_R)$	(2)

where u_R and v_R are the discrete coordinates on the imaging chip of the range imaging sensor, $d_R = f(d_{Rm})$ is the distance to the range imaging sensor that is a function of the distance reading d_{Rm} delivered by the sensor, and u_C and v_C are the discrete coordinates on the imaging chip of the color imaging sensor. Furthermore, the linear relationship termed *distance function* is adopted from [6]:

Distance function:	$d_R = c_{R1}(d_{Rm} + c_{R0})$	(3)

We will now derive the equations necessary for the direct calibration and for the real-world calibration of the sensor head described in the chapter above. See figure 5 for the notations that are borrowed from [16] and [17]. As a reference frame we chose the frame assigned to the range imaging sensor $\{R\}$. The reference frame of the color imaging sensor is described with $\{C\}$. The point $\mathbf{p}_R = (x_R, y_R, z_R)$ is projected onto the image planes of the two sensors. $\mathbf{p}'_R = (x'_R, y'_R, z'_R)$ and $\mathbf{p}'_C = (x'_C, y'_C, z'_C)$ describe the projections. The z-components z'_R and z'_C are fixed and equal to the corresponding focal lengths f'_R and f'_C of each sensor device. The x- and y-components (x'_R, y'_R, x'_C, y'_C) describe the 2d coordinates in the image planes of the sensors. On the left side in figure 5 u_R, v_R, u_C, and v_C denote the discrete sensor readings resulting from the point's projections.

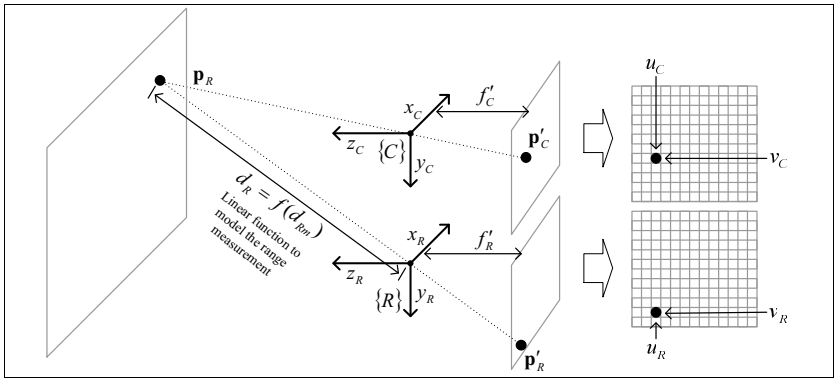


Figure 5 Notation for the transformation equations of the sensor head used in this paper

Now let us derive the equations needed for the direct calibration and the real-world calibration. First we introduce the perspective projection (as described in e.g. [16]) of the range imaging sensor using the pin-hole camera model.

Projection range imaging sensor:	$x_R = \frac{x'_R}{\lambda_R},\ y_R = \frac{y'_R}{\lambda_R}$ and $z_R = \frac{f'_R}{\lambda_R}$	(4), (5) and (6)

Here, λ_R is a scaling parameter which can be calculated for the range imaging sensor by:

Scale parameter range imaging sensor:	$\lambda_R = \frac{\sqrt{(f'_R)^2 + (x'_R)^2 + (y'_R)^2}}{d_R}$	(7)

Eventually, we assume scaling and shifting between the lens and the sensor chip as described in [15].

Scaling and shifting range imaging sensor:	$u_R = k_R x'_R + u_{R0}$ and $v_R = l_R y'_R + v_{R0}$	(8) and (9)

For the direct and real-world calibration we want to derive (u_C, v_C) and (x_R, y_R, z_R) on the basis of (u_R, v_R, d_R). Thus, we reverse equations (8) and (9):

Scaling and shifting range imaging sensor:	$x'_R = \frac{(u_R - u_{R0})}{k_R}$ and $y'_R = \frac{(v_R - v_{R0})}{l_R}$	(10) and (11)

The real-world calibration can now be calculated by combining equations (4) to (7), (10), and (11). For the direct calibrations we need to introduce the relationship between the coordinate systems of frame $\{R\}$ and frame $\{C\}$. We do this by introducing a coordinate transformation

Coordinate transformation:	$\mathbf{p}_C = \mathbf{R}_{CR}\mathbf{p}_R + \mathbf{o}_{CR}$	(12)

Where $\mathbf{R}_{CR}$ is the rotation matrix and $\mathbf{o}_{CR}$ the translation vector of $\{C\}$ relative to $\{R\}$.
Now we can derive the projection model of the color imaging sensor:

Projection color imaging sensor:	$x'_C = \frac{x_C}{z_C}$ and $y'_C = \frac{y_C}{z_C}$	(13) and (14)

Note, again we assume $f'_R = 1$. In coherency with the range imaging sensor we model scaling and shifting of the imaging chip relative to the lens.

Scaling and shifting color imaging sensor:	$u_C = k_C x'_C + u_{C0}$ and $v_C = l_C y'_C + v_{C0}$	(15) and (16)

We can now derive the full model for the direct transformation by combining equations (4) to (7) and (10) to (16).

4. IMPLEMENTATION

4.1. Detection of the Reference Object in Both Sensors

As a reference object we chose a red ball and a point associated to the centre of the cyclic region that the projection of the ball yields in the image plane of the corresponding sensor. This is rotationally invariant and can be detected easily in both sensors. The detection programs first generate binarized images and then searches for circular 'blobs' using a simple sub-procedure. In the color image only red areas are binarized. In the range image consecutive depth intervals are evaluated separately. The first (starting from the sensor and moving into the scene) interval that contains a circular blob is used for further processing. Note that this method is subject to errors since the detected points in the sensors are not located precisely on the same position.

4.2. Using Particle Swarm Optimization for Parameter Estimation and Model Search

The particle swarm optimization (PSO) algorithm is based on a directed random search in the parameter space. Please refer to [14] for a detailed description. A fixed-size group of so-called individuals searches for maxima. Each individual has access to a pre-defined set of neighbors in that it can see their fitness values and stores its own best-so-far fitness value. Using these inputs a velocity vector is generated and added to the current position. In high-dimensional parameter spaces the group behaves like a swarm that first moves to and then oscillates around potential maxima. In our calibration method the PSO repeatedly 'suggests' parameter vectors and gets a fitness value as feedback. The fitness value is in this paper defined as a negative quadratic error average between the estimated values using the suggested parameters and the real values. For our parameter estimation problem we enhanced the algorithm with the following features:

- Store the overall best-so-far individual to not loose a good solution that has been found already.
- Overwrite the initial values of the individuals with a pre-defined vector if there is an initial assumption of the parameters available.
- Initialize with a high number of individuals and take only the first few bests for further processing. This is to get better starting values as in the original version of the algorithm.
- Reduce the maximal velocity if there is no improvement of the overall best-so-far individual after a defined number of iterations. This is to find refined solutions if the algorithm oscillates around a potential maximum.
- An *incremental mode* that first optimizes the first parameter and then successively releases the remaining parameters. The starting parameters are set to neutral values.

The incremental mode is used to support to find simple calibration models automatically. First the optimizer finds a good solution to the problem by only estimating the first parameter of the first equation. The remaining parameters are set to a neutral value (i.e. to one in product terms and to zero in sum terms and so on). After a defined number of iterations the next parameter is released and the optimizer now varies both parameters. Then the next parameter is released and so on. While the equations are important to the quality of the model the user can observe a clear reduction of the error. If the error does not decrease anymore while new parameters are released one can assume that further parameters are not relevant for the problem that is defined ,by the reference points.

5. EXPERIMENTS

5.1. Real-World Calibration of the Range Imaging Sensor

The real-word calibration of the sensor head of the form $(u_R, v_R, d_R) \mapsto (x_R, y_R, z_R)$ was conducted using our approach described in chapter 3. First, an industrial robot was moved through the scene of the range imaging sensor. This is shown in figure 6.

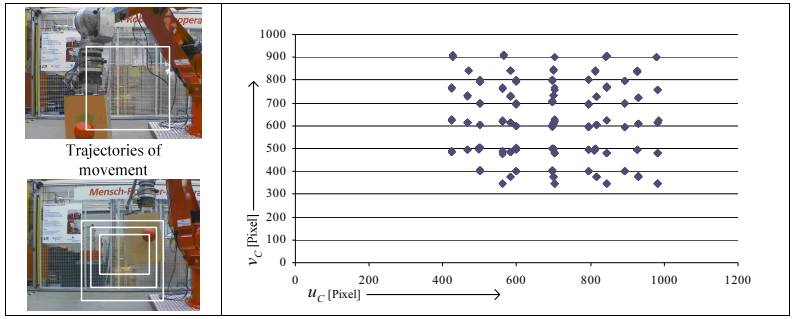


Figure 6 Set-up and results of the generation of reference points using the range imaging sensor. The robot was moved along three consecutive rectangles (left). Right: The resulting points in the image of the range imaging sensor.

We set up a data based with 73 reference points:

$$\{(u_{R1}, v_{R1}, d_{R1}, x_{R1}, y_{R1}, z_{R1}), \ldots, (u_{Rn}, v_{Rn}, d_{Rn}, x_{Rn}, y_{Rn}, z_{Rn})\} \quad (17)$$

where $n = 73$ is the number of reference points.

The real-world calibration can be calculated by combining equations (4) to (7), (10), and (11). The particle swarm algorithm was used to estimate four parameters: u_{R0}, l_R, v_{R0} and f'_R. Since the scaling parameters and the focal length are not independent we set $k_R = 1$. In order to use the particle swarm algorithm as fitness function was defined:

$$fit = -\frac{1}{3n} * \sum_{i=1}^{i=n} (x_R - \hat{x}_R)^2 + (x_R - \hat{y}_R)^2 + (z_R - \hat{z}_R)^2 = -err^2 \quad (18)$$

where $(\hat{x}_R, \hat{y}_R, \hat{z}_R)$ are the estimated values using the suggested parameter set.

The resulting error on the reference data set was $err = 44.52$ mm and is therefore higher then the error reported in [6]. It is not expected that the PSO algorithms missed a maximum point. Rather, there are errors expected in the detection algorithm for the reference object.

5.2. Direct Calibration of the Sensor Head

The direct calibration of the sensor head of the form $(u_R, v_R, d_R) \mapsto (u_C, v_C)$ was carried out in the same fashion then the real-world calibration. However, here we not only searched for the parameters of a model. We also tried to derive a simplified model for the direct calibration. The reference points for the direct calibration do not need to be annotated with real-world coordinates. This is an advantage that can significantly simplify the generation of reference points. To demonstrate this we set up a data base through moving the reference object manually through the scene (see figure 7

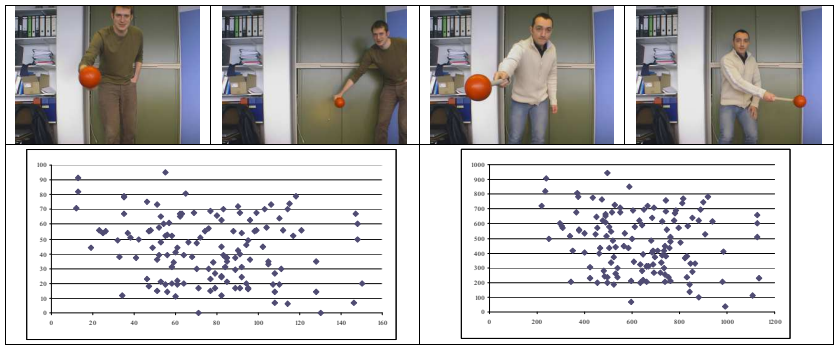


Figure 7 Set-up for the direct calibration. Top row: example images from the color sensor. Bottom row: reference points detected in the range imaging sensor (left) in pixel coordinates and reference points detected in the color imaging sensor (right) in pixel coordinates.

We set up a data base with 134 reference points of the form:

$$\{(u_{R1}, v_{R1}, d_{R1}, u_{C1}, v_{C1}), \ldots, (u_{Rn}, v_{Rn}, d_{Rn}, u_{Cn}, v_{Cn})\} \quad (19)$$

where n is the number of reference points.

As fitness function was defined:

$$fit = -\frac{1}{2n} * \sum_{i=1}^{i=n} (u_R - \hat{u}_R)^2 + (v_R - \hat{v}_R)^2 = -err^2 \quad (20)$$

where $n = 134$ is the number of reference points and $(\hat{u}_C, \hat{v}_C)$ are the estimated values using the suggested parameter set. Note that this fitness measure does not correspond to an error measurement in real-world.

We can construct the full model for the direct transformation by combining equations 4 to 14. Again we set $k_R = 1$ and also $f'_C = 1$ to account for the dependencies between the parameters. We also simplified the coordinate transformation by assuming $\mathbf{R}_{CR}$ is the identity matrix. There were eleven remaining parameters to estimate: u_{R0}, l_R, v_{R0}, f'_R, k_C, u_{C0}, l_C, v_{C0} and the three components of $\mathbf{o}_{CR}$.

The particle swarm algorithm optimized all eleven parameters and achieved an error of $err = 13.88$ pixels in the color imaging sensor. This is a good result since the range imaging sensor has 8 times more pixels in width and 7.74 times more pixels in height than the color imaging sensor. Thus the diagonal of 11.13 pixels would be an expected measure of this error which is given *a priory* through the different resolutions of both sensors and quantization errors. The specific values of the parameters are not published here since they depend on our specific mechanical setting. On other data sets we obtained better error rates around $err = 9.5$ pixels in the color imaging sensor.

Next, we were interested in how our calibration method can be used to find a simplified set of equations that still yields an acceptable error on the reference points. Therefore, we started the particle swarm algorithm in the incremental mode. It could be observed that only releasing the first four parameters led to significant reduction of the error.

Thus, we implemented a presumably simplest version for the direct calibration, a linear relationship described by equation (15) and (16) with directly inputting u_R and v_R by assuming $x'_C = u_R$ and $y'_C = v_R$. The best error rate achieved with this models was $err = 15.82$ pixels in the color imaging sensor. An advantage of this model is that it can be reversed. Thus, it is possible to get the range value at a corresponding pixel in the color imaging sensor.

6. Results And Future Work

For our bi-modal sensor head we found calibration models of different accuracy and complexity. The real-world calibration with an error rate of $err = 44.52$ mm needs to be improved. However, this is beyond the scope of this paper. One direct calibration model works with eleven parameters and achieves an error rate of $err = 13.88$ pixels in the color image.

Figure 8: Left: The range image (with normalized histogram for illustration). Right: The color values for pixels in the range image generated with the direct calibration that uses eleven parameters. The black banner at the bottom is caused through the mechanical setting of the sensors.

Using our method with the particle swarm optimizer in incremental mode it was possible to reduce the model to a linear version with four parameters that is even reversible such that it can give the distance corresponding to a color pixel in the color imaging sensor. Combining the direct calibration and the real-world calibration we constructed a 3d scene which is shown in figure 9. When the scene is slightly rotated (figure 9, left) then errors can be detected.

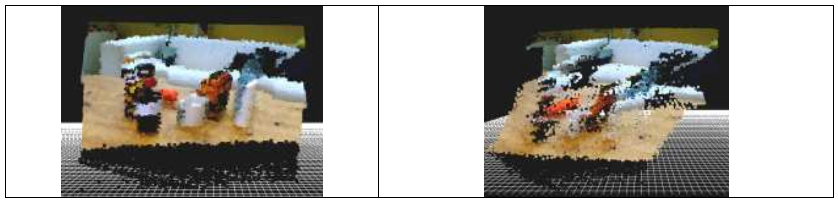


Figure 9: Left: The scene of figure 7 in a 3d space. Right: the 3d scene slightly rotated. The errors are given by principle since the scene is only taken from one view.

In future we will further enhance the calibration models for the sensor head by using our proposed method and setting up high-quality reference point data bases. Furthermore, we aim to segment object appearances in the color image using range information. A first implementation is shown in figure 10. Here a 'blob' in the range image is detected first that corresponds to the position of the gripper of a robot. Then the coordinates of the bounding rectangle of this blob are transformed to the color image. The rectangle segments the color information of the object plus some disturbances from the background and the robot's gripper.

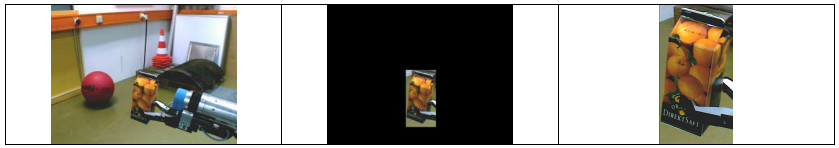


Figure 10: Left: The scene with the object attached to the robot's gripper. Centre: The segmented region by applying the linear direct transformation method. Right: The segmented region in large size.

The calibration method could potentially be extended to a fully automatic version. Here, an interesting problem lies in finding the order in which automatic methods 'try' different sub-sets of the equation to converge to a sufficient model.
Furthermore, there is current work on fusing different views of the scene into an improved 3d representation. First the coordinate transformation parameters from one view to the next are estimated then the scenes will be merged to get an improved scene.

7. Conclusion

The proposed method can easily be used to find suitable calibration models of different types incorporating different sensors. The method is simple since it uses the relevant equations explicitly and does not assume any special properties of the equations. This paper presented a scheme how the calibration model can be refined or reduced with minimal effort. Also a first attempt to reducing models automatically by using a particle swarm optimizer in an incremental mode that moves from simple to more complex models was presented.
We hope that this work can help others that use complex sensor settings or that it is may be one starting point into the wider field of automated model selection in the context of calibration problems.

8. Acknowledgements

The work described in this paper was conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

9. References

[1] Hägele M., Neugebauer J., Schraft R. D. et al. (2001): From Robots to Robot Assistants. In: Proceedings of the 32nd International Symposium on Robotics in conjunction with IMS 2001 April 19 - 21, Seoul, Korea, Vol. 1, pp. 404 - 409

[2] Raja C. (2004): The Cognitive Robot Companion and the European Beyond Robotics Initiative. In: 6th EAJ International Symposium "Living with Robots" 2004 October 4 and 5, Tokyo, Japan. http://www.cogniron.org/Publications.php.

[3] Hans M., Graf B., Schraft R. D. (2002): Robotic Home. Assistant Care-O-bot. Past - Present - Future. In: IEEE Industrial Electronics Society u.a.: IEEE ROMAN 2002: Proceedings, 11th IEEE International Workshop in Robot and Human Interactive Communication, 2002 September 25 - 27, Berlin. Piscataway, NJ: IEEE, 2002, pp. 380 – 385

[4] Perceptron Inc. (1993): LASAR Hardware Manual. 23855 Research Drive, Farmington Hills, Michigan, 1993.
[5] Oggier, T. et al. (2003): An all-solid-state optical range camera for 3D real-time imaging with sub-centimeter depth resolution (SwissRangerTM). In: Proceedings of the SPIE, Vol. 5249 No. 65, 2003. ttp://www.swissranger.ch/pdf/CSEM_manuscript_5249-65.pdf
[6] Weingarten J., Gruener G., Siegwart R. (2004): A State-of-the-Art 3D Sensor for Robot Navigation. In: Proceedings of IROS' 2004, Sendai, September 2004. http://asl.epfl.ch/aslInternalWeb/ASL/publications/uploadedFiles/2004iros_weingarten.pdf
[7] Swiss Ranger SR2: CSEM SA. Swiss Ranger SR-2 Datasheet. <http://www.csem.ch/corporate/Awards/p_531_SR-2_Preliminary-0355.pdf>. Rev. 2005.07.22
[8] PMD[vision] 1k-s (development kit) PMD Technologies: PMD Technologies. Datasheet PMD[vision]® 1k-s. <http://www.pmdtec.com/inhalt/produkte/documents/PMDvision1k-S_000.pdf>. ISSN: Rev. 2005.07.22
[9] PMD[vision] 19k PMD Technologies: PMD Technologies. Datasheet PMD[vision]® 19k. <http://www.pmdtec.com/inhalt/produkte/PMDvision19k_001.pdf.pdf>. Rev. 2005.07.22
[10] Canesta (development kit): CanestaVision™ EP Development Kit. Datasheet with detailed specifications. <http://www.canesta.com/downloads/11005-01%20rev%206%20EP%20DevKit%20Data%20sheet%20PSD.pdf>. Rev. 2005.07.22
[11] Zmini: 3DV Systems. Zmini datasheet. <http://www.3dvsystems.com/products/Zmini/Zmini%20Data%20Sheet.pdf>. Rev. 2005.07.22
[12] DMC 100 3DVSystems: 3DV Systems. DMC100 System Specification. <http://www.3dvsystems.com/products/dmc.html>. Rev. 2005.07.22
[13] Range Camera Prototype: Matsushita Electric Works, Ltd. Press release. <http://www.mew.co.jp/e-press/2004/0402-01.htm>. Rev. 2005.07.22
[14] Eberhart, R., Shi, Y. and Kennedy, J.: Swarm intelligence. Morgan Kaufmann, 2001.
[15] 1394 Imaging. Web page of the color imaging sensor DFK42F02: http://www.1394imaging.com/products/cameras/dfk41f02/?sid=85d8622059239e8ffa36d8c8c25762c0
[16] Forsyth, D. A., Ponce, J. (2003): Computer Vision: A Modern Approach. New Jersey. Prentice Hall, 2003
[17] Craig, J. J. (1989): Introduction to Robotics: Mechanics and Control. New Jersey. Addison-Wesley, 2nd edition, 1989
[18] Vapnik, V. (1998): The Nature of Statistical Learning Theory. New York: Springer-Verlag, 1998.

Combining Structural Descriptions and Image-based Representations for Image, Object, and Scene Recognition*

Nicolas Do Huu†, Williams Paquier and Raja Chatila
LAAS-CNRS
7, avenue du Colonel Roche 31077 Toulouse Cedex 4 - France
nicolas.dohuu@laas.fr, william.paquier@laas.fr, raja.chatila@laas.fr

Abstract

Object and scene learning and recognition is a major issue in computer vision, in robotics and in cognitive sciences. This paper presents the principles and results of an approach which extracts structured view-based representations for multi-purpose recognition. The structures are hierarchical and distributed and provide for generalization and categorization. A tracking process enables to bind views over time and to link consecutive views. Scenes can also be recognized using objects as components. Illustrative results are presented.

1 Introduction

Object and scene learning and recognition is a major issue in computer vision, in robotics, in neuroscience, and in cognitive sciences as well. And one of the main questions to this respect is how to extract knowledge from 2D patterns of light in the camera or the retina.

For the recognition of objects, two models were proposed in the literature. In the 'image-based' or 'view-based' model, an object is represented as a collection of view-specific local features [Poggio and Edelman, 1990; Ullman, 1998; Tarr and Bülthoff, 1998; Riesenhuber and Poggio, 1999]. Representations are organized in trees in which a set of view-tuned units constitutes the weighted inputs to a higher level object-invariant unit. Each unit measures the similarity between the input image and its stored view, and the higher level unit computes the weigthed sum from its incoming connections. If the resulting value reaches a given threshold, then the learned object is recognized. Riesenhuber's HMAX model of object recognition in the ventral visual stream of primates also proposes a similar grouping method where higher level cells compute the maximum response of view-tuned cells [Riesenhuber and Poggio, 1999].

The second model is based on structural descriptions. One of the most important approaches is the "Recognition-By-Components" (RBC) model [Biederman, 1987] in which each object is represented as a collection of volume parts. Thus, there is no need of multiple view-tuned representations since 3D models can be virtually rotated and compared to the input image. Moreover, the use of such a model can make the recognition invariant to illumination and color.

We can identify certain properties of the modeling process for efficient learning and recognition. Firstly, the training and thus the construction of new representations must be sufficiently fast. Lack of speed is the principal cutoff of models based on structural descriptions because building 3D models from images remains a non trivial task. Secondly, an efficient modeling process must organize knowledge in a structured way. A first means of organizing the data is to build categorical representations. Categorical structures make it possible to obtain capacities of class generalization at low cost. Indeed, to effectively exploit the extracted data, those must be describable at various levels of specificity: a bottle can be described as a container, a plastic object for recycling, etc. Such a structure can also accelerate recognition that deals with large collections of stored objects by reducing the number of candidate object models. Another mode to structure meanings is the decomposition of a representation as the set of its parts. Thanks to structural descriptions, RBC offers more robustness, in particular with respect to noise and occlusion. Moreover, it is interesting to share components among several structural descriptions to save memory: one could use the representation of a wheel to describe either a car, a truck or a bike. Thirdly, a learning system for knowledge extraction and structuring must allow the addition of new representations at non-prohibitive memory cost and without explosion of complexity. This open-endedness property can be approached by using the two kinds of data structures referred to above, in which a representation can be factorized in categories or shared as components.

While considering object recognition one must mention important results achieved by template-based approaches for object classification and recognition [LeCun *et al.*, 2004]. Nearest Neighbor methods, Support Vector Machines and Convolutional Networks provide efficient solutions but the needs of structured knowledge, reusability, incremental and autonomous learning are several points addressed here which are not, to the best of the authors' knowledge, well dealt with by the pre-cited methods.

This paper presents an efficient approach [Paquier, 2004]

*The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Companion") funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

†supported by the European Social Fund.

(which is partially implemented) for building structured representations without using 3D primitives and exhibiting the properties mentioned above. The paper is organized as follows. Section 2 presents the model we choose to both extract and organize representations. Each essential property is presented and illustrated by an example produced by the implemented system. Section 3 introduces the view-binding algorithm and the incremental building of object-invariant detectors. The use of objects as landmarks for buidling structured representation of scenes is presented in section 4. We conclude in section 5.

2 Architecture for View-based Recognition

We will first describe the *map* structure which constitutes the basic building block of our model, and discuss its inherent shift-invariant property. In the learning process, the maps will specialize in certain features, and we will present the way we associate them in order to prevent redundancy in this specialization and structure extracted features.

2.1 *Maps* and Inter-*maps* Connectivity

The *Map*: a Set of Local Classifiers

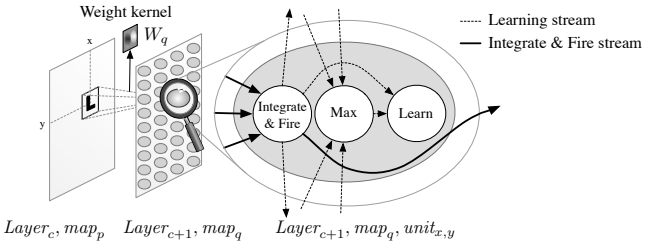


Figure 1: A *map* is a collection of *units* sharing the same set of weights (*kernel*). Each unit is composed as a three stage calculation pipeline. It receives signals incoming from previous layers for integration and lateral signals from *maps* situated on the same layer for competition.

In order to satisfy real-time constraints, the learning and recognition system must be able to update its representations at a frequency that is compatible with the modifications of the environment, and process images at a rather high frequency (e.g., 15 fps). With this end in view we chose to implement an integrate-and-fire model in local classifiers built as calculation pipelines (figure 1).

We call a *map* a collection of local classifiers called *units*, organized in a retinotopic[1] way. Each *map* is associated with one or more afferent *maps* which are the locations of its *units*' input domains (or their receptive fields). Receptive fields of each *unit* are static and each *unit* is thus associated to a set of afferent *units* in each afferent *map*. We denote Ω_i the set of afferent *units* of a given *unit* U_i.

At time t and for a given unit U_i, information flows from its receptive fields through a set of learning weights $W_i(t)$. Learning weights are organized in *kernels*, and there are as many weight kernels as afferent *maps*. All the *units* of a *map* are sharing the same set of weights, thus they can detect and learn a pattern which is at different positions in the input *maps*. As shown in figure 1, the coordinates of a unit in a given map and its receptive field in the afferent *map* are the same. Therefore the position of an active unit corresponds to the position of the detected pattern, which provides for a shift-invariance property.

This type of mapping has been previously introduced by Fukushima's *Neocognitron* and was successfully applied to handwritten digit recognition [Fukushima, 2003]. Figure 2 illustrates this intrinsic *map* feature. In this experiment, we produce an image containing a set of four randomly distributed letters (k, p, s, u). After less than a minute, the system has learned autonomously its own distributed representation of the input image. Each resulting *map* can extract its learned feature at any location in the input image. To obtain this result we must also provide our system with an inter-*maps* communication channel, so that *maps* don't learn the same feature several times. This procedure is called "local competition" and is introduced next.

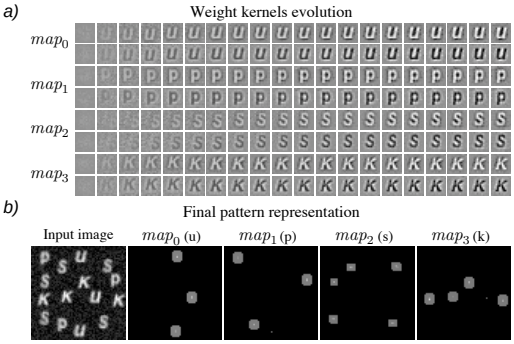


Figure 2: Letter extraction from noisy image input (20% random noise, 20% scale and angle variation, 70 steps, 2 images per second). a) weight kernels evolution across time. b) input image and associated *maps* activations.

Preventing Redundancy with Competition

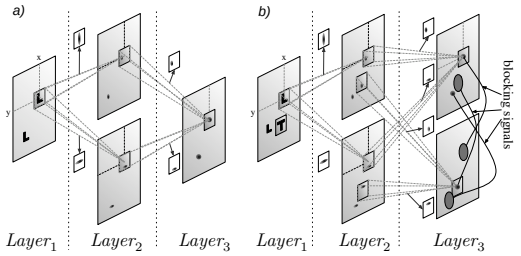


Figure 3: Hierarchical *maps* connectivity and local competition. a) The letter "L" is detected as a relative positioning of horizontal and vertical bars. b) input image containing the letters "L" and "T": inter-*maps* competition with local inhibition leads to distinct specialisations.

One previously mentioned requirement concerns the struc-

[1] A retinotopic mapping means that stimuli that are adjacent to each other in the visual world are processed by adjacent sets of *units*.

tural decomposition into several components. To achieve such a decomposition, different maps observing the same inputs must "decide" to specialize into distinct component detectors. The undergoing process is illustrated in figures 3.a and 3.b. In this example, we want to train two *maps* to detect the letters "L" and "T" in the input image. A first step of extraction is composed of a layer of two *maps* which respectively detect vertical and horizontal bars in their inputs. When the letter "L" is present in the image (figure 3.a), the pattern is decomposed as the positions of these two local orientations. The relative positioning of the two patterns is then detectable and can be learned by a dedicated *map* on the next layer ($Layer_3$).

From this point, if we add another letter ("T") to the input image and a second learning *map* to the third layer to learn it (figure 3.b), then we must prevent this new *map* from learning the "L" pattern one more time. This differentiation is achieved by using a local competition in which we compare *map* detection values and choose the best fitted (See section 2.2 for the computation of the detection value).

Right after calculating its detection value, every unit of each *map* broadcasts it to all other *units* on *maps* of the same layer and at the same coordinates. Thus, every *unit* whose coordinates correspond to the position of the letter receives incoming values at the 'Max' stage of its pipeline architecture (see figure 1). The 'Max' stage, as its name indicates, computes the maximum incoming value and sends it to its own 'Learn' stage. The learning process is then able to compare the local *unit* activation incoming from the 'Integrate & Fire' stage to distant activations. By allowing learning only to the best fitted *unit*, we ensure that no *map* could learn one pattern if another is already specialized to detect it. We illustrate this inter-*maps* competition on Figure 3.b with the dark discs which represent the blocking signal.

Connecting *Maps* to Build Distributed and Hierarchical Representations

We adopted a layered hierarchical architecture for several reasons. Firstly we need this architecture for *categorization*. As our units achieve linear separation in their input set, complex features cannot be extracted with a single layer. This is a known limitation of the single layer perceptron that cannot simulate an exclusive disjunction (logical XOR). The second reason is *reusability*. A huge number of extracted features are encountered in different shapes: oriented segments, arcs of circles or color blobs are building blocks for more complex images. Since extracted information is shared in the network, we avoid redundancy and the resulting computational overhead. Following the same idea, similar meanings should be encoded using shared sets of units. For example, the internal representations of a truck and of a car should intersect. This intersection representing the shared meaning could contain the internal representation of the four wheels, the steering wheel, etc. Thanks to this distributed representation architecture, it is easier to pool similar objects into cross categories (e.g., wheeled vehicules).

This capability of building hierarchical and shared representations is shown in figure 4. In this experiment we trained a network to recognize faces using a set of ten different images of each of 40 distinct subjects[2]. In the same way as in the previous example of letter extraction, $Layer_2$ learns to detect the position of each eye in one *map* and the position of the mouth in the other one, sharing the same segments extraction in the previous layer ($Layer_1$). In $Layer_3$, we train another *map* which receives as inputs the detection values of the mouth and eyes specialized *maps*. As a result, this last *map* learns the representation of the face with relative positioning of eyes and mouth. We also see in this example that the resulting face recognition is robust to high variations in subject posture and morphology. Nevertheless, this kind of recognition only applies to images with limited face orientations. In section 3, we will propose additional methods to recognize 3D objects based on the pooling of overlapping views.

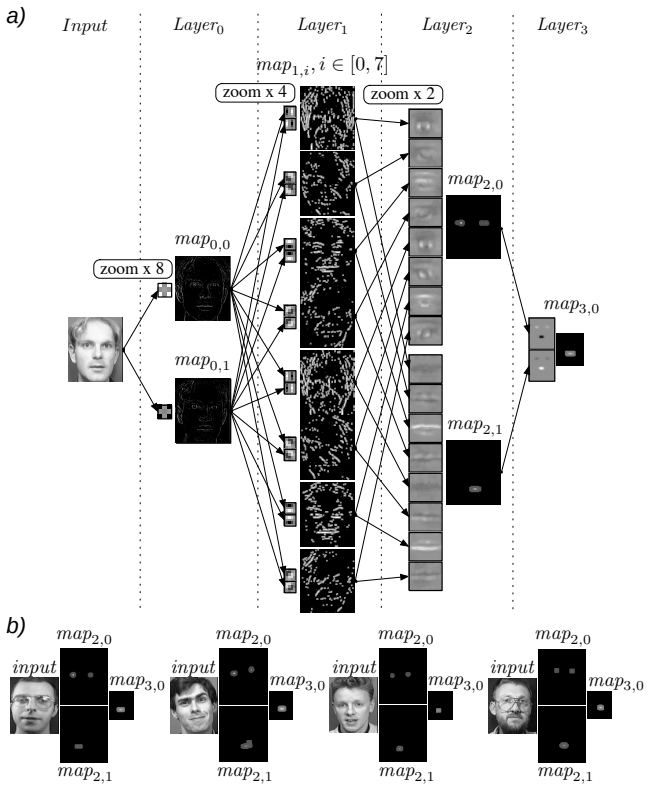


Figure 4: An example of a hierarchical representation of faces. a) network connectivity showing weight kernels and *maps* activities. $Layer_0$, $Layer_1$, $Layer_2$, $Layer_3$ respectively extract contrasts, oriented segments, eyes and mouth, and face. b) recognition robustness for different subjects and variable postures.

2.2 Integration, Firing and Learning

Now that we have presented the global mechanisms involving the computing *units*, we present the internal workings of the system more precisely and the computation of the detection values mentioned in section 2.1.

[2]We used the faces database of the AT&T Laboratories, Cambridge http://www.uk.research.att.com/facedatabase.html

Integration
Given a *unit* U_i, we denote $\beta_i(t) \in [0,1]$ its calculated output burst at time t. The burst is the detection value mentioned before. Let $w_{ij}(t) \in W_i(t)$ be the learning weight associated to the connection between *units* U_i and U_j, and $\Omega i^+(t)$ a subset of afferent Ω_i defined as:

$$W_i(t) = \{w_{ij}(t), U_j \in \Omega_i\} \quad (1)$$

$$\Omega_i^+(t) = \{U_j \in \Omega_i, \beta_j(t) > 0\} \quad (2)$$

We also define three weight sums as follows :

$$S_i^\star(t) = \sum_{w_{ij}(t) \in W_i(t)} \mid w_{ij}(t) \mid \quad (3)$$

$$S_i^{\beta^+}(t) = \sum_{U_j \in \Omega_i^+(t)} w_{ij}(t) \quad (4)$$

$$S_i^+(t) = \sum_{w_{ij}(t) \in W_i^+(t)} w_{ij}(t) \quad (5)$$

where $W_i^+(t) = \{w_{ij}(t), U_j \in \Omega_i \; and \; w_{ij}(t) > 0\}$.
Then the integrated potential P_i at time $t+1$, which expresses a level of similarity between the learnt pattern and the inputs, is given by:

$$P_i(t+1) = \frac{\alpha_P P_i(t) + (1-\alpha_P) S_i^{\beta^+}(t)}{S_i^+(t)} \quad (6)$$

Where $\alpha_P \in [0,1]$ is a fixed potential leak.

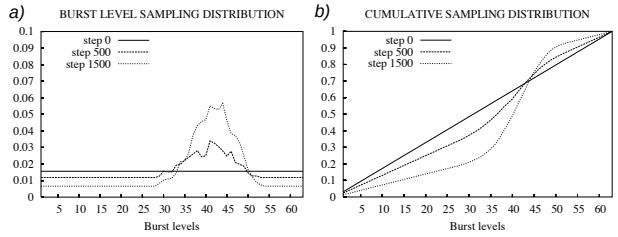


Figure 5: Burst sampling distribution and cumulative distribution evolution. a) burst sampling distributions computed at steps 0, 500 and 1500. Burst values are discretized within 64 levels. b) corresponding cumulative distribution used as a transfer function for integration.

Firing
Output bursts β_i generated by a unit U_i are produced both for integration processes situated on the the next layer and for its own "Learn" stage (see figure 1). In the first case, burst levels are not taken into account since all positive bursts participate to integration (see equation 2). In the second case, we previously explained that we need to compare *units* activation in order to find the best fitted *map* in a competition process. The burst level is thus computed as follows:

$$\beta_i(t) = \begin{cases} 0 & \text{if } P_i(t) < T_i, \\ F_i(t, P_i(t)) & \text{otherwise.} \end{cases} \quad (7)$$

Where T_i is a fixed threshold and F_i is an adaptive transfer function whose variations across time are illustrated by figure 5.b. The underlying idea of this function is that, as units are learning, their output function must converge to a binary-like response corresponding to a more strict linear separation. At each step we compute a sample distribution of the input burst by discretizing the output burst value in 64 levels. The transfer function F_i is in fact the associative cumulative distribution at time t:

$$F_i(t,x) = \sum_{0<b\leq x} f_i(t,b) \quad (8)$$

Where function f_i is a sampling distribution of burst levels updated at each step (figure 5.a). We thus obtain a sigmoidal transfer function which permits a binary-like classification behavior and a more effective competition process. Until now, object detection is internally represented as the activation of a limited set of computing *units*. Due to our connectivity scheme, recognitions are indeed only reflected at positions located near the centre of the learned pattern. However patterns presence are more than just their center since the whole surface they cover is relevant. So we increase the pool of activated units to cover a wider surface. As a first approach, we chose a simple disc shape whose surface is given by $S_i^\star(t)$.

Learning
We use in our model a hebbian-like learning rule in which *units* tend to learn patterns responsible for their activation. In other words, a learning process occurs when one *unit* has both fired and won the competition in its layer. Moreover, we compute a stochastic standard deviation σ_{ij} for each weight w_{ij} which is computed and used as coefficient in the learning calculation as follows:

$$\mu_{ij}(t+1) = (1-\alpha_W)\mu_{ij}(t) + \alpha_W \mid \gamma_{ij}(t) \mid$$
$$\sigma_{ij}(t+1)^2 = (1-\alpha_W)\sigma_{ij}(t) + \alpha_W[\mu_{ij}(t+1) - \gamma_{ij}(t+1)]^2$$
$$w_{ij}(t+1) = (1-\alpha_W)w_{ij}(t) + \alpha_W[1 - 2\sigma_{ij}(t+1)]\gamma_{ij}(t+1)$$

Where μ_{ij} is a stochastic mean and α_W the learning rate. Hence we ensure that weights corresponding to noisy inputs (with high standard deviation) will tend to 0 and then won't take part in the representation. γ_{ij} is a function of input burst of *unit* i which takes its values in $\{-1, 0, 1\}$. *Maps* among the same layer are in competition as shown in figure 3. If at time t, an input burst from *unit* j is received $\gamma_{ij}(t) = 1$, else if a burst is received from any other afferent *unit* then $\gamma_{ij}(t) = -1$. Finally if no burst was present $\gamma_{ij}(t) = 0$.

3 From Views to Objects

In this section, we consider the 3D-object recognition problem. Let us consider a camera rotating around an object. For a given orientation we can train a dedicated *map* and obtain enough robustness to detect this view, say within about a 45 degrees angle interval centered on the learnt prototype. Wider intervals can be obtained with "simple" objects, such as balls or paper cups, whose views are relatively invariant during view-point modification, while turning around them. Our approch is to pool overlapping view detectors. Then, a view-invariant (or object-tuned) detector can be obtained simply by computing the maximum view-tuned *map* response [Riesenhuber and Poggio, 1999]. This will provide for continuous object recognition.

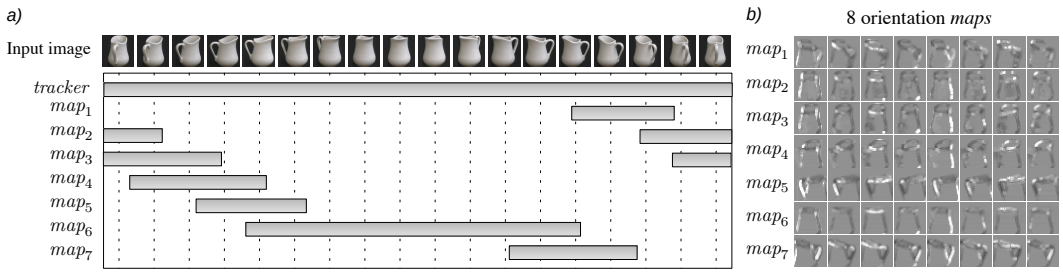


Figure 6: Object invariant detection with view-tuned *maps*. a) *maps* activities overlap to produce continuous detection. b) resulting learning weights. These are composed of 8 kernels for each *map* which correspond to 8 afferent oriented segment detectors situated on previous layer (see figure 4).

Assuming the support of an external module producing the initial object detection (this could be simply a user's selection on the input image, but this module could be based on saliency, motion, etc...), we must resolve several remaining problems. Firstly, the object-tuned *map* must keep track of the object during point of view modifications (temporal binding). Secondly, as we have no initial information about the object's complexity and, therefore cannot anticipate the exact number of required view-tuned *maps*, we must find a method which incrementally adds new *maps* to the network and associate them together.

Map-tracker : Short-term Memory for Temporal Binding

The different views of the same object tend to occur close together in time and space [Tarr and Bülthoff, 1998]. Several authors proposed to take advantage of this property [Stinger and Rolls, 2002; Wallis, 1996]. We particularly want to mention Stinger and Rolls's hypothesis about the functional architecture and the operation of the ventral visual system tested in their model called VisNet . In this model, each activation of a view-invariant neuron is maintained during a short period of time. The pooling is then achieved by using an associative memory to link temporal memory trace and current view detection. Although this method seems biologically plausible, it seems difficult to apply to real-time robotics. As the view-to-object membership is not *a priori* known, there is no criteria to decide whether or not a newly specialized *map* corresponds indeed to the view of an object. If no preselection is applied, the size of the associative memory could exceed computable capacities. So, it is crucial to restrict *map* specialization to relevant views. A solution comes from a tracking approach. In this domain, temporal coherence is also used but in another way. The aim of tracking is not so much extracting knowledge but rather following the position of a collection of pixels. Temporal coherence can here be exploited by considering that persitent informations are preserved between two frames and can thus be extracted. We propose to use the *map* architecture for pixel tracking in order to follow the position of the object and use this information as a first step for an incremental view-tuned *map* creation.

We can obtain the desired tracking capability by increasing the learning rate α_W of a map (see section 2.2), which we will call tracker-*map*, to a value near 1. The tracker-map is trained at the time the first object position is produced by the external module. We call this process "one-shot learning" because of the use of a high learning rate. In the next frame, thanks to the robustness of our model, the tracker-map can detect the object even if it has started to move. According to the hebbian rule, this detection is followed by a learning phase. Then, another learning occurs which almost completly renews the stored prototype due to the high learning rate. From now the process can restart allowing continous detection. Tracker-*maps* can be viewed as short-term memories that never reach stable representations.

Temporal Binding

Now that we can use temporal coherence to track the object position, we must resolve the remaining problem of building incrementally view-specific representations of the tracked object. To reach this goal we granted tracker-*maps* the capability of creating new maps during runtime. As we are in the context of moving objects and/or moving point of views, one-shot learning (learning with high learning rate will also be used to catch object's views during movement. The very specific resulting prototype will be refined as view-tuned maps compute standard learning rates. The binding algorithm develops as follows: if no view-tuned *map* recognizes the object at the position of the tracker-*map*'s detection during a short period of time (we use a 5 frames time-out), which is initially the case, then the tracker creates a new *map* and forces it to learn the unknown view with a one-shot-learning signal. If a view is detected, because a *map* has previously learnt it, then a standard learning process occurs in this *map* and the detection produces a one-shot learning signal for the tracker-*map* in order to focus tracking on this view. This process ensures that long-term memory of a specific view has a higher priority during object detection than the tracker. This algorithm has been used in the experiment of figure 6 in which the image of a rotating object has been used to train a three-layers network. In figure 6.a, we see that the tracker always detects the object, thanks to the feedback from long-term to short-term memory, and that activity of map_6 is about 3 times longer than the others. Indeed, this *map* learned to detect orientations that are rather invariant during rotation. Our model can thus adapt the number of required *maps* depending on the complexity of objects.

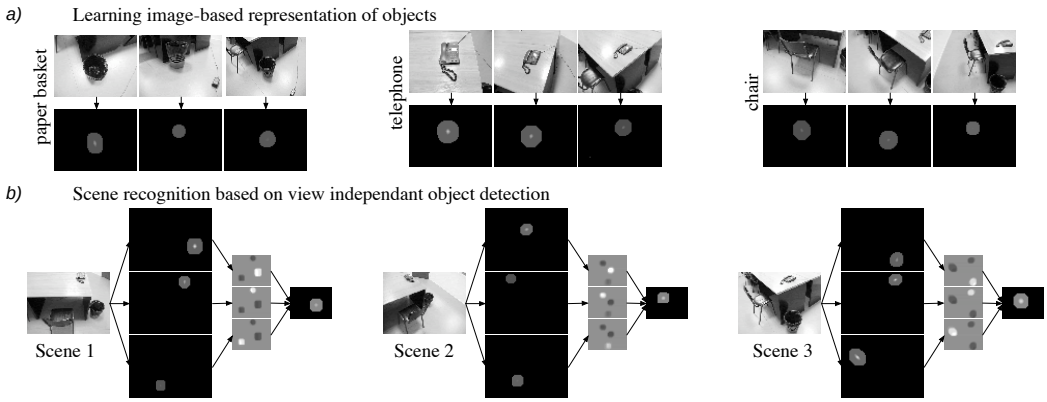


Figure 7: Representing scenes as arrangements of objects in an office environment. a) examples of object-invariant detections from different points of view learn. b) results of scene recognition by relative positioning of object-invariant detections.

4 Objects as landmarks for scene recognition

We present in this section experimental results of our system in the context of scene recognition. We can define a scene as a particular arrangement of objects. We demonstrated previously with face recognition that our architecture is able to construct hierarchical representations of an image. We can use this same technique in order to learn a scene as relative arrangement of objects. For this experiment we observed a standard office environment including 3 basic objects: a paper basket, a phone and a chair. In a first phase, object representations are learnt on the fly using temporal coherence by the means of tracker-*maps* (section 3). The resulting network then detects each object simultaneously in view-specific *maps* and in the corresponding object-invariant tracker-*map*. In a second phase, three higher level *maps* connected to all object-invariant detectors learn representations of the scene from three different points of view. Figure 7.a shows the outputs of the three object-invariant detectors built as presented in section 3 with natural images. Without needing any additional algorithm, our model extracts a view-point specific representation of the scene in a hierarchical structure (figure 7.b). From now, a location-invariant *map* can be trained thanks to tracker-*maps* for scene-invariant *maps* pooling as explained in section 3. These multi-layered architecture allowed by parallel-pipeline calculations is the key property which permits to combine view-based representations and structural descriptions.

5 Conclusion

In this paper we have introduced several methods and algorithms to extract knowledge from visual data. Our model is able to build representations by decomposition of image features into local components structured in a global hierarchy of concepts. Such decompositions are minimal requirements for both sharing representation in order to save memory and computational time, and providing generalization capabilities due to the fact that extracted components can be used as description blocks to recognize new elements. The main advantage of our model is that one does not need any prior knowledge or model to build robust dedicated detectors of simple to complex features in a unified language. Thus, building heterogeneous libraries of meanings becomes feasible. These representations could be then provided as input to a symbolic reasoning level.

References

[Biederman, 1987] I Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[Fukushima, 2003] K. Fukushima. Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180, April 2003.

[LeCun *et al.*, 2004] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR'04*, 2004.

[Paquier, 2004] W. Paquier. *Apprentissage Ouvert de Représentations et de Fonctionnalités en Robotique : Analyse, Modèle et Implémentation*. PhD thesis, 2004.

[Poggio and Edelman, 1990] T. Poggio and S. Edelman. A network that learns to recognize three dimensional objects. *Letters to Nature*, 343:263–266, 1990.

[Riesenhuber and Poggio, 1999] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 28:1019–1025, 1999.

[Stinger and Rolls, 2002] S. M. Stinger and E. T. Rolls. Invariant object recognition in the visual system with novel views of 3d objects. *Neural Computation*, 14(11):2585–2596, November 2002.

[Tarr and Bülthoff, 1998] M. Tarr and H. Bülthoff. Image-based object recognition in man, monkey and machine. *Cognition*, (67):1–20, 1998.

[Ullman, 1998] S. Ullman. Three-dimensional object recognition based on the combination of views. *Cognition*, 67:21–44, 1998.

[Wallis, 1996] G. Wallis. How neurons learn to associate 2d-views in invariant object recognition. Technical Report 37, Max-Planck-Institute für Biologishe Kybernetik, Tübingen, Germany, 1996.

Book Title
Book Editors
IOS Press, 2003

A foveal 3D laser scanner integrating texture into range data

Marcus Walther, Peter Steinhaus and Rüdiger Dillmann
Institute for Computer Science and Engineering
Universität Karlsruhe
email: {mwalther,steinhaus,dillmann}@ira.uka.de

Abstract. In the meantime the acquisition of dense point clouds or triangulated surface data with 3D laser range scanners is in the meantime a widely discussed topic. Especially in the areas of world modelling or autonomous navigation reliable 3D data is necessary. Mobile systems applications like collision avoidance or simultaneous localisation and mapping (SLAM) require high data rates, dense point clouds, constant availability of data and exact single scans. Commercial 3D laser scanner are mostly expensive, not mobile, too slow or not exact enough. We present in this article a 3D laser range scanner which is based on a commercial 2D scanner with a movable origin. In contrast to other "self-build" 3D scanners our scanner reaches high data density in heading direction and lower density at the boundaries (foveal vision), which is important for autonomous navigation.

Keywords. 3D vision, depth image, texture integration, world modelling

1. Introduction

In autonomous navigation, collision avoidance and world modeling are still major topics towards real autonomy. Open question is how to collect, arrange and store environment information suitably. Depending on the task different types or amounts of data have to be collected. In general it can be seen that dense 3D point clouds suit well for most applications. Hence the task for the robot's sensory equipment is as follows:

- Scanning of a wide region of the surrounding space
- Scanner range from some centimeters up to several meters
- High accuracy of the laser scanner
- High scanning velocity
- Possibility to configure the device for different resolutions of the point cloud
- Easy and robust construction
- Foveal vision for having the highest resolution in the heading direction

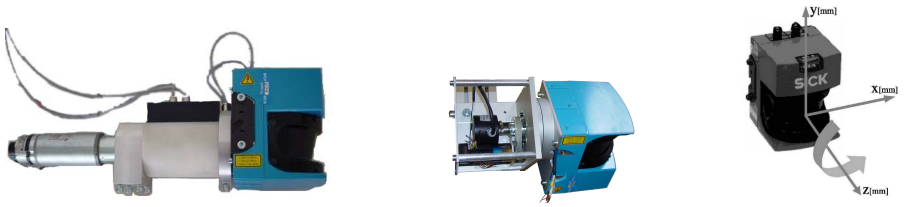
Different variations of moving 2D scanners have already been proposed. In [1], [2], [3], [4], [5] or [6] single scan lines are taken with a horizontal movement. This approach can generate a good model of the environment, but cannot avoid collisions. Depth image based approaches can be found in [7],[8], [9]. Here first steps towards autonomy or semi-autonomy are taken. The taken depth images show a high resolution in (for the robot)

uninteresting areas like the cealing [7] or the area laterally [8] of the robot. As stated above it is important for us to have the highest resolution in the heading direction of the robot, especially for collision avoidance and model building with a single sensor device. The current commercial avaible 3D scanners have a very small viewing cone, which is bad for registration and do not support foveal vision. To overcome these problems we developed a 3D sensor called RoSi (Rotating Sick) which is a common Sick LMS200 scanner rotating continuously around its optical axis. This setup results in the foveal resolution of the point cloud. To improve navigation capabilities we also detect remission values and integrate texture and colour information into the model. The remainder of the article is as follows. Chapter 2 describes the setup of the Rosi scanner. Chapter 3 shows the implemented texture integration. In chapter 4 we show some experimental results.

2. The Rosi Scanner

2.1. Hardware Setup

The RoSi scanner consists of a commercial Sick LMS 200 which is attached to a rotation axis. In contradiction to the known systems, the rotational axis lies in the middle of the scanner's field of view, and hence in the area, which is observed. To rotate the scanner we use a common DC-motor. In figure 1a the setup of the first RoSi scanner (RoSi I) is depicted. A 24V DC-motor is used with a epicyclic gear and an optical encoder. For power and data transmission a 10 channel rotary feedthrough with slip rings is used. The optical encoder has a resolution of 2000 impulses per revolution. With the 36:1 gear a theoretical rotatory resolution of $0,005°$ can be archieved. Since the motor, feedthrough and gear are oversized a smaller setup was developed therefore (1b). Here a small 24V



(a) **(b)** **(c)**

Figure 1. RoSi I (left), RoSi II (middle) and the corresponding coordinate sytem (right)

DC-motor is used with a commercial 7 channel rotary feed through. Since we use Sick LMS the depth resolution lies within a $\pm 2cm$ boundary. The angular resolution depends on the rotational velocity of the scanner, but also on the configuration of the LMS. It is possible to configure the LMS resolution to $0,25°$, $0,5°$ or $1°$. To get an angular resolution of $1°$ in both angular directions the rotational velocity v_{rot} has to be (assuming a single scan takes $t_{scan} = 15ms$)

$$v_{rot} = \frac{180°}{t_{scan} \cdot NrofDeg} = \frac{1°}{t_{scan}} \approx 67°/s$$

since the scanner acquires a full model after $NrofDeg = 180°$. In that case, a full point cloud acquisition takes

$$T = \frac{180}{v_{rot}} \approx 2,7s.$$

with a depth image consisting of about 32000 single points. It can be seen easily, that the spatial resolution in the center tends to be infinte, where the resolution gets lower towards the boundaries of the depth image, while the angular resolution is still $1°$ in both angular directions. The scanning cone is variable $100°$ or $180°$ (which results in a complete hemisphere). The maximal range is 8m (for indoor applications) or 80m (for outdoor applications). The impluses from the optical encoder are counted by a microcontroller and send to the PC via a digital I/O interface card.

2.2. Spatial Data Acquisition

With the coordinate system shown in fig 1c a horizontal scan point without rotation would have the coordinates

$$\begin{aligned} x_i &= \cos(\alpha_i) \cdot r_i \\ y_i &= 0 \\ z_i &= \sin(\alpha_i) \cdot r_i \end{aligned}$$

with α_i as scanner angle and r_i as corresponding distance. Taking into account the additional rotation around the optical axis leads to

$$\begin{aligned} x_i &= \cos(\beta) \cdot \cos(\alpha_i) \cdot r_i \\ y_i &= \sin(\beta) \cdot \cos(\alpha_i) \cdot r_i \\ z_i &= \sin(\alpha_i) \cdot r_i \end{aligned}$$

with β as the rotation angle taken after the scan. Since the scanner needs some time t_{scan} to do a single line scan, the rotation angle is variable in one scan line. Taking this into account leads to

$$\begin{aligned} x_i &= \cos\left(\beta - \left(1 - \frac{i}{MaxVal}\right) \cdot v_{rot} \cdot t_{scan}\right) \cdot \cos(\alpha_i) \cdot r_i \\ y_i &= \sin\left(\beta - \left(1 - \frac{i}{MaxVal}\right) \cdot v_{rot} \cdot t_{scan}\right) \cdot \cos(\alpha_i) \cdot r_i \\ z_i &= \sin(\alpha_i) \cdot r_i \end{aligned}$$

with $MaxVal$ as the number of maximal values per linescan (e.g. 180).

2.3. Sample Depth Images

Figure 2 shows a sample depth image with 80m maximum range and the corresponding original. Figure 3 shows several point clouds (left column) taken with different configurations. The same data triangulated as surface model (see chapter 4) is depicted in the right column. Each point cloud was acquired with a maximum range of 8m. The rotational velocity v_{rot} was kept constant.

(a) **(b)**

Figure 2. Sample with 80m range (original left, triangulated point cloud right)

3. Texture integration

The texture of a scene is mapped on the model as follows: the texture image is acquired with a digital camera, which is mounted below the RoSi sensor and has a fixed, but unknown offset and orientation relative to the scanner (see figure 4a). Therefore, a calibration between the 3D coordinate system of the RoSi sensor and the 2D coordinate system of the digital camera is necessary. In the first step the point cloud is triangulated, so that every triangle patch is filled with texture from the grabbed image. The triangulation algorithm uses known neighborship relations between the points, due to the specific architecture of the sensor system. The triangles are built by connecting two neighboring points in the same scan row ($P_{i,j}$ and $P_{i+1,j}$) with the corresponding point in the next row ($P_{i+1,j+1}$) (respectively the former row, $P_{i,j-1}$, see fig. 4b). A variable range threshold avoids the system to triangulate areas which do not belong to a reflected object. The texture is mapped on every triangle by calculating the corresponding 2D image pixel to every 3D point belonging to the triangle. Therefore a transformation from the scanner coordinate system SCS to the camera coordinate system CCS of the camera is necessary. An intermediate coordinate system ICS is used to connect these two coordinate systems. This ICS is located in the lower left front corner of a calibration object. This object is a box which contains 16 pin heads on two different levels (see fig.5a). A transformation C_S is calculated to transform the SCS to the ICS. Another transformation C_C is estimated to transform ICS to CCS. The complete transformation is

$$C_{complete} = C_C \cdot C_S.$$

As soon as the transformation matrix $C_{complete}$ is computed the calibration object is not needed anymore. Contrary to the camera transformation, the scanner transformation is determined directly from three intersecting orthogonal planes, the left side, the floor, and the back plane of the calibration object. Due to the low number of laser measurements on the pin heads and the scanner uncertainty, the pin heads are not used here.

The determination of these three planes is interactive. The user selects three points on each plane. The calibration algorithm calculates the three planes from these points. To take the scanner uncertainty into account, it uses the mean value of a number of surrounding points, which are less distant than a threshold $DMAX$. The mutual intersections between two planes determine the axes of the ICS. The origin of the ICS is de-

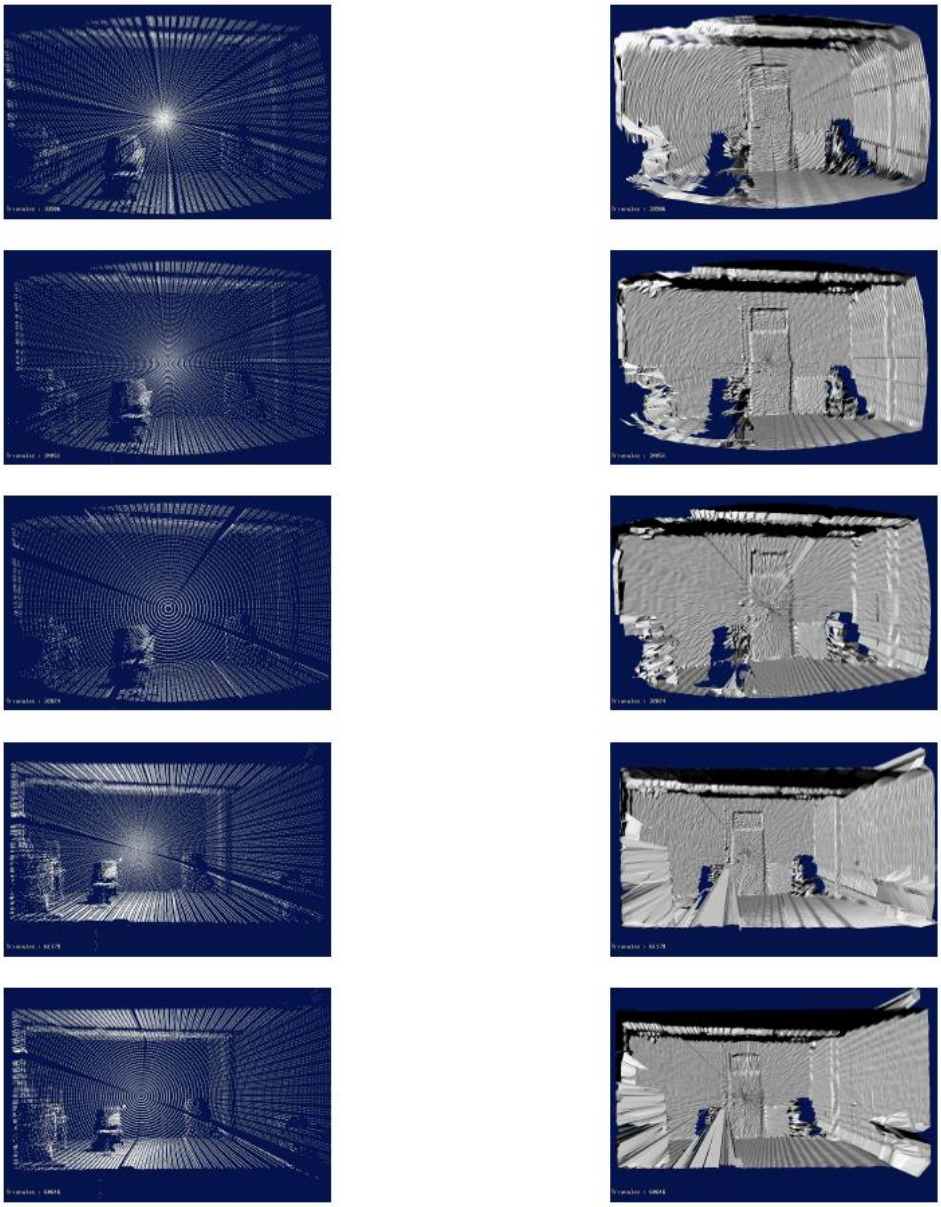
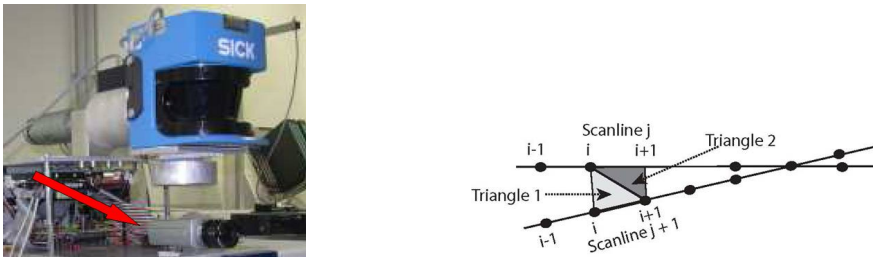


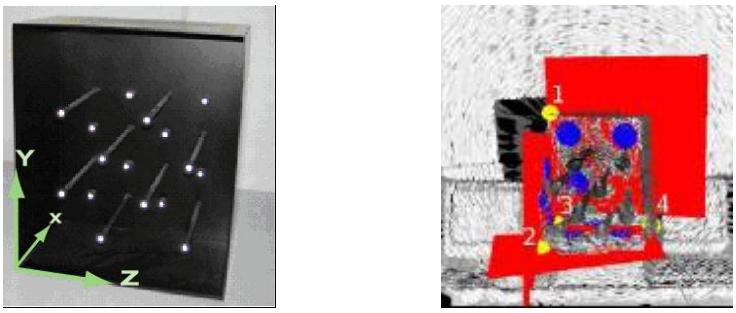
Figure 3. Depth Images with all variations in configuration concerning opening angle and scanner angular resolution (from top to bottom)$100°/0.25°, 100°/0.5°, 100°/1.0°, 180°/0.5°, 180°/1.0°$

termined by the intersection of the three planes and by the known distance of the back wall from the front corner (see figure 5b). The camera calibration matrix C_c is calculated via the 16 measurement points within the calibration object. In a first step a color image segmentation of these pin heads is performed, and a connected components labeling algorithm is executed on the resulting binary image. The image coordinates of the pin head center points make a computation of the calibration matrix C_c possible. With the



(a) **(b)**

Figure 4. The camera of the Rosi scanner (left) and a visualization of the triangulation algorithm



(a) **(b)**

Figure 5. Calibration object (left) and found planes (right)

two calibration matrices the scanner to camera transformation matrix $C_{complete}$ is given.

A triangle point can by calculated in image coordinates as

$$\vec{x}_c = C_{complete} \cdot \vec{x}_s$$

Note that if the texture coordinates $\vec{x}_c$ lie outside the field of view of the color camera FOV_{cc}, the texture of the corresponding 3D triangle point $\vec{x}_s$ will remain undefined. Also, it is possible for the transformation $C_{complete}$ to map distinct 3D points onto the same texture coordinate, where the color camera sees only the texture value $I(x_c)$ of the point closest to it (to its origin O_{cc}). The other 3D points are occluded and must therefore not be textured from this view. Therefore, the algorithm for single view texture assignment is summarized by the following formula, which is applied to all triangle points:

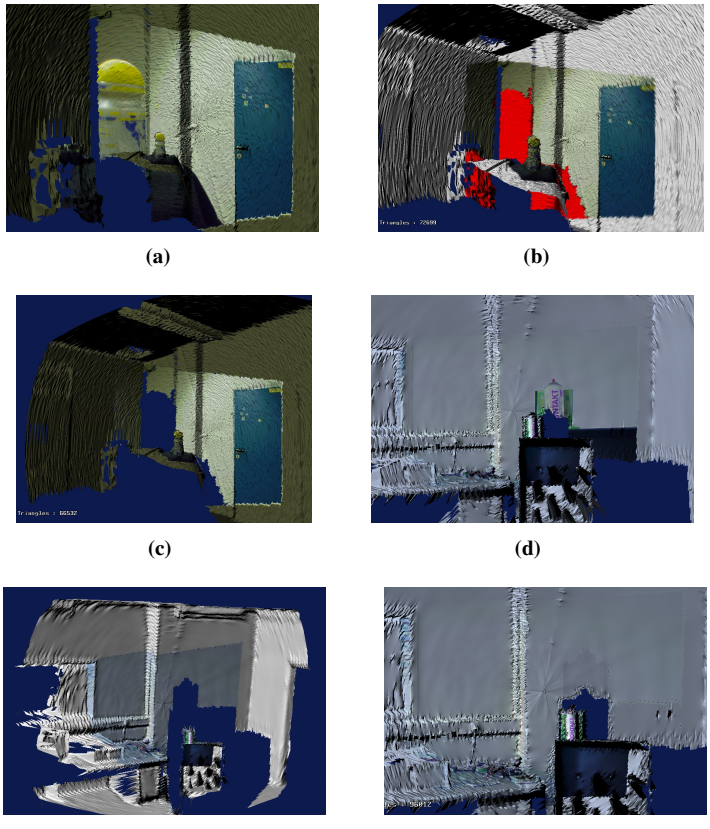
$$I(\vec{x}_s) = \begin{cases} I(\vec{x}_s) & if\ \|\vec{x}_s - O_{cc}\| = min\ \|\vec{x} - O_{cc}\| \\ undefined & else \end{cases}$$

Nevertheless, the same point $\vec{x}_s$ remaining untextured from this view may receive its texture from another view where it is not occluded through another object. To cope with this problem a method similar to [10] was implemented. Here, different memory areas

with size of the camera image are allocated to store distance information and the corresponding triangle. In this way it is not difficult to determine the closest point with a specific texture value and delete texture information from occluded areas.

4. Experimental Results

Figure 6 depicts an occlusion situation. It can be seen that the areas which lie out of the cameras view stay untextured. The bottle with the yellow cap occludes the wall. The texture of the bottle is incorrectly projected onto the wall (a). The occluded area is marked in red (b) and removed (c). Figure 6d shows another occlusion situation. The occlusion is removed (e) or an estimated texture is put on the undefined areas (f). The texture is estimated using a mean value between the last valid point and the first valid point after the occlusion.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 6. Occlusion, its detection and removal (a,b,c) and another occlusion, its removal and a estimated texture for the untextured area (d,e,f)

Acknowledgments

The work described in this paper was conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion", s. `http://www.cogniron.com`) and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

References

[1] C. Früh, *Automated 3D Model Generation for Urban Environments*, 2002, Universität Karlsruhe, phd thesis

[2] S. Thrun D. Hähnel, W. Brugard.*Learning compact 3d models of indoor and outdoor environments with a mobile robot* In IJCAI, 2001.

[3] S. Kristensen and P. Jensfelt. *Active global localisation for a moble robot using multiple hypothesis tracking*. In Proceedings of the IJCAI99 Workshop on Reasoning with Uncertainty in Robot Navigation, pages 13-22, Stockholm, Sweden, 1999.

[4] W. Burgard S. Thrun, D. Fox. *A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping*. In IEEE International Conference on Robotics and Automation, San Francisco, 2000.

[5] F. Blais J.A. Beraldin S.F. El-Hakim, P. Boulanger and G. Roth. *A mobile system for indoor 3-d mapping and positioning*. In Proceedings of the International Symposium on Real-Time Imaging and Dynamic Analysis, 1998

[6] A. Walthelm and A. M. Momlouk *Multisensoric active spatial exploration and modeling*. Dynamische Perzeption: Workshop der GI-Fachgruppe 1.0.4, pages 141Ű146, Ulm, 2000.

[7] Oliver Wulf, Bernardo Wagner, Mohamed Khalaf-Allah, *Using 3D data for Monte Carlo localization in complex indoor environments*, European Conference on mobile Robotics, ECMR, 2005, Ancona

[8] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. *Heuristic-Based Laser Scan Matching for Outdoor 6D SLAM*, in KI 2005: Advances in Artificial Intelligence. 28th Annual German Conference on AI, Proceedings Springer LNAI vol. , Koblenz, Germany, September 2005.

[9] H. Andreasson, R. Triebel and W. Burgard, *Improving Plane Extraction from 3D Data by Fusing Laser Data and Vision*, In Proc. of the International Conference on Intelligent Robots and Systems(IROS), 2005

[10] Kia Ng, et. al, *An integrated multi-sensory system for photo-realistic 3d scene reconstruction*, School of Computer Studies, University of Leeds and European Commission - Joint Research Centre.

SURFACE-SEGMENTING MODIFIED BALL-PIVOTING-ALGORITHM

Andrés Restrepo Specht *Michel Devy*

LAAS-CNRS
7, avenue du Colonel Roche
31077 Toulouse, France
E-mail: {arestrep, michel}@laas.fr

ABSTRACT

Ball Pivoting Algorithm (BPA) was originally made for computing a triangular mesh from a regular distributed cloud of points after a registration process. In this article this method is presented and adapted for the mesh creation of a non-regular distributed surface of points and the simultaneous surface segmentation. Results are obtained from 3D data acquired by several sensors: laser range finder, stereo or profilometer.

1. INTRODUCTION

In the last years, a lot of work has been made on 3D object- or environments modelization. The applications that need these type of models are numerous, in addition after the developments in virtual reality (remote controlled processes, video games, virtual visit of touristic sites or museums, building or town modelization). Our interest is mostly centered in robotics.

We have presented before some works dealing with the 3D modelization tasks, registration [1, 2] and the incremental construction of triangular meshes [3]. In this article, we suppose that the problem of registration is solved and we are interested in the mesh construction or the set of surface primitives in a 3D cloud of points. Therefore, we propose to adapt the algorithm given by G.Taubin et al. the *Ball Pivoting Algorithm* (BPA) [4]. The original method treats the construction of triangular meshes. Our BPA adaptation intends to make the algorithm more robust to registration errors and measure noise and treat the surface segmentation at the same time as the mesh is created, taking advantage of the "region growing" behavior of the algorithm.

This article is composed of 4 sections: the section 2 presents a short state of the art in the mesh construction methods. The sections 3 and 4 are devoted to the description of the method and the result presentation on different types of 3D images. At the end, in section 5 we will present the conclusions and evoke our actual works.

2. STATE OF THE ART

The technological advances in 3D data acquisition systems have motivated the conception of different methods for object reconstruction. These methods are adapted to the application properties (type of objects, resolution, 3D model precision) and to the acquisition methods. An object or an environment can be represented in many forms, but triangular meshes were imposed thanks to their generality. The incremental construction techniques for triangular meshes, based on 3D points, can be classified in two big categories:

Iterative subdivision-based methods: they use typically the Delaunay triangulation, which gives a theoretical guarantee for the final mesh quality. The disadvantages are the amount of memory needed and the computational time and also the numerical instability.

Region growing-based methods: the principal advantage is the robustness to acquisition errors, but on the other side these methods need a seed triangle to start with the surface creation process.

More explicitly, Hoppe et DeRose [5] have proposed a subdividing method, considering the distance function between each point and a tangent surface, while Bernardini et Bajaj [6] use the distance function in the alpha form of the cloud of points. Attali [7] make the surface reconstruction using a subset of the Delaunay diagram. Boissonnat et Cazals [8] applied a hybrid technique, with points and surface normals, calculating the surface as a set of zeros of a function, which gives a natural neighborhood relation between points. In [3], we deal with the incremental mesh construction using *Split* techniques; one of the mayor difficulties consists in the detection of common parts in the actual mesh, created with the preceding images, and the local mesh constructed with the actual image, taking into account the possible occultations.

The following works made similar choices in the method presented in this article. Fisher, Fitzgibbon et Eggert [9] perform a point cloud tessellation according to the Hoppe et DeRose method, then execute a classification of the local geometric form of the object to be modelized, in order to adapt the model construction by region growing. Bernardini, Mittleman, Rushmeier, Silva et Taubin [4] developed the BPA triangulation method; it's an iterative model based on alpha forms, allowing to merge in a mesh previously registered data. This is the method that we have adopted, since it can handle a great number 3D points with little calculation cost. We have introduced two modifications: 1) Treat the 3D images incrementally, therefore merge each image after being acquired and registered with the mesh in construction. 2) Simultaneous segmentation of the model in a set of surface primitives (here planes).

3. BPA ALGORITHM DESCRIPTION

The BPA principle consists in: the process starts with a *seed* triangle, whose three points belong to a sphere with radius ρ chosen by the user; see figure 1. This ball pivots around a triangle edge, always maintaining contact with the two vertices of the pivot edge, until it finds another point of measure, used to form a new trian-

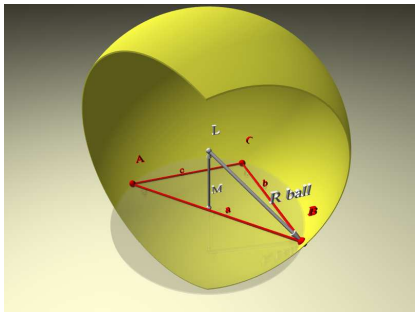


Fig. 1. The pivot ball with radius ρ and center L, lay on a triangle ABC. M is the center of the circumscribed circle to ABC.

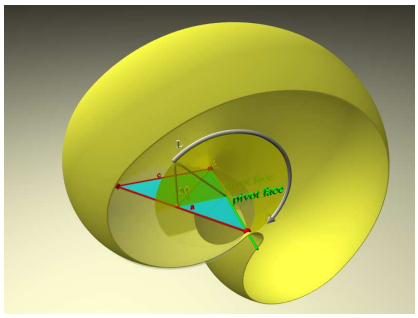


Fig. 2. The pivot ball around the BC axis, which is the designated pivot edge.

gle (figure 2), or if no point found, the edge will be marked as a frontier edge of the surface. The process will repeat with an edge of the new triangle or on a free edge of another already created one. If all edges have been analyzed, a new seed triangle must be selected between the not utilized measure points and the process can start again. The process stops if all edges have been checked and all points have been analyzed.

The original method [4] was modified to segment simultaneously the surfaces and to be more robust for variable resolution images. Some term definitions to be used are:

seed triangle: it is the triangle from which the surface mesh is generated.

pivot triangle: is the triangle used by the BPA to look for new triangles. The first pivot triangle is the seed triangle; in general, the created triangle of the previous iteration is the actual pivot triangle.

pivot edge: is the edge of the pivot triangle, around which the ball is turning to find new triangles.

frontier edge: it's an edge without any adjacent triangle, so it's on the frontier of the surface.

3.1. Initialization: seed triangle selection

A critical step in the process is to find a good seed triangle to start the method, once the pivot ball radius is given. The selection of the radius must be adapted to the point cloud resolution to be meshed, on the other side this radius will set the quality of the built model. If the radius is too small, the ball can't include the seed triangle, or it can't find any measure points pivoting around the available edges, which will turn to be frontier edges of the surface. The model will not be continuous and will have many holes. On the other side, if it's too large, the ball will find too much points; but only one can be chosen to construct the next triangle, while the rest will be marked as invalid. As the chosen point must be in contact with the pivot ball, the distances between the selected points will be about the ball radius, therefore the final model will be continuous but the form will not be as faithful as the original cloud of points.

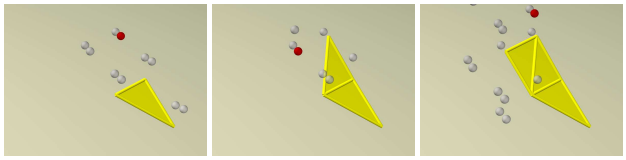


Fig. 3. The triangle construction step by step: (left) seed triangle and candidate points found by the ball rotation around the pivot edge; the dark point is the selected point. (center) second triangle, (right) third triangle.

3.2. The triangulation process

The triangulation process for a surface starts with the seed triangle and continues until there are no more integrable points on this surface. If free points remain, a new seed triangle will be selected and a second surface will be created, until the end of all free points. The figure 3 illustrates the creation of the three first triangles of a surface. The steps are explained below.

3.2.1. Candidate point exploration

Starting with a pivot edge, a base task consists in finding the candidate points for the vertex of the next triangle in the cloud of points. The point scan routine takes advantage of a *voxel map*, with a resolution proportional to ρ, to optimize the CPU time; only the voxels around the pivot triangle are analyzed.

The point selection criteria are applied in the following classification steps. At first, all the points that are inside the pivot ball trajectory around the pivot edge are selected. Afterwards, for every candidate point that would form a triangle with the two points of the pivot edge, the normal, area and angles are calculated. The mean normal of the surface is used to found the candidates on the surface. At last, for the candidate points that made it through this step, the formed candidate triangle is checked for intersection with other existent triangles.

At the end, we have a set of free and frontier candidate points from which the best two ones have to be picked. This best candidate selection is based on a triangle quality test, that depends of form, area and normal of the triangle.

3.2.2. Selection of the pivot edge

In the most cases, the pivot triangle has two frontier and one occupied edges, being the last one the pivot edge of the preceding iteration, edge which is in contact with the surface in construction. These two edges are checked out to pick up the best to be the next pivot edge.

It starts to look on the edge that is oriented in the same manner as the edge of the precedent iteration. After examination of the candidate points of the edge 1, if there is another edge available,

it will be scanned only if no points were found on edge 1 or if there are no frontier points between the candidates of edge 1. Once the second edge has been analyzed, the selection between the two edges depends from the frontier points found in each edge.

In the worst case, for each edge, the candidate points selection will give two candidate points: one free point and one frontier point. Therefore, for a "normal" pivot triangle with two frontier edges , we will have a pair of best candidates, 4 points in total: 2 free and two frontier points, which now will be compared to choose the actual pivot edge.

This treatment seems complex, but it creates a spiral-formed mesh growing around the seed triangle, which permits to reduce the hole creation. The frontier point is generally preferred in the selection, even if the resulting triangle has lower quality.

3.2.3. New triangle creation

The selected point forms with the pivot edge vertices a new triangle, which will become the next pivot triangle. The *new pivot edge* has by default the same reference as the previous one. If the edge is marked as not-free, it must be changed. If the new triangle has no frontier edges, no growing is possible, then a new pivot triangle on the same surface has to be chosen from the list of created triangles, looking for free edges. If there are no more free edges, the process stops for this surface. Another seed triangle must be chosen from the free points. The process ends if there are no more free points available.

3.3. Pseudo-code of our BPA method.

```
TS=Seed_Triangle_Selection();
pivot_edge=search_edge(TS);
create_surface();
while(TS != 0) {
  No_candidates = Search_points(pivot_edge);
  if(No_candidates != 0)    { Select_candidat();
                              Add_triangle(TP);
                              update_structure();
                              pivot_edge=search_edge(TP); }
  if( (pivot_edge == 0) || (No_candidates == 0) ) {
      TP=Select_pivot_triangle();
      if(TP == 0) { close_surface();
                    TS=Seed_Triangle_Selection();
                    pivot_edge=search_edge(TS);
                    create_surface(); } } }
```

3.4. Incremental mesh construction or improvement.

The algorithm has been modified so that it is possible for the user to start the process with an existing mesh. This aspect is also useful for the creation of a variable resolution mesh, a mesh made with patches created by different radius pivot balls. It helps to solve some intrinsic problems of the algorithm philosophy, more precisely the tendency to create holes. The BPA algorithm is applied iteratively starting from an existing mesh, selecting a frontier edge as the pivot axe and using a greater ball radius in the hope to "catch" an already created triangle vertex or a free point, in order to close a hole.

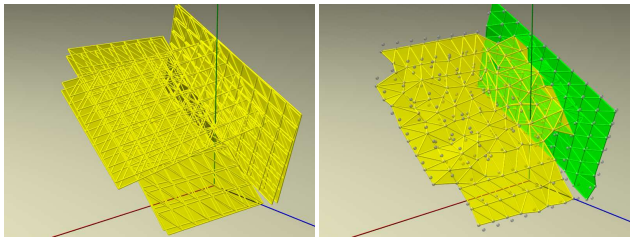


Fig. 4. Result of BPA with segmentation (right) on a synthetic cloud composed of six views of two orthogonal planes (left)

4. RESULTS AND DISCUSSION

The BPA have been tested on different types of data, with or without polygonal meshes, only points and also on a set of post-ICP surfaces:

Synthetic Images: we have generated synthetic polygonal surfaces with uniform distributed points, in order to evaluate the basic behavior of the BPA algorithm. The measure and registration errors has been simulated mounting multiple views together. Every time in these examples, the BPA use the local mesh of the different views to look for the seed triangle. The figure 4 present six views of the two orthogonal planes. They are also lightly mounted one over the other, again, in order to evaluate the segmentation properties of BPA. The two expected surfaces has been detected.

Mesh after an ICP registration: The results of a registration process with ICP have been used to evaluate the BPA algorithm behavior producing a new mesh after an ICP. We show here a synthetic and a real example. In the figure 5, 9 synthetic meshed laser sensor views of a geometric object has been used. All 9 views have been previously decimated to reduce the number points. Afterwards the images has been registered with ICP in pairs (introducing even more errors into the registering process, being an incremental registration preferable) forming the complete cloud of points. The final ICP result gives a non uniform distributed cloud of points with overlapping meshes. The BPA detected successfully every object surface and took the needed points for the new mesh. The figure 6 shows a cloud composed of two 3D images, which are dense stereo reconstructions. In the same manner, these images have been decimated before the registration with ICP. The BPA has tried to produce a new mesh and simultaneously a segmentation in surface primitives. The point distribution of this type of image is locally dense, but not uniform, in particular on the soil. Therefore the ball radius has to be chosen greater than the distance between most of the neighbor pairs, so to reduce the number of holes.

Profilometry images: Images from a profilometry for human body modelization have been used: The profilometry acquisition method gives as output sequences of circular horizontal point scanlines, where the horizontal distances between scan points inside a scanline and the vertical distances to the other scanlines are different. There is no local mesh available, therefore the seed triangle for BPA is extracted from the points of the cloud.

4.1. Performance

The CPU times for the the chosen examples are given for a Pentium III 933MHz; the results are presented in the table 1.

Example	Points	Initial Polygons	Ball radius	Final Polygons	Segments	CPU time [s]
Stereo	8561	15998	0.31	5250	6	138.00
geometric object	6362	10838	0.3	1919	9	86.00
profilometry 1	9356	0	7	4247	24	76.00
profilometry 2	5952	0	7	3428	16	64.00
synth-6planes	480	756	0.47	348	2	2.00
synth-2planes	320	504	0.47	135	1	0.92

Table 1. Results of the BPA process on different types of data

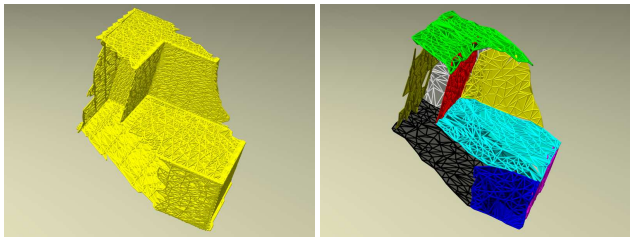


Fig. 5. BPA results (right), with construction of a mesh and segmentation, on 9 registered views acquired from a polyhedric object (left). The faces given by BPA are illustrated in false color.

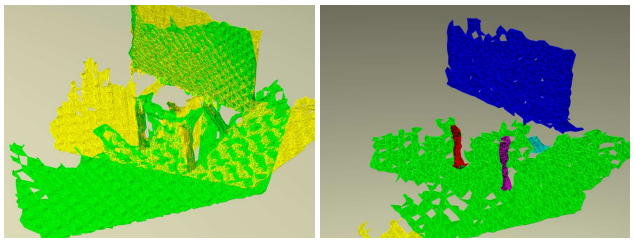


Fig. 6. BPA results (right), with construction of a mesh and segmentation, on 2 registered views acquired on a scene with dense stereo (left).

5. CONCLUSIONS

The triangulation algorithm behaved as expected in [4], for the regular distributed cloud of points, found typically in laser sensor and profilometry images (*Bunny* sequences, body modelization). For variable distributed clouds of clouds (images acquired with stereovision and those resulting from an ICP process), our proposed modifications on the method gave good results.

The segmentation in surface primitives, in the moment planes and regular curved surfaces, gave acceptable results for structured scenes.

As observed in the tests on the images, we can conclude that our adapted version of the BPA algorithm is robust.

At last, like we have done in our precedent works [3], the algorithm can be optimized using information given by the 3D edge extraction, so as to impose the vertex presence over the discontinuities of the scene to be modelized. The edges can be extracted very efficiently either analyzing line by line and column by column each acquired 3D image on the scene [10, 11] , or exploiting a stereovision image by segment pair matching in order to extract the 3D segments.

6. REFERENCES

[1] A. Sappa, A. Restrepo Specht, and M. Devy, "Range image registration by using an edge-based representation," in *Proc. 9th International Symposium on Intelligent Robotic Systems (SIRS'2001), Toulouse (France)*, July 2001.

[2] M. Devy, A. Restrepo Specht, and A. Sappa, "Comparaison de méthodes de recalage de données 3d par construction d'un maillage triangulaire ou extraction des contours," in *13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle (RFIA'2002), Angers (France)*, Jan. 2002.

[3] J. Bozier, M. Devy, and A. Sappa, "A geometrical approach for the incremental modelling of free form surface by triangular meshes," in *8th Int. Symp. on Intelligent Robotic Systems (SIRS'2000), Reading (UK) , pp.13-21, Rapport LAAS N.00332*, July 2000.

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[5] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[6] C. L. Bajaj, F. Bernardini, and G. Xu, "Automatic reconstruction of surfaces and scalar fields from 3D scans," *Computer Graphics*, vol. 29, pp. 109–118, 1995.

[7] D. Attali, "r-regular shape reconstruction from unorganized points," in *Symposium on Computational Geometry*, 1997, pp. 248–253.

[8] J.D. Boissonnat and F. Cazals, "Smooth surface reconstruction via natural neighbour interpolation of distance functions," in *Symposium on Computational Geometry*, 2000, pp. 223–232.

[9] R. Fisher, A. Fitzgibbon, and D. Eggert, "Extracting surface patches from complete range descriptions," *Proc. International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa*, 1997.

[10] A. Sappa and M. Devy, "Fast range image segmentation by an edge detection strategy," in *Proc. Third International Conference on 3D Digital Imaging and Modeling, Quebec (Canada)*, May 2001.

[11] X. Jiang and H. Bunke, "Edge detection in range images based on scan line approximation," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 183–199, Feb. 1999.

Grasp Planning for Non-Convex Objects

Efrain Lopez-Damian[†], Daniel Sidobre[†‡], and Rachid Alami[†]
(edamian, daniel, rachid)@laas.fr

[†] LAAS-CNRS
7, avenue du Colonel Roche, 31077 Toulouse, France

[‡] Université Paul Sabatier
118, route de Narbonne, 31062 Toulouse, France

Robotics research aims also to develop autonomous systems in dynamically changing environments. One desirable functionality is manipulation that allows the robot to modify its surroundings. Generally, a grasp is the beginning of any manipulation task and robots must be capable to handle most common objects. The shape of these objects is varied and many are non-convex which make difficult to grasp them. This work presents a grasp planner that deals with this problem using an approximate convex decomposition of the object to plan grasps.

1. INTRODUCTION

In the field of autonomous robotics, one kind of system that interest researchers today is service or personal robot. Such machines must have the capabilities to interact and make decisions in an automatic way depending on the constraint imposed by the always changing environment and the task to do. To reach this goal it is necessary to develop a set of functionalities. Among all the functions, the manipulation of objects has a decisive role to allow the robot to interact and modify its environment.

A grasp is generally the beginning of any manipulation task and robots must be capable to handle most common objects in the surroundings. The shape of these objects is varied and many are non-convex which difficult the grasp planning. The object model can be obtained by a stereovision or 3D laser sensors. These models are generally complex and composed by many details. Grasping an unknown object is one of the tasks that a personal robot must be able to accomplish. A clue point to solve is the automatic grasp of the object. In this paper we present a grasp planner for complex objects and more particularly the use of an approximate convex decomposition.

2. RELATED WORK

In literature, we can find work that propose algorithms for grasp planning of polygonal objects [12]. The grasp planning is converted into an optimization problem. In ref. [6] the planning for 2D curved objects is proposed, an antipodal grasps are computed in [5] for arbitrary shape objects. The problem becomes much more difficult in three-dimensional space. For polyhedral objects, work has been done to characterizes and compute three and four-finger grasps [13].

Based on the hypothesis that instead of compute optimal grasps a set of ranked grasps can be generated using a quality criteria, a good grasp can be chosen among grasps in a set [3]. The use of heuristics accelerate the process to find a grasp on the object. Borst [2], [8] uses a random generation strategy, they define an arbitrary point and frame inside the object. We have proposed a semi-random approach in ref. [11], where generation of grasps is guided by the mass center and inertial axis of the object.

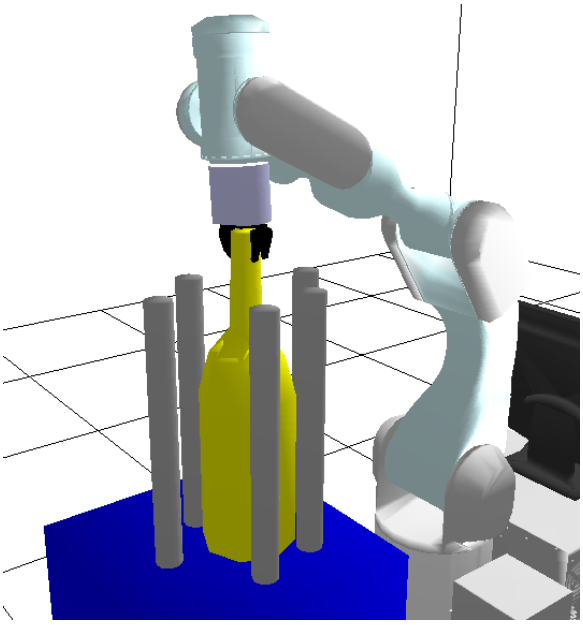


Fig. 1. Example where a bottle is surrounded by cylindrical obstacles, a grasp is found near the bottle neck.

3. GRASP PLANNING

We can define a grasp as the end-effector placement relative to the object and the contact points on the object surface. Then the grasp planning problem becomes the generation of feasible grasps on the object that satisfy some constraints. A grasp planner must produce a grasp that fulfill some fundamental criteria: the grasp must be stable, the grasp must be reachable and the grasp must be free

of collision. Grasp planning is computationally intensive due to both the large search space and the incorporation of geometric and task-oriented constraints. The search space for an optimal grasp can be reduced by using certain heuristics. Shape of object, relative size of object and gripper are the principal geometric characteristics. Shapes are generally non-convex and constitute a more difficult problem to solve for a grasp planner. In next sections we describe a planner to deal with such kind of objects.

4. NON-CONVEX OBJECT GRASP PLANNER

The approach that we proposed, firstly try to generate grasps on the whole object. Secondly, decompose the object in smaller components and compute a set of grasps for each component generated. Every grasp is ranked depending on a quality criterion, the grasp with best quality is chosen as the output of the planner.

We present here the basic algorithm framework of the semi-random generator of grasps for non-convex objects. The grasp planner is based on the work done in [11].

Algorithm Framework

*loop:*object decomposition
for each component
1. Compute inertial axis from component
2. Semi-random generation of grasps
3. Apply filters
4. Assign a quality value
5. Choose the best feasible grasp
if grasp founded
then Stop planner and output grasp
end if
end for
end loop

The grasp planner take as an input the geometric model of the environment, including the object and the geometric and kinematics models of the robot and gripper. The object decomposition process is the first step of the algorithm, here two strategies for the planner can be followed, call a complete decomposition process that gives as a result the series of components of the object and generate a set of grasps for each component in the list or as we present in the framework, at each iteration of the decomposition process two components of the input are produced, we choose one of the two components and we try to generate a feasible grasp on it. If a grasp is founded the planning process is terminated and the grasp is given as output. If not we take the second component and we call again the grasp planner. In case that a feasible grasp cannot be generated for both components a new iteration is performed.

This process is repeated until one of two conditions arrives: a grasp is founded in one component or decomposition is terminated. A feasible grasp can be found before complete decomposition. The last components found are generally small and less interesting. Next we give a briefly description for each part of the grasp planner. A more detailed description of certains parts can be founded in [11]. The description of the object decomposition is given in next section.

Inertial Axes: the location of the center of mass and the inertia tensor can be computed by the conversion of the integrals of mass into the volume integrals. We suppose the polyhedron (P) has a mass m and a uniform density ρ, we can relate the volume as $m = \rho V$.

Semi-random Generation: First we define a frame at or near the inertial frames and secondly we compute contact points according to the gripper used. A method for a gripper with two fingers and three contact points is presented in [11]. A specialized grasp generator is needed for each different gripper to compute the contact points from the grasp frame position.

Filter: As quality determination is computationally expensive, we introduce the filter step to reject as soon as possible unfeasible grasps. Some constraints are imposed by the system itself, the grasp must be kinematically reachable by the robot and free of collision with the possible obstacles in the environment, simulation tools are used for this [14]. To guarantee that the object is firmly held with no slippage, we use a force closure filter.

Quality Measure: Several grasps can be produced after the first two steps are executed. The final step is the assignment of a quality measure to the grasps.

4.1 Force-closure filter

One of the most important properties for a grasp is the notion of force closure. A grasp is defined geometrically

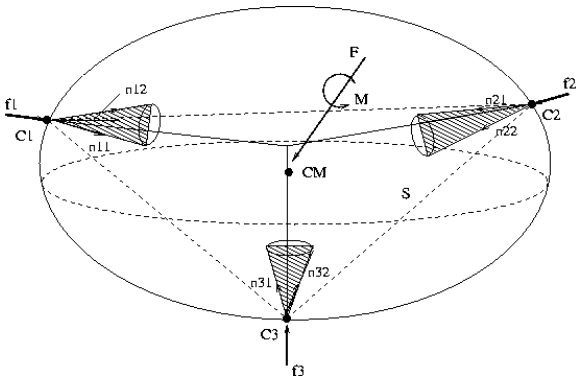


Fig. 2. 3D Grasp with three contact points can be see as a plane grasp considering the sections of friction cones by the plane containing the three points.

by the position C_i of d hard fingers or contact points on the object surface, with $i = 1, ..., d$. Hard finger contact model and Coulomb friction are assumed between the object and the fingers. Each finger exerts in C_i a force f_i and a moment $C_i \times f_i$ with respect with some point on the object,

the center of mass in this case. Force and moments are combined and form a six-dimensional vector called wrench $w_i = [f_i, C_i \times f_i]^T$. A grasp achieves equilibrium when the sum of the wrenches is zero $\sum_{i=1}^{d} w_i = 0$. A grasp is force closure if it can balance any external forces and moments (w), exerted on the object

$$w + \sum_{i=1}^{d} w_i = 0 \tag{1}$$

The forces applied by the finger f_i must remains in the friction cone to avoid the slippage, see Fig. 2. The grasp then is force-closure if and only if there exists a force in each friction cone such that the sum of the corresponding wrenches is zero. For a three-finger grasp as in Fig. 2, a necessary condition for force closure is that there exists a point in the intersection of the plane formed by the three contact points with the friction cones at these points [13] [9].

4.2 Quality Measure

The planning of a good grasp is important when the robot has to take an object in a firmly way, for this a quality criterion has been developed in [7]. The criterion try to quantify the notion of a good grasp for a force closure grasp. Again a hard finger contact model and Coulomb friction are assumed. We must discretize the cone of friction to represent it by a finite set of *m* vectors.

For the quality measure, Ferrari [7] consider that the sum of the magnitude of the forces applied by the gripper at the *n* contact point is 1, then f_i can be written as:

$$f_i = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{i,j} f_{i,j} \tag{2}$$

with $\alpha_{i,j} \geq 0$. Similarly we have that the total wrench applied on the object is expressed by:

$$w = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{i,j} w_{i,j} \tag{3}$$

and the set of all wrenches is:

$$W = CHULL(\bigcup_{i=1}^{n} w_{i,1}, ..., w_{i,m}) \tag{4}$$

The quality measure is the distance of the nearest facet of the Convex Hull from the origin.

Other criteria can be added, for example to try to favor grasps that contact points are in the middle of the facets and defines a more stable grasps.

5. OBJECT DECOMPOSITION

Sometimes the grasp planner doesn't find any valid grasp, caused by several reasons. Size and geometry complexity of the object are the most common reasons for the planner to fail. One possible solution is to decompose the object in several smaller parts.

Convex polyhedra decomposition is not new, in the computational geometry community a solution is given for convex decomposition [1] [4]. The algorithms make a partition of the model into convex components. Two main problems arise in the use of these kind of algorithms: an unmanageable number of parts and the tiny size of these parts. Both results are undesirable in grasp planning because of the excessive computing time and the impossibility to grasp tiny parts.

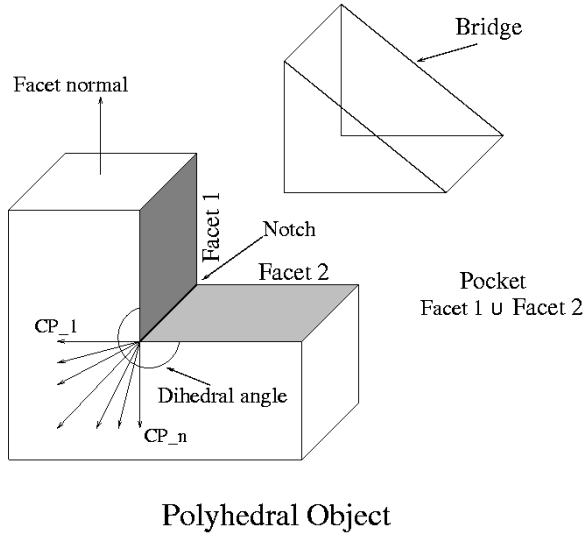


Fig. 3. Example of Polyhedral object, bridges and pockets are indicated. The notch is the reflex edge formed by the two facets with an internal angle greater than $180°$. A series of cutting planes CP_1 ...CP_n are shown.

An alternative approach is the use of an approximate convex decomposition(ACD) [10] to partition the object in approximate convex pieces.

5.1 Approximate convex decomposition

The ACD takes as an input the geometric model of a component, this can be the object or a part of the object, represented by a polyhedron P_H and a tunable convex tolerance λ.

The approximate convex decomposition removes at each iteration the most significant non-convex feature called notch (a reflex edge where the inner dihedral angle subtended by two incident facets is greater than $180°$) by defining a cutting plane and partitioning the object in exactly two components. The resulting components are the

input for the next iteration of the algorithm. The process is repeated until all components C_i satisfy the convex tolerance λ. The definition of a λ-approximate convex component is a polyhedron whose concavity is at most λ. The convex decomposition is defined as:

$$CD(P_H) = \{C_i \subset P_H | concavity(C_i) \leq \lambda\} \; with \\ \cup_{i=1}^{n} C_i = P_H and \; C_i \cap C_j = \emptyset, \forall_{i,j} \; i \neq j \quad (5)$$

For remove the most significant notch we need a measure of its concavity. In contrast of some measures as radius, area and volume, concavity is not a well defined measure. Here, we use the concavity as a distance from the concave features to the convex hull of the object. To measure concavity, the notions of bridges and pockets of polyhedron are used, see Fig. 3.

$$Bridge(P_H) = \{f_{C_H} | f_{C_H} \in C_H \wedge f_{C_H} \backslash S_{P_H} \neq 0 \\ Pocket(P_H) = \{f_{C_H} | f_{C_H} \in S_{P_H} \wedge f_{C_H} \backslash C_H \neq 0 \quad (6)$$

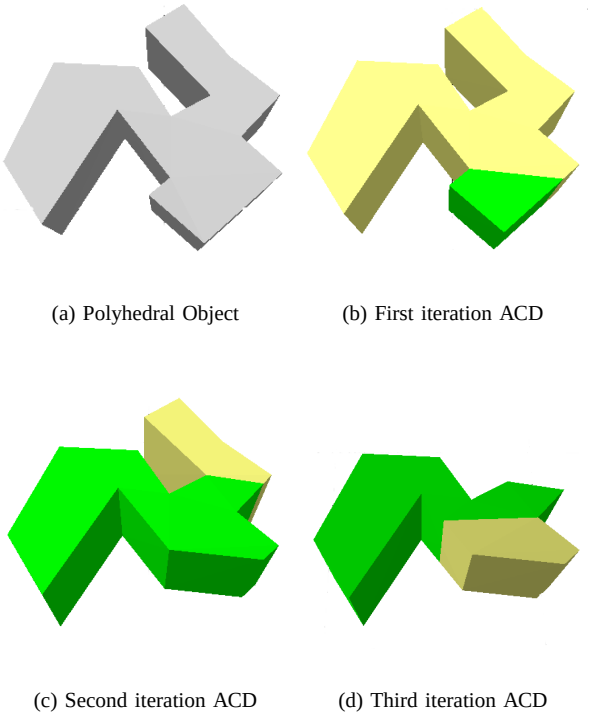
where, C_H is the convex hull of the polyhedron, f represents a facet and S_{P_H} is the surface of the polyhedron P_H composed by a set of facets. The concavity measure is then the distance between the middle point of each notch to the bridge.

The bridges are facets of the C_H that are not part of polyhedron. The algorithm identify the match between facets of C_H and P_H and discard them. Pockets are facets of polyhedron P_H and they are not part of the convex hull. As the facets of C_H are not generated from the facets of P_H, it is to say, if P_H would be convex its correspondent C_H will be the same P_H, a test is required to find the facets that correspond to pockets. Such test is done by a user tolerance in the distance between the P_H facet and the C_H facet and a tolerance in the orientation between facets planes, after test we obtain the pockets.

When bridges and pockets are computed and a concavity measure is assigned to each notch. The notch with the highest concavity is selected and a series of cutting-planes (C_p) are formed to remove the notch. We select one C_p, taking as criterion that the cut has the smallest surface. Once the cut is made, we generate the set of facets that are on the C_p and we build the two polyhedra.

6. RESULTS

We present the first results of the non-convex grasp planner. In Fig. 5 we can observe the inertial axes of a mug, that were used to generate the grasps on the object. These axes do not give any valid grasp. After the first call to the object decomposition process, the grasp planner found one feasible grasp on one of the generated component. We test the algorithm with a glass, see Fig. 6. A bottle object is used as third example, we can see the bottle size is big with respect of the gripper, and the only possibility is to grasp the bottle neck, see Fig. 7. Finally in Fig. 1, we



(a) Polyhedral Object (b) First iteration ACD

(c) Second iteration ACD (d) Third iteration ACD

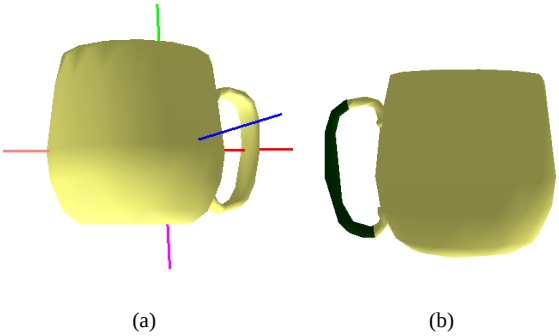
Fig. 4. The figure shows the output of the ACD algorithm. After three iterations, after first iteration is done, two components are produced. Second iteration taking as input one component of the first iteration generates two more subcomponents, the second component result of the first iteration is convex and it is not decomposed

can see how the planner is capable to compute a different grasp when the object is surrounded by obstacles. The grasp planner was implemented within the motion planner tool Move3D developed at LAAS-CNRS. Move3D counts with a series of motion planners, collision checkers and steering methods for non-holonomic car-like robots. More detailed description of the tool can be found in [14].

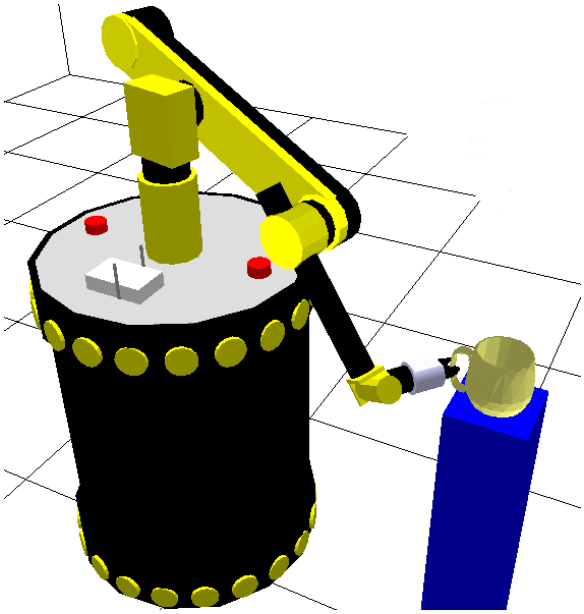
The experiments were performed using a 500 MHz Solaris SunBlade. The processing time that the algorithm requires to decompose an object is determined by the complexity of the object model. In Table 1 we can see the time after one iteration of the object decomposition algorithm for different object models.

Table 1 Results for Several Object Models

Object	Time	Facets
Polyhedral object	0.08 s	60
Bottle	0.12 s	80
Mug	0.71 s	499
Glass	13.58 s	2750

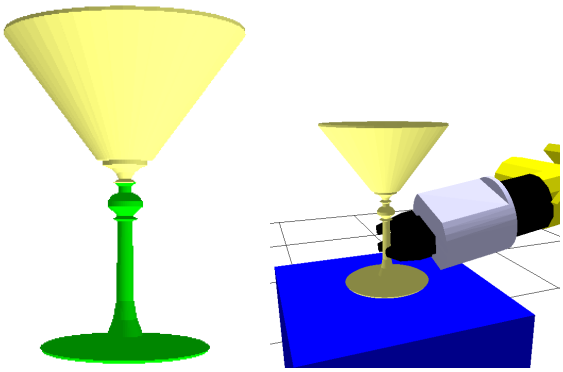


(a) (b)



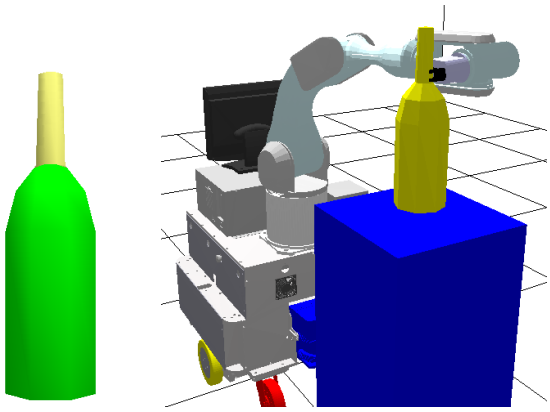
(c)

Fig. 5. a) Inertial axes used in grasp planner to generate a set of grasps on the object b) Decomposition of object after one iteration of the process c) The final grasp founded by the grasp planner after one iteration in the decomposition process



(a) (b)

Fig. 6. a) Decomposition of glass after one iteration of the process b) Feasible grasp generated after object decomposition



(a) (b)

Fig. 7. a) Decomposition of a bottle b) Grasp founded on the upper part of the bottle because bottom part is too big. As the quality criterion is global, the grasp computed is in the bottom of the bottle neck.

7. CONCLUSION

The grasp planning for non-convex 3D objects is a challenge for service robotics. In this paper we have proposed a grasp planner for these kind of objects. The idea to decompose the object in smaller parts seems to be a good solution. Strategy to generate grasps on components at each iteration of the decomposition process for the grasp planning allows to save computing time in the execution of the algorithm. The decomposition by removing concavities gives interesting results but a main point that would help to improve the algorithm is the selection of the cutting plane, until now the criterion used does not guarantee the best way to partition an object for grasping, a more intentional manner has to be found.

8. ACKNOWLEDGMENT

The work described in this paper was partially conducted within the EU Integrated Project COGNIRON (”The Cognitive Companion”) and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020. The authors acknowledge the support of the National Council of Science and Technology of Mexico (CONACyT).

9. REFERENCES

[1] C.L. Bajaj and T.K. Dey. Convex decomposition of polyhedra and robustness. *SIAM Journal of Computing*, 13(3):488–507, 1984.

[2] Ch. Borst, M. Fischer, and G. Hirzinger. A fast and robust grasp planner for arbitrary 3d objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1890–1896, Detroit, Michigan, May 1999.

[3] Ch. Borst, M. Fischer, and G. Hirzinger. Grasping the dice by dicing the grasp. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3692–3697, Las Vegas, Nevada, October 2003.

[4] B. Chazelle. Convex partitions of polyhedra: A lower bound and worst-case optimal algorithm. *SIAM Journal of Computing*, 13(3):488–507, 1984.

[5] I.M. Chen and J.W. Burdick. Finding antipodal point grasps on irregularly shaped objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2278–2283, Nice, France, May 1992.

[6] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2d objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 424–429, Sacramento, California, April 1991.

[7] C. Ferrari and J. Canny. Planning optimal grasps. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2290–2295, Nice, France, May 1991.

[8] M. Fischer and G. Hirzinger. Fast planning of precision grasps for 3d objects. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 120–126, Grenoble, 1997.

[9] J.W. Li, H. Liu, and H.G. Cai. On computing three-finger force-closure grasps of 2-d and 3-d objects. *In IEEE Transactions on Robotics and Automation*, 19(1):155–161, February 2003.

[10] JM. Lien and N. Amato. Approximate convex decomposition of polyhedra. Technical Report TR05-001, PARASOL LAB, Texas A&M University, January 2005.

[11] E. Lopez-Damian, D. Sidobre, and R. Alami. A grasp planner based on inertial properties. In *Proc. IEEE International Conference on Robotics and Automation*, pages 766–771, Barcelona, April 2005.

[12] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *In IEEE Transactions on Robotics and Automation*, 11(6):868–881, December 1995.

[13] J. Ponce, S. Sullivan, J.D. Boissonnat, and J.P. Merlet. On characterizing and computing three- and four-finger force-closure grasps of polyhedral objects. In *Proc. IEEE International Conference on Robotics and Automation*, pages 821–827, 1993.

[14] T. Siméon, J-P. Laumond, and F. Lamiraux. Move3d: A generic platform for motion planning. In *Proc. 4th International Symposium on Assembly and Task Planning (ISATP'2001)*, 2001.