Visual Context Classification for an Autonomous Robot

Lennart Quispel

August 1998

TCW 968

Supervised by:

dipl.ing. Thomas Bergener
Institut für Neuroinformatik
Lehrstuhl für Theoretische Biologie
Ruhr Universität Bochum
Germany

dr. H.A.K.Mastebroek
Department of Cognitive Science and Engineering &
Department of Neurobiophysics
University of Groningen
The Netherlands

Abstract

A system is described that is able to classify visual contexts. It can be used in a supporting role, to help the navigation of a robot, help object recognition, or to impose behavioral constraints. A visual context is taken to be the visually perceived environment. Because the system is to be distance independent, image pyramids are used. The system is to be applicable to a wide range of possible visual contexts. Therefore, no specific information about the scene or task is used. Instead, autocorrelation functions of the various scales of the pyramid are calculated and added. The resulting feature vectors are classified using a linear Baysian classifier.
The system can function supervised, with a set of pretrained classes, as well as unsupervised, making its own classes at runtime. It is integrated with the behavioral architecture of an autonomous robot. The use of the system and the architecture is discussed. Also, the system is tested on a set of different tasks, both with pretaken images and in real time tasks. The advantages and disadvantages of the system are discussed.

Contents

1 Using Visual Context in an Autonomous Robot 2

2 Background Theory 5
- 2.1 Behavior Control through the use of Dynamical Systems 5
  - 2.1.1 How to select appropriate Actions ? 5
  - 2.1.2 Dynamical Systems. 6
  - 2.1.3 The Complete Architecture 8
- 2.2 Image Recognition based on Higher Order Autocorrelation Features 10
  - 2.2.1 Scale Invariancy through the use of Image Pyramids. . . . 11
  - 2.2.2 Higher Order Autocorrelation Features 12
- 2.3 Feature Classification using a Linear Bayesian Classifier 13
  - 2.3.1 Bayes Decision Rule 13
  - 2.3.2 Dimension Reduction 15
  - 2.3.3 The Final Algorithm 15

3 Arnold 17
- 3.1 Hardware 17
- 3.2 Planet 18

4 Classifying Scenes on an Autonomous Robot. 19
- 4.1 Image Content 19
- 4.2 Using Stereo Images 20

5 Using the Classifier Stand-alone. 22
- 5.1 Separability of Classes 22
- 5.2 Recognizing Walls 25
- 5.3 Cutting out Parts of Images. 26
- 5.4 Recognizing Rooms 28

6 Dynamic Behavioral Classifier Module 32
- 6.1 Scenario 32
- 6.2 Using the Classifier in the Dynamical Systems Control Architecture 32
- 6.3 Stabilizing the Recognition with a Low Pass Filter 33
- 6.4 Results 34

7 Autonomous Classifying 38
- 7.1 Unsupervised Learning 38
  - 7.1.1 Maximum Likelihood Methods 38
  - 7.1.2 Unsupervised Bayesian Learning 38

7.1.3 Basic Isodata Clustering 39
7.2 On Line Learning 39
7.2.1 Within Class Borders 40
7.2.2 Between Class Borders 40
7.3 Real Life Experiment 41

8 Summary and Concluding Remarks **44**

9 Documentation **46**
9.1 QNX programs 46
9.1.1 recog_module 46
9.1.2 pic_stereo 47
9.1.3 train 47
9.1.4 clust 47
9.1.5 setcl 47
9.1.6 recog_dyn 47
9.2 Solaris programs 47
9.2.1 cluster_test 47
9.2.2 Utilities 47
9.3 Configuration file 49

List of Figures

2.1 Overview of the complete dynamic behavior architecture. 9
2.2 Overview of the recognition system 10
2.3 Scale Invariancy in a band-pass pyramid 11

3.1 Arnold, an anthropomorphic autonomous robot for human environments. 18
3.2 The Stereo Camera Head 18

5.1 Plot of feature vectors of three classes after all images have been trained.The plot was generated using a small set of mono images with 3x3 autocovariance features, an image pyramid with 6 scales and half octave scale distance and a Laplace filter. 23
5.2 Plot of feature vectors of the room set (figures ?? and ??). Per class, four images have been trained, and two are solely used to calculate feature vectors from. It can be clearly seen that the separability is not so good. 24
5.3 Room set; four images per class. 24
5.4 Room set; two untrained images per class. 25
5.5 Plot of feature vectors of all images, after 6 images had been trained. 25
5.6 Plot of feature vectors of all images, after 8 images had been trained. 26
5.7 Plot of feature vectors of the room set (figures ?? and ??) after all images have been trained, using *autocorrelation* features . . . 27
5.8 The Foyerset; 33 sets belonging to 5 classes. Only one image per head angle per set is shown. 30

6.1 Arnold at various stages in the scenario. 34
6.2 Distances to the four trained classes of the images that were taken at the moments the door extractor found doors. Classes 1 and 3 are the doors. Class 2 is not a door, but a spot at which the door extractor always found doors. Class 4 is a waal that has nothing to do with doors. 35
6.3 Plot of the activity and context values for the room scenario. Only the most relevant behavioral variables are plotted to keep the plot from becoming too crowded. The activation of the *Recognition* and *Tracking* behaviors after the *Visual Search* has found a door can clearly be seen. 37

List of Tables

4.1 Recognition rates using various configurations. Three images each of four classes were trained, three other images per class were classified. 1 scale, 3 scale and 6 scale pyramids were used, the higher scale pyramids with half and whole octave distance between the scales. Autocorrelation (ACF) and autocovariance (ACV) features were used, both with a 3x3 and 5x5 displacement grid. 19

5.1 Results from the first Foyer test with 6 scale pyramid. Autocorrelation and autocovariance functions were both used, with 3x3 and 5x5 displacement grids, and octave as well as half octave distances between scales. Furthermore, the test was done with the feature vectors of mono images, concatenated feature vectors of stereo images, and added feature vectors of stereo images. . . . 27

5.2 Results from the second Foyer test with 6 scale pyramid, 10 images trained. The same classifier configurations as in ?? were used. 28

5.3 Recognition rates using cut out images vs. whole images. 10 images of each class of the foyer set were trained, using autocorrelation and autocovariance functions with a 3x3 displacement grid. 1 scale and 3 scale pyramids were used. For comparison, also complete images were classified using the same configurations. 28

5.4 Results from the first roomtest with 3x3 Mask 31

7.1 First unsupervised clustering test in the vision lab, using between class borders. 5 passes were made. With each pass, pictures were taken at pan angles 45 degrees apart. These angles were decreased by 5 degrees every pass. 41

7.2 First unsupervised clustering test in the vision lab, using whithin class borders. Again, 5 passes were made, and with each pass, pictures were taken at pan angles 45 degrees apart; these angles were again decreased by 5 degrees every pass. 42

7.3 Second unsupervised clustering test in the vision lab, using between class borders. The procedure followed was similar to the previous two tests, except more pan-angles were used, and four classes were trained beforehand (marked with a *) 43

Chapter 1

Using Visual Context in an Autonomous Robot

The goal of the system is recognizing *visual contexts*. A visual context is the visually perceived environment of a certain object or agent. To make this somewhat more clear, one could think of *scenes*. In the real world, almost everything perceived will be a part of a certain scene (for example, a car will mostly be seen in a kind of road scene). A visual context can be seen as that part of a scene around an object or agent that can contain relevant information. For instance, the visual context of a door will be a part of the wall around the door that can be helpful in identifying it. If a robot looks at a wall, then the wall will be the visual context of the robot. On the wall can be various objects, such as posters, doors, switches, etc., that all share the same visual context as the robot. These examples indicate that what constitutes a visual contexts is dependent on the task it is to be used for. The same picture of a wall could function as a visual context for the recognition of a poster, but also for the navigation of a robot.
Recognition of visual contexts can be helpful, because it makes the perceiving of the environment of the robot a lot easier. If it it is known that a system is in a certain context, it is known that it can only be in certain states, or only has to exhibit certain behaviors. It is not necessary to have a complete model or complex detection algorithm for specific states of the environments. The task of perceiving the environments is split; first, determine the context, then determine what is in that context. The latter task is made a lot simpler because the search space is a lot smaller. In areas as speech recognition or natural language processing, context models are widely used to improve systems, because of the ambiguity of language. If there is uncertainty about which of two words is heard, the context of the word can give clues.
In autonomous robotics, context information is specifically desirable. Fast methods are needed for perceiving the environment, because the robot will have to operate in real time. If the task of perceiving the environment can be made more efficient, this will certainly help. Furthermore, visual context can be used by various behaviors. If the perceived context is a certain room, for example, a variety of behaviors could use that information: navigation behaviors, that can determine whether to go ahead to the next room or not, search behaviors, which can have knowledge where to search in certain rooms, recognition behaviors,

which can have knowledge that certain structures, for instance edges, are doors in one room and posters in others, or even obstacle avoidance behaviors, which can have knowledge about what obstacles to expect or how destructive collisions can be. Roughly said, using visual context information in an autonomous robot can be helpful in three ways:

- **It can help the navigation of the robot by providing a sense of location.**
  Contexts could be bound to specific locations. A certain wall of a room might be such a context, or the view the robot gets when he is in the dooropening of a room. A context recognizer can give the robot a rough sense of location. This is not a precise localization, of course. However, this is not needed for some tasks. If the robot knows it is in the room where it can get coffee, for instance, a new behavior can be started to look for the coffee machine, which doesn't need exact location information. Furthermore, it can be used to recalibrate navigation systems based on other information, such as dead-reckoning or distance-from-walls information.

- **It can help the recognition of objects.**
  Object recognition can be a complicated issue. If one wants to recognize specific objects, mostly complicated computational procedures are required, and much information is needed beforehand. If one can use contextual information, however, one can make the recognition simpler. For example, it is fairly easy to recognize a round shape. This round shape can be various things, mugs, thrashbins etc. However, a thrashbin will usually be encountered in a specific context (the floor or a corner of a room), and a mug will usually be in different contexts (for instance a table or a kitchen sink). Split in the two steps of roughly recognizing shape and coupling this with the context can make object recognition much easier.

- **It can impose behavioural constraints on the robot.**
  Some actions should better not be performed in certain contexts. If a robot sees the context of a door, it is allright to move to it. However, if the perceived context is that of a staircase going down, it might be a good idea to go some other way.

Visual context information must thus be used in a supporting role; it always functions as an enhancement of other information. Furthermore, the contexts used or perceived are dependent on the task that needs to be fullfilled. The system therefore has to be integrated in the robot control architecture, so it can be used by other systems.

To be able to recognize scenes or visual contexts, one must use information about the whole image that is to be processed. It is not known beforehand which parts of the image are fairly specific for the context, and which parts are not. Determining that would be a recognition problem itself. Also, the system was meant to be applicable to a range of tasks without needing much information about the task or task environment beforehand.
Because the system is to function in the real world and be applicable to a range of tasks, a wide range of different contexts have to be handled. Furthermore,

this means the used method has to be relatively invariant with respect to translations of the robot. These two requirements imply one cannot use recognition scheme's that rely on very image specific features (like shape). Rather, one would use a fairly general method based on statistical properties of the images. Furthermore, one would like these properties to be relatively independent of the distance and position relative to the robot. These considerations lead us to the use of autocorrelation features calculated from image pyramids.
The output of the system is to be used to augment other information. Specific information about the scene is not required, just an identifier of the visual context. Therefore, the autocorrelation features are classified. Each resulting class represents a visual context in the specific task.
The various contexts can be trained beforehand, if the task to be performed is well known in advance. Although this is very usefull for some tasks, other tasks require that the contexts can be autonomously learned. When exploring an environment, it is of course not known which contexts will be encountered. Therefore, the system also has to be able to learn unsupervised.

Chapter 2

Background Theory

2.1 Behavior Control through the use of Dynamical Systems

2.1.1 How to select appropriate Actions ?

To control an autonomous agent, one needs an architecture which selects the appropriate action for each situation. There are basically three approaches to realize such control.
The *Control Theoretic* approach is based on making explicit links between sensory input and actuator output. There is no need for extracting the relevant information from the sensors to represent the environment; the raw sensor data is used for control. For example, a robot can be constructed to navigate to a certain point of light. This is realised by placing two light sensitive sensors on the left and right side of the front of the robot. The output from these sensors is used as inhibition for the motors on the same side of the robot (thus, if the light is left before the robot, the left motor will be inhibited, making the robot turn to the left). Although fast and reliable, this approach has its problems. It can not switch between different actions if the environment changes, nor choose between several potential targets. If there are more points of light, or if there are obstacles on the path to the light, this architecture will fail.
The *Artificial Intelligence* approach is based on internal, symbolic representations of the environment, which are invariant with respect to transformations and actions. On the basis of these representations, decision algorithms can be used to select the appropriate action. For example, to reach a point of light, such a robot will have a map of the room with the light, and its own position. A path planning algorithm can then be used to determine a path to the point of light. The robot will then follow this path. However, this approach has its problems too. The representations of the environment must both be adequate and constantly updated. If, for example, the point of light is moving, or there are moving obstacles in the room, this architecture will have serious difficulties reaching its target. Also, decision algorithms have to be given for each possible situation. These can be time-consuming both to design and to execute.
The *Behavior Based* approach tries to control the agent by specifying elementary *behaviors*. These low level behaviors can be viewed as control systems, and

require no extraction of relevant information and representations. However, these behaviors can interact with each other. The overall behavior is generated by the interaction between the individual, elementary behaviors. For example, our robot could have a control system like behavior to reach the point of light, similar to the first example. Also, it would have an obstacle avoidance behavior (for example based on close range sensors which inhibit the opposite motor). If an obstacle is near, the obstacle avoidance behavior will become active, inhibiting the light searching behavior. If a choice is to be made between different points of light, another behavior can be specified which looks for the right point of light and inhibits the light searching behavior while active. In its most strict formulation [5, 6], the interaction between behaviors consists solely of on/off signals; however, other versions include various forms of interaction. Indeed, the big problem with this approach is the nature of the interactions between elementary behaviours. It is still difficult to genererate complex task solving behavior. If a sequence of behaviors is to be carried out, only sensory information is not enough. If, for example, our robot first has to go to a red light, and then to a green light, it would need a kind of memory, and behaviors to look for both lights.

2.1.2 Dynamical Systems.

A solution to the problem of action selection in behavior organization is to use *dynamical systems* to control this interaction, as described in [20, 22, 23]. Dynamical systems in this context are nonlinear differential equations. These systems exhibit certain types of solutions, *fixed points*, at which the rate of change of the system variable is zero; these points are constant solutions of the system. A fixed point can be *asymptotically stable*, which means the system converges in time to the fixed point from points nearby. Asymptotically stable fixed points are called *attractors*, and are precisely the kind of solutions usefull for behavioral control.

Behaviors are activated by evolving a dynamical system in time. In [23], each of the i behaviors is characterized by a continuous *behavioral variable* n_i. This variable represents the current activity of the behavior; if $n_i = 0$, the behavior is not active, if $n_i = 1$ the behavior is active. The vector of behavioral variables is the behavioral state of the system, and can be seen as a point in *behavioral space*. This vector is updated by the dynamical system. The task to be fullfilled must be expressible as points or sets of points in the behavioral space, independent of the current behavioral state of the dynamical system. The dynamical system has to be chosen so, that it has two attractors; one for activation and one for deactivation of the behavior.

The use of dynamical systems for behavior control has many advantages. Stability of the system can be analyzed algebraically, and is guaranteed by the use of attractors. Also, sensor outputs can be explicitly included in the system; because the system is defined mathematically, sensor values can be directly incorporated in it. Furthermore, if correctly chosen, the behavioral variables represent a close link between observed behavior and behavioral state of the system.

Competitive Dynamics

The dynamical system that governs the behavioral variables has two attractors. To which of these attractors the behavioral variable converges is controlled by the *competitive advantage* α_i. This competitive advantage in turn is dependent on sensor input and logical context, as will be explained later. Also, the convergence is dependent on competition with other behavioral variables. This is realised through a *competition matrix* $\gamma_{i,j}$, with elements valued 0 or 1. This competitive term ensures, that behaviors which cannot be active at the same time inhibit each other. For example, one can have a target tracking behavior i; this behavior has to be (temporarily) switched off if an obstacle is detected, and the obstacle avoidance behavior j has to become active. In this case, one sets $\gamma_{i,j} = 1$. If behaviors i and j can be active independently, one sets $\gamma_{i,j} = 0$. Whith these terms, we are led to the following differential equation:

$$\tau \dot{n}_i = \alpha_i n_i - |\alpha_i| n_i^3 - \sum_{j \neq i} \gamma_{i,j} n_j^2 n_i + \zeta_t \tag{2.1}$$

τ influences the timescale on which the system relaxes to the attractors. ζ_t is a gaussian white noise term, which is included to prevent the system from remaining in unstable states.

Competitive Advantage and Refractory Dynamics

The competitive advantage a_i is dependent on sensor input and logical context. However, for a stable behavior organization, oscillations have to be avoided. These are likely to happen because of noisy sensor input. For example, if an obstacle is relatively near but not near enough for a collision, the obstacle detector might switch swiftly between detecting and not detecting the obstacle, resulting in the continuously switching on and off of the obstacle avoidance behavior. Therefore, a mechanism is needed which can keep a behavior inactive for a while after it has been inactivated. To this end, a refractory term r_i is used. Sensor input and logical context are combined in an input term I_i; further on we will explain how this is done. This I_i is used, together with r_i, to determine a_i. This can be done with the following equation:

$$a_i = 2 r_i I_i - 1 \tag{2.2}$$

The refractory term r_i can then be determined by the input term I_i:

$$\dot{r}_i = \frac{(1 - r_i) I_i}{\tau_{r,2}} + \frac{r_i (I_i - 1)}{\tau_{r,1}}, \text{with } \tau_{r,1} \ll \tau r, 2 \tag{2.3}$$

Normally, 2.3 will be in the stable fixed point $r_i = 1$. An input $I_i \simeq 1$ will lead to a positive competitive advantage $a_i \simeq 1$, which enables n_i to converge to 1 if there is no competion with other behaviors. If the input is switched off now ($I_i \simeq 0$), 2.3 will converge to the fixed point $r_i = 0$ on the fast timescale $\tau_{r,1}$. a_i will be -1 now, so the behavior can not be activated. Eventually, the refractory term will again converge to $r_i = 1$ on the much slower timescale $\tau_{r,2}$. This enables the behavior to be switched on again. The duration of this period, determined by the time constant, can be made to reflect the demands of the behavior.

Sensor Input and Logical Context

The input term I_i is a function of sensor input and logical context. Sensor input is incorporated through a *sensor context* term C_i. This term defines the neccessary sensor condition for activation. The sensor doesn't have to be a physical sensor. For example, the output of an obstacle recognition algorithm can function as a sensor, with as output the confidence with which an obstacle is recognized. The sensor context enables the coupling of activation of behaviors to sensor information. However, not only sensor context has to be included. If the overall behavior consists of a sequence of events, it is also important to incorporate information about the logical structure of the sequence. Therefore, a *logical context* term is included. The logical dependencies are specified in the *activation matrix* $A_{i,j}$. An element $A_{i,j} = 1$ means behavior j must be active before behavior i. Furthermore, a *short term memory* is included. This ensures that the switch between two behaviors is more stable. For example, if behavior i must follow on behavior j but cannot be active simultaneously, it is possible behavior i is never switched on because behviour j becomes inactive too fast to have an influence on the logical context. With the short term memory, although behaviour j is already switched off, its memory is still high, enabling behavior i to switch on. This leads to the following equations:

$$I_i = C_i \prod_j (1 + A_{i,j} \frac{\tanh(cm_j) - 1}{2}), \text{ with } c \gg 1 \tag{2.4}$$

$$\dot{m}_i = \frac{(1 - m_i)n_i^2}{\tau_{m,1}} + \frac{(1 + m_i)(n_i^2 - 1)}{\tau_{m,2}}, \text{ with } \tau_{m,1} \ll \tau_{m,2} \tag{2.5}$$

If $|n_i| = 1$, the memory state relaxes to the attractor $m_i = 1$ on the fast timescale $\tau_{m,1}$ (following the activation of the behavior immediately). If the behavior is subsequently switched off again ($n_i = 0$), the memory stays in the state $m_i = 1$ for a time $\tau_{m,2}$ before it relaxes to the attractor $m_i = -1$.

2.1.3 The Complete Architecture

The complete dynamic behavior architecture is shown in figure 2.1. The time constants used can be specifically tailored to the behavior it belongs to, but mostly one can use the same numbers for every behavior. As mentioned, it has several advantages. It is stable, and this stability can be algebraically analyzed. It can incorporate sensor data and logical dependencies. It allows for multiple behaviors to be integrated in the same system and can handle sequences of behaviors. It exhibits a close link between observed behavior and internal behavioral variables, which makes it relatively easy to design and debug. Also, the system is *reactive*. If a sequence of behaviors is disturbed by an unexpected event, the architecture can respond swiftly by switching to an appropriate behavior because of the direct sensor coupling. After this, the sequence can be picked up again, because of the memory dynamics, or if the situation has changed too much, a new behavior can be activated or the sequence can be started anew. Furthermore, because of the strictly local logical dependencies in the competition and activation matrices, it is possible to *integrate new behaviors* without reconfiguring the whole system. All that is needed are an additional row and column in the two matrices γ and A, and perhaps new time constants

for the refractory- and memoryterms. These features make the architecture very flexible.

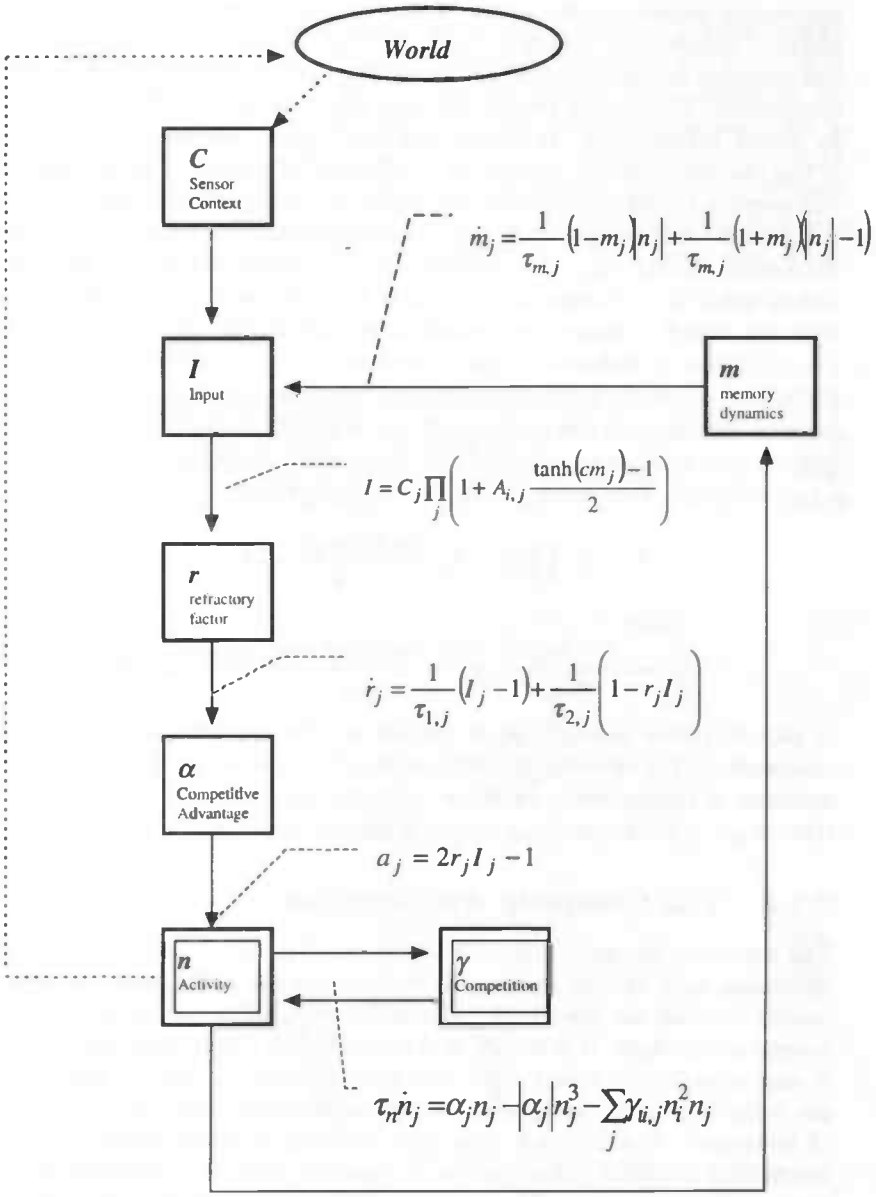


Figure 2.1: Overview of the complete dynamic behavior architecture.

2.2 Image Recognition based on Higher Order Autocorrelation Features

To be able to make use of contextual information one must be able to recognize or classify images. Since the system is to be used on an mobile agent in the real world, it has to have certain characteristics:

- Translation invariancy, both in distance and in place.
- No a-priori knowledge or models needed.

The used system is based on classification of higher order autocorrelation features [14, 15, 16, 17]. In figure 2.2 a schematic overview of the system is given.

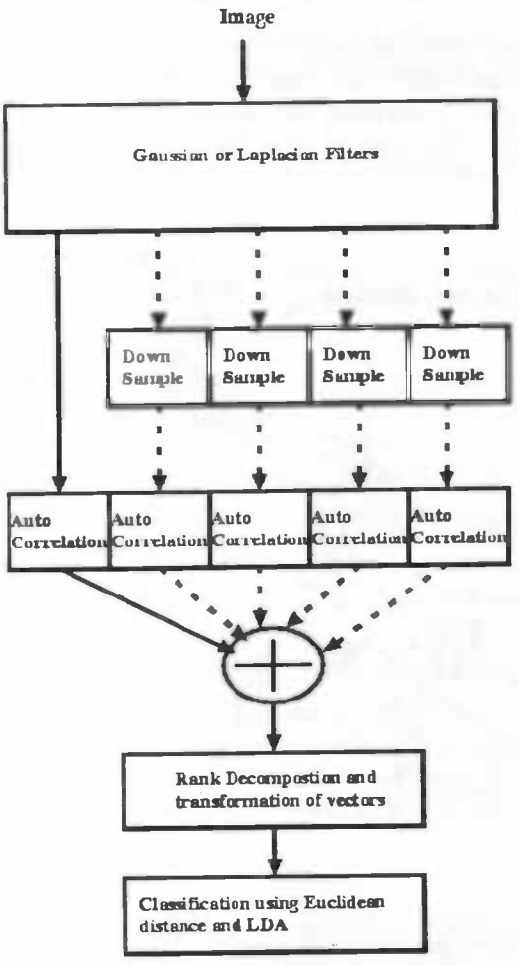


Figure 2.2: Overview of the recognition system

2.2.1 Scale Invariancy through the use of Image Pyramids.

Low Pass Pyramids

Image Pyramids are a wide spread tool to represent images on different scales. The image is downsampled with factors corrresponding to the scales to be used. For instance, say an image has a size of 256x256 pixels. This would be the first scale in the pyramid. For the next scale, the image is downsampled to 128x128 pixels. Each downsampled image thus forms one scale in the pyramid. Of course, the image first has to be low-pass filtered to ensure the Nyquist theorem is obeyed, so no aliasing effects occur. An image pyramid is an image representation consisting of various scales containing a lower resolution image of the original.

Band Pass Pyramids

One can also use band pass filters to construct an image pyramid. Instead of low pass filtering the image before each resampling for a scale, the image is band pass filtered. This gives a pyramid with scales that represent the image on various frequency bands. The sum of these bands gives the original image back.

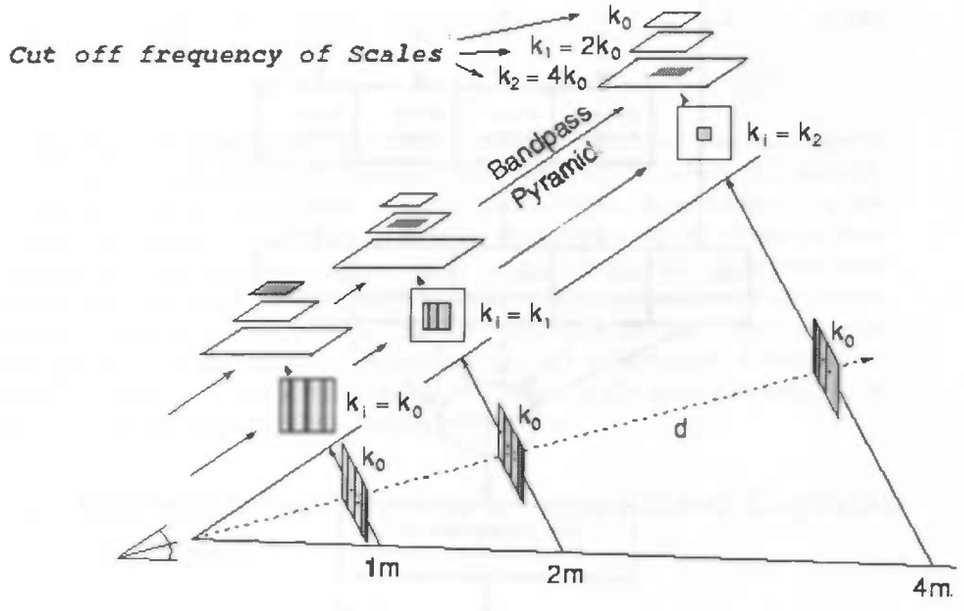


Figure 2.3: Scale Invariancy in a band-pass pyramid

Scale Invariancy

How do image pyramids ensure scale invariancy ? To show this we need the concept of *Scale translation*. Figure 2.3 shows the representation of an object in a band pass pyramid with different distances between camera and object. The object in this case is a plane with a texture consisiting of monofrequent bands. If the distance increases, the frequency of the texture in the recorded image also increases. The distances of the object and the scales are chosen to ensure this frequency is identical to the cut-off frequency of the filter at every scale. Doubling the distance will give the filter with double cut-off frequency maximal output, and correspondingly the representation of the image will shift downwards in the scales of the pyramid. For a double distance, the frequency will also have doubled, wich means the previous scale will get maximal filter input and represent the object. The size of the object will decrease with increasing distance. But because the sampling rate increases with increasing frequency (thus with increasing distance) the size of the object in the scale will in theory stay the same. A distance translation (confusingly also called scaling) of the object will result in a shift of its representation in the image pyramid to another scale while keeping its size.

2.2.2 Higher Order Autocorrelation Features

Definition

To extract features from the resulting image pyramid higher order autocorrelation functions are used. Higher order autocorrelation functions of a function f with domain $D(f)$ are defined by [18]:

$$R_f^k(a_1, \ldots, a_k) = \int_{D(f)} f(t)f(t+a_1)\ldots f(t+a_k)dt \tag{2.6}$$

where k is the order of the function, and the a_i the translation vectors. $t + a_i$ must be within the domain $D(f)$. These functions are translation invariant and additive, which means:

$$R_g^k = R_f^k \quad \text{for translation } g(t) = f(t+a) \tag{2.7}$$

$$R_{f+h}^k = R_f^k + R_h^k \quad \text{for function } h(t) \tag{2.8}$$

If white noise is added to f, it has no influence on the autocorrelation except at $(0, \ldots, 0)$. Is is a usefull technique for noise suppression. A global intensity change of the values of f, for example a change in lighting conditions, can be described as a multiplication of f with a scalar value.

Features

Because of their translation invariancy the values from autocorrelation functions seem to be good candidates for features to be used by a classifier. The first use of autocorrelation features was made by Otsu et. al. [16, 17]. They concatenated the feature vectors from various scales of an image pyramid. Kreutz extended [14, 15] this work of Otsu by using the additiveness property and adding the autocorrelation functions of different scales of an image pyramid to one feature

vector. Because an object seen from different distance in a bandpass pyramid is represented as having the same size but being shifted in scale (see 2.2.1), this means the features will be distance invariant as well as translation invariant. The computation of autocorrelation functions from an image must of course be done discretely, using:

$$R_f^k(a_1, \ldots, a_k) = \sum_{t \epsilon D(f)} f(t)f(t + a1) \ldots f(t + ak) \tag{2.9}$$

However, because of computation time, the number of evaluated displacements k must be sufficiently small. Therefore, only second order ($k = 2$) autocorrelation functions are used and the displacement region is not made bigger than a 5 x 5 grid.

However, this is not so bad as it seems. We calculate the autocorrelation functions from the various scales of an image pyramid. Therefore, relatively large structures will be represented in these functions calculated from the higher scales, while relatively small structures will be represented in the functions calculated from the lower scales.

Higher Order Autocovariance functions

Analogous to autocorrelation functions autocovariance functions are defined by:

$$C_f^k(a_1, \ldots, a_k) = \int_{D(f)} (f(t) - E(f(t)))\,(f(t + a_1) - E(f(t + a_1))) \ldots (f(t + a_k) - E(f(t + a_k)))dt \tag{2.10}$$

The difference with autocorrelation functions is that here the mean value (average grey value of an image) is subtracted from the function values. Autocovariance funcions too are translation invariant and additive. In addition, they are invariant with respect to offset translations. Features based on autocovariance functions have the same properties. Both autocorrelation and autocovariance features can have slight advantages, depending on the application. If the image contrast is poor, it is better to use the autocovariance features. However, the mean grey level in an image can certainly hold some information. Compare for example a black door and a white wall; both will have the same autocovariance features, but very different autocorrelation features

2.3 Feature Classification using a Linear Bayesian Classifier

2.3.1 Bayes Decision Rule

Bayesian statistics is a foundation of object classification [9, 8]. According to Bayesian statistics, the a posterori (afterwards) probability of environment state ω_k given observation x can be calculated from the a priori (beforehand) probabilities of occurence of ω_k, x and x given ω_k:

$$P(\omega_k \mid x) = \frac{P(x \mid \omega_k)P(\omega_k)}{P(x)} \tag{2.11}$$

with

$$P(x) = \sum_{i=1}^{M} P(x \mid \omega_i) P(\omega_i). \tag{2.12}$$

We can construct a decision rule using 2.11 which assigns a class k to a vector x, by choosing the class which has the highest probability of producing the observed vector. This is known to be optimal in terms of minimizing the risk of misclassification. When making a linear classificator, this decision rule can be stated in terms of the Mahanalobis distance [19, 9]:

$$\text{Assign } x \text{ to class } k, \text{ if}$$
$$(x - \mu_k)\sigma_{w,k}^{-1}(x - \mu_k)^\top = \min_{j=1,\ldots,M} (x - \mu_j)\sigma_{w,k}^{-1}(x - \mu_j)^\top \tag{2.13}$$

with:
x = the vector to be classified
μ_k = mean vector of class k
$\sigma_{w,k}$ = *within*-covariance matrix of class k

Some assumptions have to be made for this:

- It is assumed the feature vectors of each class obey a normal distribution.
- We have to know the matrices σ_w and the vectors μ_k's.
- Since we assume linear separability of classes, we assume all classes have the same covariancematrix. Hence, the individual *within*-covariance matrices are replaced by the *overall within*-covariance matrix σ_w. which is the covariance matrix of all (trained) vectors and their classes.

Because we do not have full knowledge of the underlying probability distributions, we use estimates for σ_w and the μ_k's, based on training data. This results in:

$$\text{Assign } \mathbf{x} \text{ to class } k \text{ ,if}$$
$$(\mathbf{x} - \bar{\mathbf{x}}^k)\mathbf{W}^{-1}(\mathbf{x} - \bar{\mathbf{x}}^k)^\top = \min_{i=1,\ldots,M} (\mathbf{x} - \bar{\mathbf{x}}^i)\mathbf{W}^{-1}(\mathbf{x} - \bar{\mathbf{x}}^i)^\top \tag{2.14}$$

$$\mathbf{W} = \sum_{k=1}^{M} \frac{N_k}{N \sum_j (\mathbf{x}_j^k - \bar{\mathbf{x}}^k)(\mathbf{x}_j^k \bar{\mathbf{x}}^k)^\top} \tag{2.15}$$

with:
$\mathbf{x}$ = the vector to be classified
$\bar{\mathbf{x}}^{\mathbf{k}}$ = the mean vector of class k
$\mathbf{W}$ = the total within-class covariance matrix
M = number of classes
N = number of samples
N_i = number of samples per class i

2.3.2 Dimension Reduction

In order to be able to perform the matrix inversions which are necessary for the above mentioned calculations, the matrix has to be singular. However, this means that the number of linearly independent samples has to be much greater then the dimension of the featurevector. In order to circumvent this problem, rank decomposition is used.

Because we assume normal distributions, we can use Linear Discriminant Analysis to create a subspace of the featurespace containing the centroids of the M classes. We can thereby reduce the dimensionality of our classification space to M - 1, reducing the computational requirements. Linear Disrcriminant Analysis was proposed by Fisher [10]. It is very similar to the variance-maximizing rotation of Principal Component Analysis. It is based on a discriminant criterion, the well known Fischer ratio.

2.3.3 The Final Algorithm

The system has to be able to train new classes and classify vectors interchangeably. Because we do not want to calculate the within covariance matrix from the entire training data set again every time we want to classify something, we accumulate the trained vectors in an autocorrelation matrix R_{xx}. Furthermore, we accumulate the total sum and the per class sum of the vectors. When a classification is needed, the new covariance matrix S_w can be recomputed by computing the total covariance matrix and the between-class covariance matrix; the within covariance matrix is simply the difference between these two. This gives the following algorithm:

IF Training:
Get new vector x of class k
$R_{xx} = R_{xx} + xx^\top$
IF k is new:
make new classvector $\hat{x}^k = 0$ and new class k
$\hat{x} = \hat{x} + x$
$\hat{x}_k = \hat{x}_k + x$
$N = N + 1$
$N_k = N_k + 1$
MODIFIED = true
IF Classifying:
Get vector x
IF MODIFIED
$S_w = R_{xx} - \sum_k (\hat{x}_k \hat{x}_k^\top)/N_k)$
$S_b = \sum_k (\hat{x}_k \hat{x}_k^\top)/N_k) - \hat{x}\hat{x}^\top/N$
Λ_w = diagonal matrix of Eigen values of S_w
Γ_w = matrix with corresponding Eigen vectors
$T = \Gamma_w \Lambda_w^{1/2}$
Λ_b = diagonal matrix of Eigen values of S_b

Γ_b = matrix with corresponding Eigen vectors
$T = T\Gamma_b$
MODIFIED = false
KLASS = k **when**
$| T^\top x - T^\top \hat{x}_k/N_k | = \min_i | T^\top x - T^\top \hat{x}_i/N_i |$

Chapter 3

Arnold

3.1 Hardware

Arnold (see figure 3.1) is the robot used for the experiments. He consists of the following components:

- A TRC "Labmate" robot platform (modified).
- A stereo camera system, based on the system "Zebra", also manufactured by TRC. It includes two AMTEC MORSE wrist modules for pan (rotation in the horizontal plane) and tilt (rotation in the vertical plane), controllable through software. Most of the other degrees of freedom can be manually adjusted. The system is equipped with two Sony XC-999P high resolution color CCD-cameras (PAL) with 6mm lens (Fovea camera's) and two Sanyo VCK-465 monochrome cameras with an opening angle of nearly 90 degrees, and a 3.6mm lens (Periphery cameras) . For a picture of the complete camera system, see figure 3.2
- An arm consisting of AMTEC MORSE modules. This arm has 7 degrees of freedom and is thus able to do obstacle avoidance during grasping operations. The special layout of this arm has been developed at the Institut für Neuroinformatik.
- Two Pentium P166 PC's. One of them is equipped with a harddisk on its board and is the master of the whole platform. Using an I/O-card it has complete control over the power supply of all the other hardware. This PC also contains the PCI-bus-framegrabbers that are used to acquire images. The second PC essentially contains the hardware to control the robot-arm and a soundcard Soundblaster 16. The network connection between the PC's is realized using 100 Mbit Fast Ethernet cards CT-FE120 by Corman Technologies. The adapter on the second board is equipped with the boot-ROM CT-FE002, also manufactured by Corman Technologies. The interface to "outside" is realized as a 10 MBit Ethernet which is connected to an inhouse-network.The PC's are running under the realtime multiuser multitasking operating system QNX.

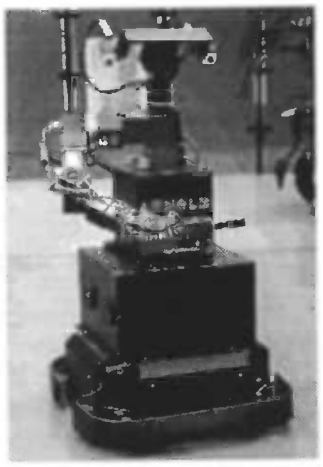


Figure 3.1: Arnold, an anthropomorphic autonomous robot for human environments.

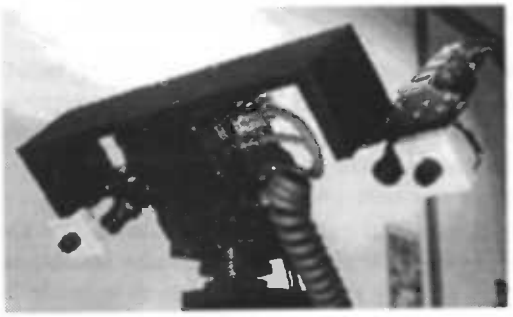


Figure 3.2: The Stereo Camera Head

3.2 Planet

To control the various hardware modules and enable different behavioural modules to run at the same time PLANET is used. PLANET is a platform for process control and -communication based on the QNX operating system. It was developed specifically for this purpose by a cooperation between the Universität Dortmund, fachbereich Informatik and the Institüt für Neuroinformatik, Bochum.

Chapter 4

Classifying Scenes on an Autonomous Robot.

4.1 Image Content

The system performs very good when it is used on images containing mostly the object to be recognized. Kreutz [15] was able to get very high recognition rates, as was I on a small set of images taken using Arnolds camera. (see table 4.1)

However, when using the system 'in real life' care has to be taken. The task is somewhat different in this case, because we want to recognize scenes, not objects. These kind of problems are already somewhat apparent in table ??; sometimes surprisingly low recognition rates appear. In this simple test, the distances at which the images were taken were fairly similar, which explains why the 1 scale pyramid performs surprisingly good. Including other scales here does not add any information, and can only distort information present in the original image. This does not explain all of the strange results though.
In essence, we are classifying using the statistics of the whole image. In Kreutz' applications of the system, mostly the object to be recognized composed a significant part of the image, or had no background. The only real variation in the

	1 Sc.	3 Sc., oct.	3 Sc., 1/2 oct	6 Sc., oct.	6 Sc., 1/2 oct.
ACF, 3x3	92	66	75	92	92
ACV, 3x3	75	100	16	100	33
ACF, 5x5	92	83	75	75	45
ACV, 5x5	100	100	92	100	92

Table 4.1: Recognition rates using various configurations. Three images each of four classes were trained, three other images per class were classified. 1 scale, 3 scale and 6 scale pyramids were used, the higher scale pyramids with half and whole octave distance between the scales. Autocorrelation (ACF) and autocovariance (ACV) features were used, both with a 3x3 and 5x5 displacement grid.

images therefore was caused by the object in it. However, in this application, the variation in the images is caused by everything in the image. This makes it a lot harder for the classifier to discriminate between images. Autocorrelation features are just statistical features, and can not discriminate between variation that is not so important for the classification and information that is. A solution to this is to train many images per class.
Furthermore, because the position of the robot will change during various classifications, the image is very likely to contain significant parts that were not in the trained images. This can severely disturb the classification. This is an especially hard problem, because one cannot know a priori which part of the context will be on an image. Even when the center of two images contain essentially the same, the surrounding can influence the classification heavily. This is not a question of translation- or scale invariancy, but simply has to do with the content of an image.
There are three possible solutions of this problem:

- Tasks can be severely structured, so it is fairly certain in advance that the images the robot has to classify have approxamitely the same content as the trained images. This is the approach followed in 5.4 and 6.

- One can use depth information to make sure the images to be classified represent areas of approximately the same size in the real world. However, this may still not work very good, because the content of the images can still be very different. For instance, if one classifies images of two meters of a wall, it can still be that the images are from fairly different parts of the wall (one with a poster in it, one with a door, etc.) However, this idea was roughly tested (see 5.3), and didn't seem to improve the classifying significantly.

- One can use a method to determine which part of the image contains the relevant information to be used for the classification. This is a -difficult enterprise. A method has to be devised that can filter out irrelevant parts of an image, while there is very little information present about what these irrelevant parts are. Furthermore, the relevant parts of the image will differ per class, so for all classes trained, the image has to be processed differently. This means two additional steps in the classifying algorithm, that would increase the complexity and computational requirements of classification considerably.

4.2 Using Stereo Images

Arnold's vision system consists of two stereo camera systems. Instead of one, that means we can work with two images at a time. This can give a more reliable picture of a specific scene. How can we use two images with the classifier ?
We could of course devise a way to combine the results of classifying the two images. However, this is no direct sound solution. We then have to classifiy a two dimensional vector with nominal values. If one of the images is wrongly classified, then the whole classification will suffer. Adding or correlating the images are computationally expensive operations, which run the risk of blurring information that was present in the images. Also, the result would be very

dependent on the precise alignment of the two images.
It would be better to combine the resulting featurevectors from the two images. As remarked in 2.8, autocorrelation functions are additive. Clearly, what we want to classify is the whole of the two images; therefore, taking the sum of the two featurevectors and feeding this to the classifier seems a good way to handle stereo images. Alternatively, the two feature vectors could be concatenated. Both approaches have been pursued (section 5.4). Adding the vectors gave better results.

Chapter 5

Using the Classifier Stand-alone.

5.1 Separability of Classes

In order to test whether the images to be used were separable in principle, some classifiers were trained with only three classes. Due to the dimensional reduction performed to make the classification easier (see 2.3.2), the distance calculation will be done with twodimensional featurevectors. These vectors can be conveniently plotted as points in a two dimensional space. If the linear separability holds, then one must be able to distinguish clearly separated groups. In figure 5.1 one can see that this clearly is the case.

However, these feature vectors were calculated after all the classes had been trained. A different picture emerges if we introduce feature vectors in the plot, whose images haven't been trained. These new images appear on the plot as the outliers in every class (figure 5.2) . It can clearly be seen that two images that actually belong to respectively class one and class two are wrongly classified in class two and class one. Still, class zero can be clearly separated from the other two.
There are a couple of reasons for this. The number of images trained is, of course, very small. In fact, when one looks at the images used (figures 5.3 and 5.4, class one and class two share about one third of their image content. This means it is only possible to differentiate between the two using the remaining two thirds of the image.
Training more images might solve a lot of these problems. Therefore, we used another, a much larger set of images of the same scene. Three classes, each consisting of 10 images, were trained with 6 or 8 images. Thereafter, the featurevectors of all 10 images were calculated. The results, shown in figures ?? show, that training more images makes the spread of the featurevectors smaller, and the classification more reliable. Furthermore, the type of features also can have a profound impact on the separability. When one looks at the images used one can get the idea that although the images of class one contain much more fine detail than the images of class two, their mean pixel values might approximately be the same. As mentioned, the area of the image in which one can

differentiate between the two classes is only about two thirds of the image. This is largely composed of large homogenous areas. Therefore, it might be a good idea to use autocorrelation in stead of autocovariance features. This makes the results a lot better; in figure 5.7 one can clearly see three well separated classes.

The use of a 5x5 grid for the autocorrelation- or covariance functions gave approximately the same results. However, because the use of a 5x5 grid is computationally more expensive, it is desirable to use a 3x3 grid when the results are comparable.

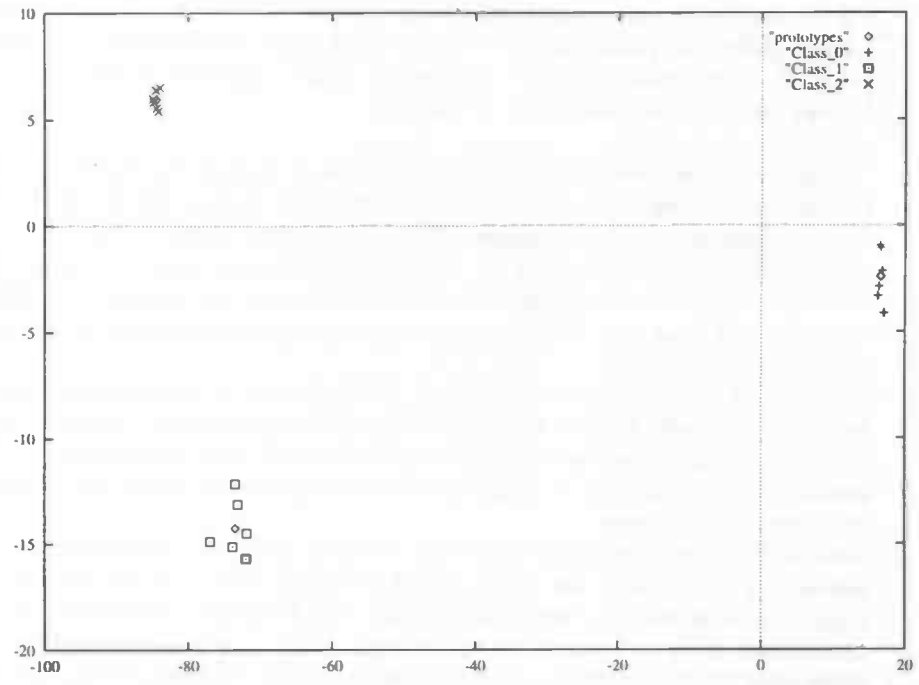


Figure 5.1: Plot of feature vectors of three classes after all images have been trained.The plot was generated using a small set of mono images with 3x3 autocovariance features, an image pyramid with 6 scales and half octave scale distance and a Laplace filter.

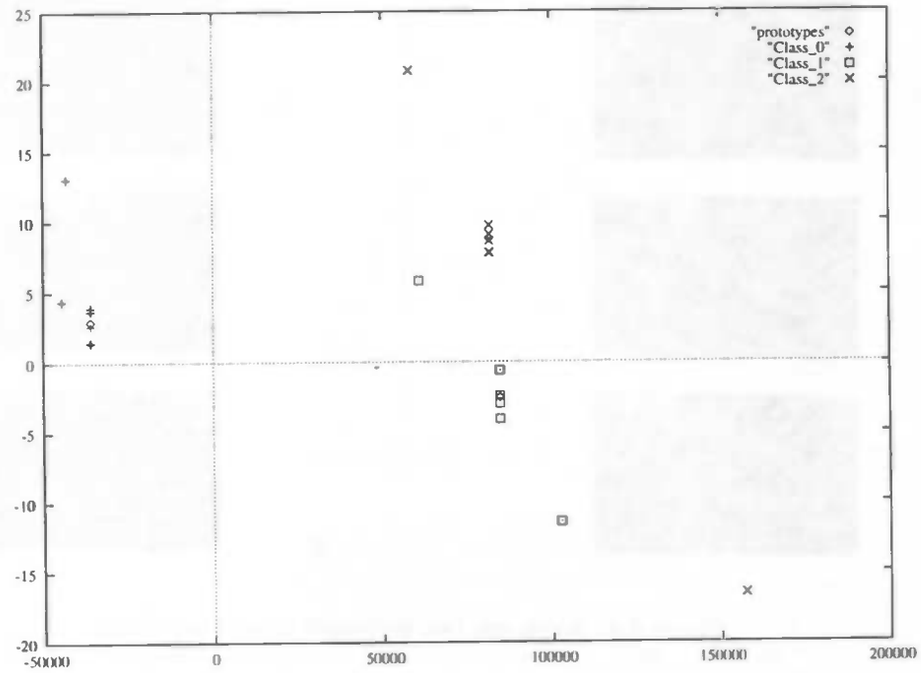


Figure 5.2: Plot of feature vectors of the room set (figures 5.3 and 5.4). Per class, four images have been trained, and two are solely used to calculate feature vectors from. It can be clearly seen that the separability is not so good.

Figure 5.3: Room set; four images per class.

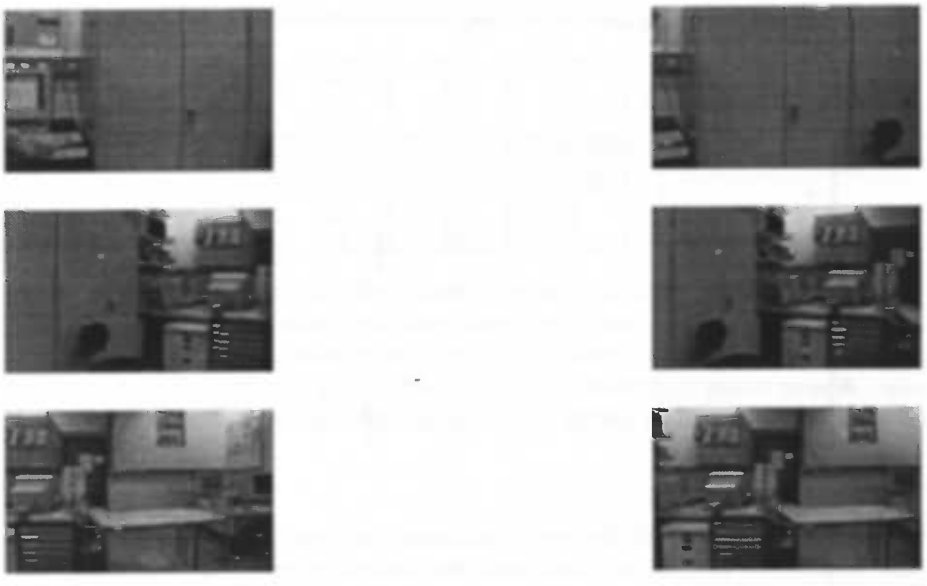
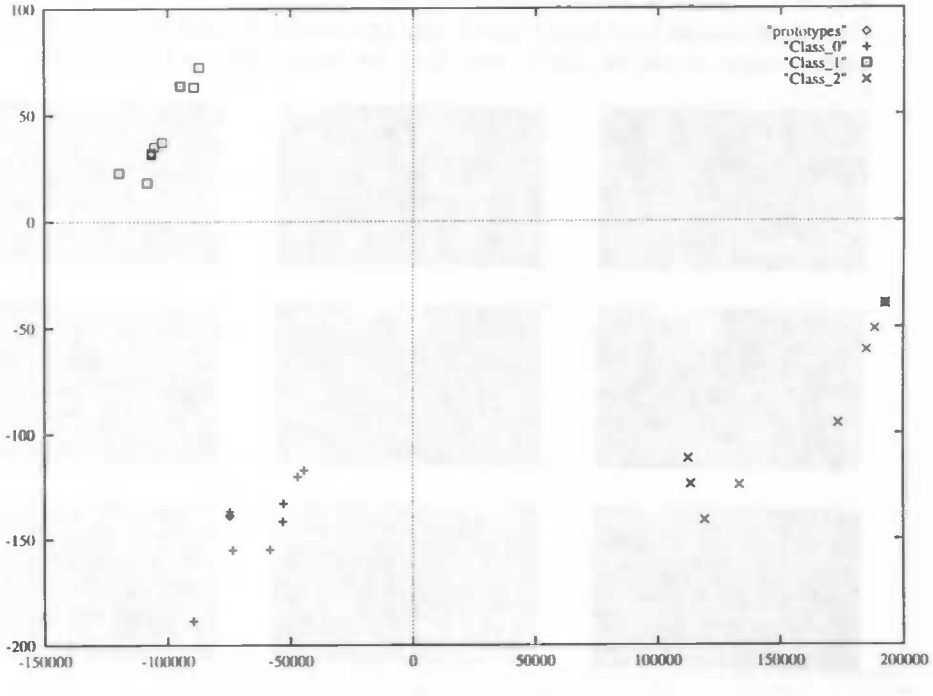


Figure 5.4: Room set; two untrained images per class.

Figure 5.5: Plot of feature vectors of all images, after 6 images had been trained.



5.2 Recognizing Walls

To get a better idea of the performance when using stereo images, it was tested whether the classifier could classify various views of walls in a room. Therefore,

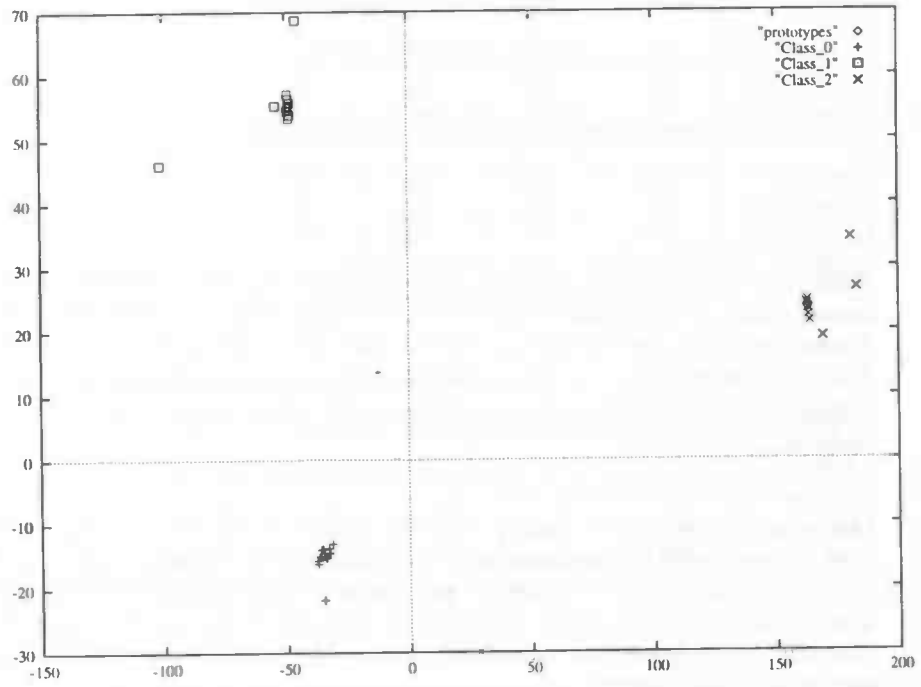


Figure 5.6: Plot of feature vectors of all images, after 8 images had been trained.

33 sets of 10 images were taken, each set with varying head angles. The sets can be divided in 5 classes (figure 5.8).

To test whether the classifier was in principle capable of discriminating between the five classes, half of the images per set were used to train a classifier, the other half were used to test. Only 6 scale pyramids were used; because the images per class were taken from different distances, such a pyramid was needed. Classification was made with mono images, and with the two methods of processing stereo images. This gave the very good results of table 5.1. Of course, images of every set were trained, but it shows that the classifier is not confused by the various distances and rotation angles that were present in each class. After this, only one set per class was trained. In an application of the classifier, this will most likely be the case. In addition to training 5 images per set, we also trained 10 images per set. The sets that were not trained were used as testset. Again, various configurations were tested. The results are shown in table 5.2

5.3 Cutting out Parts of Images.

As remarked in 4.1, the classification could perhaps be made better by extracting a fixed area out of the image and classifying this. To test whether pursuing

	ACF3x3 oct.	ACF3x3 1/2 oct.	ACF 5x5 oct.	ACF 5x5 1/2 oct.
Mono	99	99	99	100
Stereo, conc.	99	99	100	100
Stereo, add.	98	98	100	100
	ACV 3x3 oct.	ACV 3x3 1/2 oct.	ACV 5x5 oct.	ACV 5x5 1/2 oct.
Mono	100	100	83	92
Stereo, conc.	100	83	92	92
Stereo, add.	100	100	99	97

Table 5.1: Results from the first Foyer test with 6 scale pyramid. Autocorrelation and autocovariance functions were both used, with 3x3 and 5x5 displacement grids, and octave as well as half octave distances between scales. Furthermore, the test was done with the feature vectors of mono images, concatenated feature vectors of stereo images, and added feature vectors of stereo images.

this approach would be viable, manually pieces of the images were selected that corresponded to approximately the same area in the real world, and were centered in the image. A classifier was trained, using images that consisted of a solid background, in which the selected areas of the images were pasted. In this way it was hoped the classification would be less prone to failures due to untrained image content.
The results can be seen in table 5.3; the difference between the cut-out and the

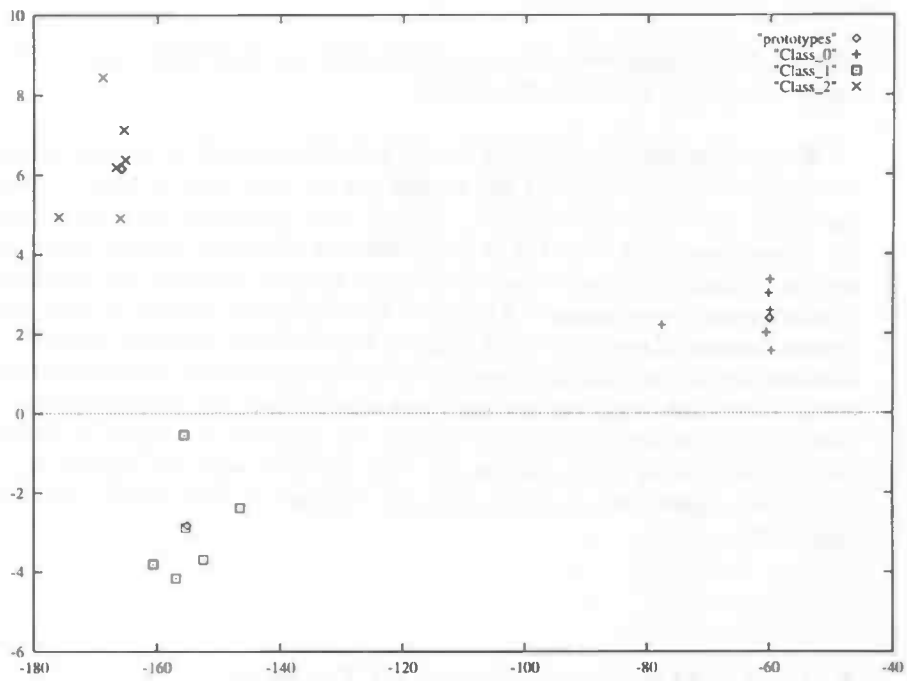


Figure 5.7: Plot of feature vectors of the room set (figures 5.3 and 5.4) after all images have been trained, using *autocorrelation* features

	ACF3x3 oct.	ACF3x3 1/2 oct.	ACF 5x5 oct.	ACF 5x5 1/2 oct.
Mono	60	62	71	65
Stereo, conc.	85	72	71	82
Stereo, add.	80	72	84	81
	ACV 3x3 oct.	ACV 3x3 1/2 oct.	ACV 5x5 oct.	ACV 5x5 1/2 oct.
Mono	61	45	50	34
Stereo, con.	23	54	37	63
Stereo, add.	60	60	24	60

Table 5.2: Results from the second Foyer test with 6 scale pyramid, 10 images trained. The same classifier configurations as in 5.1 were used.

complete images does not appear to be very large.

	ACF3x3 1 sc.	ACF3x3 3 sc.,oct	ACF 3x3 3 sc.,1/2	ACV 3x3 1 sc.	ACV 3x3 3 sc.,oct	ACV 3x3 3 sc.,1/2
Cut out images: :						
Mono	50	64	72	44	74	69
Stereo, conc.	66	64	72	50	75	69
Stereo, add.	76	65	72	73	76	71
Complete images:						
Mono	58	70	67	72	74	69
Stereo, conc.	59	71	67	44	63	72
Stereo, add.	66	68	72	76	74	76

Table 5.3: Recognition rates using cut out images vs. whole images. 10 images of each class of the foyer set were trained, using autocorrelation and autocovariance functions with a 3x3 displacement grid. 1 scale and 3 scale pyramids were used. For comparison, also complete images were classified using the same configurations.

5.4 Recognizing Rooms

For a real world test of the performance of the classifier rooms were classified. In this task the position of the robot is fairly certain, because all attempts made will be made in the door opening. From a behavioral viewpoint, this is the most sensible place to perform a recognition; the result of it will mostly be used to make a decision whether to enter the room, so we don't want the robot to be inside yet. Furthermore, an overview of the complete room can only be conveniently obtained in the dooropening. A set of 5 images from 10 rooms was taken, at head angles of -10, -5, 0, 5, and 10 degrees. The images were subsampled to 360 x 220. Classifiers were trained with the -5 en 5 degrees images, and tested on the -10, 0 and 10 images.
Because the angles turned out to be fairly large, the problem mentioned of images containing to much different content (section 4.1)arose here, which was especially apparent in the horizontal direction. Therefore the images were cropped to a width of 270 pixels. Also the images were again subsampled (to 180 x 110 respectively 135 x 110). The subsampled images were too small to be used in a 6 scale pyramid. Feature vectors based on stereo as well as mono images were used. Each was tested with the 10 possible configurations of the system. Also, to see how good the Bayesian classifier performed in comparison to a classifier based on pure Euclidean (Least Mean Square) distance, the same test

was done by a Nearest Neighbour classifier. This gave the results from tables 5.4

As can be seen from this table, this kind of recognition can be very good. Some strange results occur however; for the subsampled images, for instance, 6 scale pyramids perform badly sometimes. This is because the undersampling can give unwanted effects if the image is very small. The 1/2 octave, 3 scale pyramid has the same problems with some images; it's scaling produces unwanted artifacts or makes the features too dependent on specific frequencies. A one scale pyramid also suffers from too small images; the feature vectors become too dependent on precisely the right frequencies in the image. Eventhough, subsampling does not seem a very bad idea; this is because the autocorrelation is determined with a pretty narrow mask. Because of this, more structural information can be represented in the features. Also the beneficial effect of cropping can be clearly seen.
The use of stereo images to make featurevectors can have strange effects. While it is better for some type of images, it is bad for other types. Because the head angles were a little too high, the content of the left and right images sometimes differed significantly.

Figure 5.8: The Foyerset; 33 sets belonging to 5 classes. Only one image per head angle per set is shown.

Baysian:

	ACF3x3 1 sc. oct.	ACF 3x3 3 sc. oct.	ACF5x5 3 sc. 1/2 oct.	ACF3x3 6 sc. oct.	ACF5x5 6 sc. 1/2 oct.
Mono:					
Full Image	30	53	70	90	90
Cropped	87	87	83	86	90
Subsampled	87	90	93	–	–
Subs./cropped	33	93	96	–	–
Stereo add.:					
Full Image	63	100	43	67	43
Cropped	67	30	67	43	33
Subsampled	97	93	43	–	–
Subs./cropped	33	70	23	–	–
Stereo conc.:					
Full Image	20	90	53	87	63
Cropped	93	60	43	20	93
Subsampled	93	93	20	–	–
Subs./cropped	27	97	93	–	–
	ACV3x3 1 sc. oct.	ACV3x3 3 sc. oct.	ACV5x5 3 sc. 1/2 oct.	ACV3x3 6 sc. oct.	ACV5x5 6 sc, 1/2 oct.
Mono:					
Full Image	33	30	47	87	63
Cropped	80	80	30	83	46
Subsampled	77	13	53	–	–
Subs./cropped	70	20	10	–	–
Stereo add.:					
Full Image	87	43	83	57	33
Cropped	67	37	63	63	60
Subsampled	63	33	63	–	–
Subs./cropped	70	67	43	–	–
Stereo conc.:					
Full Image	47	60	33	57	23
Cropped	70	80	80	80	43
Subsampled	30	33	73	–	–
Subs./cropped	47	40	37	–	–

Euclidean:

	ACF3x3 1 sc. oct.	ACF 3x3 3 sc. oct.	ACF5x5 3 sc. 1/2 oct.	ACF3x3 6 sc. oct.	ACF5x5 6 sc. 1/2 oct.
Mono based:					
Full Image	50	50	50	50	50
Cropped	43	43	43	43	43
Subsampled	43	43	43	–	–
Subs./cropped	43	43	43	–	–
Stereo add:					
Full Image	10	10	10	10	10
Cropped	16	16	16	13	13
Subsampled	10	10	10	–	–
Subs./cropped	16	16	16	–	–
Stereo conc:					
Full Image	47	53	53	53	53
Cropped	47	43	50	43	47
Subsampled	47	43	50	–	–
Subs./cropped	47	43	50	–	–
	ACV3x3 1 sc. oct.	ACV3x3 3 sc. oct.	ACV5x5 3 sc. 1/2 oct.	ACV3x3 6 sc. oct.	ACV5x5 6 sc, 1/2 oct.
Mono based:					
Full Image	57	50	50	47	50
Cropped	40	40	40	37	40
Subsampled	40	40	40	–	–
Subs./cropped	40	40	40	–	–
Stereo add:					
Full Image	10	10	10	10	10
Cropped	13	16	16	16	16
Subsampled	10	10	10	–	–
Subs./cropped	16	16	16	–	–
Stereo conc:					
Full Image	47	40	40	40	40
Cropped	30	33	33	33	33
Subsampled	30	33	33	–	–
Subs./cropped	30	33	33	–	–

Table 5.4: Results from the first roomtest with 3x3 Mask

Chapter 6

Dynamic Behavioral Classifier Module

6.1 Scenario

The classifier was used in the handling of a sequence of events controlled by the dynamic systems architecture from section 2.1. The goal of the sequence was to navigate to the right door. In order to do this, the robot made a visual search of its environment. It used edge extraction and stereo vision to spot a door, which consisted of two edges placed sufficiently far apart. It then pointed at the door with its arm, to show he had spotted it. After this, the classifier would be activated. It would receive a picture of the door and classify it. If it was the right door, the robot would ride to it. The whole experiment is described in [23]. For this task, the classifier was integrated in a *Dynamic Behavioral Classifier Module*.

6.2 Using the Classifier in the Dynamical Systems Control Architecture

As explained in 2.1, in this architecture the various behaviours receive an activation value, determined by the behavioural dynamics of the complete system. For each behaviour, one has to specify wich behaviors it presupposes, with wich behaviors it competes. The recognition behavior needs to become active when a door is found. We therefore define a behaviour ***Recognition***. This behaviour presupposes the tracking of the door, which is the behaviour that becomes active when the *Visual Search* behavior has found a door, and is responsible for keeping the door in sight. Furthermore, the sensorial context (which is the term that represents input from the environment relevant for the activation of a behavior) of the Recognition behavior is always 1; this means it can become active whenever its presupposition is also active. There is no additional sensorial context that should be present for the classifier to become active. If the task was extended, one could include a sensorial context, for instance the output of a obstacle avoidance behavior; it is no use to classify images that are taken while a robot is turning and moving to avoid an obstacle. However, this is not

essential in our scenario.
The recognition behavior consist of acquiring a picture with the current head angle and classifying it. The ***Tracking*** behavior keeps the head pointed at the door. If it is not the the right door, this behavior has to be switched off. This means we define a behavior ***Door 1 Not Recognized***, which is to become active when the right door is *not* recognized, and specify a competition for this behavior with the *Tracking* behavior. The recognition module sends a message which sets the context of the *Door 1 Not Recognizd* behavior to 1 if it didn't recognize the right door, which will ensure this behavior will become active, which will ensure the Tracking behavior becomes inactive. This in turn will cause the *Recognition* behavior to switch of too, because it presupposes the *Tracking* behavior. The default behavior *Visual Search* will then become active again and try to find a new door.
If the right door is recognized, the ***Local Navigation*** behavior, which moves the robot to the door, has to become active. Therefore we define a behavior *Door 1 Recognized* . This behavior becomes active when the right door is recognized, because the recognition module then sets the context for this behavior to 1. This behavior is a presupposition for the *Local Navigation* behavior.
Furthermore, a behavior ***Not Active*** is defined whose context is 1 when the recognition module is not active or didn't recognize the right door; consequently, this behavior will be active at these instances. We didn't actually use this behavior in the experiment, but when the sequences become more complex it might be desirable to have such a behavior.
To show the robot had spotted a door, we included a ***Pointing*** behavior, which would also become active when a door is found. However, when the robot is pointing at the door, a significant part of the image will be filled with the arm. This makes recognition difficult; it is not possible to exclude the arm out of the picture, because it obscures relevant parts of the image. Therefore the recognition module presupposes another behavior, which is ***Arm Safe***. This is the behavior that becomes active when the arm is in its parking position next to the body of the robot.
The dynamic behavioral recognition module is a module that receives the activation value of the recognition behavior, and starts the classifier when this activation is above a certain threshold. On the behavioral level, it consists of four behaviors, that can be easily integrated in the rest of the behavioral architecture.

6.3 Stabilizing the Recognition with a Low Pass Filter

The *Recognition* behavior is constantly active when a door is being tracked by the *Tracking* behavior. This is necessary, because the *Door 1 Recognized* behavior is a presupposition for the *local navigation* behavior. However, there can be small disturbances during its operation. For example, people can walk through the image field, lighting of the situation may change, etc. Therefore the recognition module must be able to allow a misclassification once in a while, and still keep sending the *Door 1 Recognized* context. On the other hand, when it has classified the image as being another door for a while, one can be fairly sure it

really is another door, and the *Door 1 Not Recognized* context has to be sent. To achieve this, a low-pass filter is used on the sequence of distance vectors the classifier determines.
When a potential misclassification occurs, the distance vector, which is the vector containig the distances of the image to the various classprototypes, will undergo a sudden change. The distance to the prototype of another class will be the smallest. However, the low pass filter will filter out such a sudden change, so the classification will still be in the original class. If several potential misclassifications occur, the filtered distance vector will change eventually to a new distance vector, with the new distances, so now the classification will be in another class.

Figure 6.1: Arnold at various stages in the scenario.

6.4 Results

The scenario was run using the four recognition behaviors. A classifier was used which contained two classes for two different doors, one class trained with images in which the door extractor in the Visual Search behavior wrongly found doors, and a last class trained with images which had nothing to do with doors. As can be seen from figure 6.3, where the activation values of the relevant behaviors are plotted, the robot was perfectly able to perform the scenario.
In figure 6.2 the distances of the image to the four trained classes is plotted. The three images used were taken at the moments the door extractor found doors[1]. It can be clearly seen that class 4, that had nothing to do with doors, was at a great distance to all of the images. Also, the images were obviously the closest to the classes they should belong to.

[1]Strictly speaking, the door extractor found only two edges with a certain distance to each other.

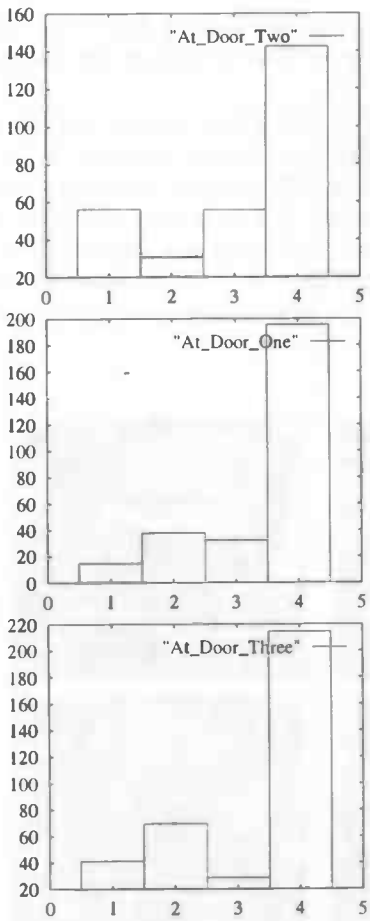


Figure 6.2: Distances to the four trained classes of the images that were taken at the moments the door extractor found doors. Classes 1 and 3 are the doors. Class 2 is not a door, but a spot at which the door extractor always found doors. Class 4 is a waal that has nothing to do with doors.

Only after nearing the target door significantly did the recognition get unreliable; this is caused by disappearence from certain structures out of the image. A door can of course best be recognized by the structures around it. The door itself will consist of a one coloured panel, with not much variation. The structures around it cannot be seen if the robot is too close. This is not a big problem; If the door really was to be passed, at this moment another behavior would be needed to guide the robot through the door. The recognition behavior could be switched off at this moment.

Running the scenario from different initial positions did not significantly change these results, although more time was needed. This was partly due to the initial edge extraction based door recognition, wich sometimes failed. From some initial positions however, the right door was only classified as such after a second pass.

Care had to be taken in training the classifier. From some initial positions, light reflections began to appear in the images which were not present in the original images. This seriously disrupted the classification. Training the classifier with images both with and without the reflections solved this problem.

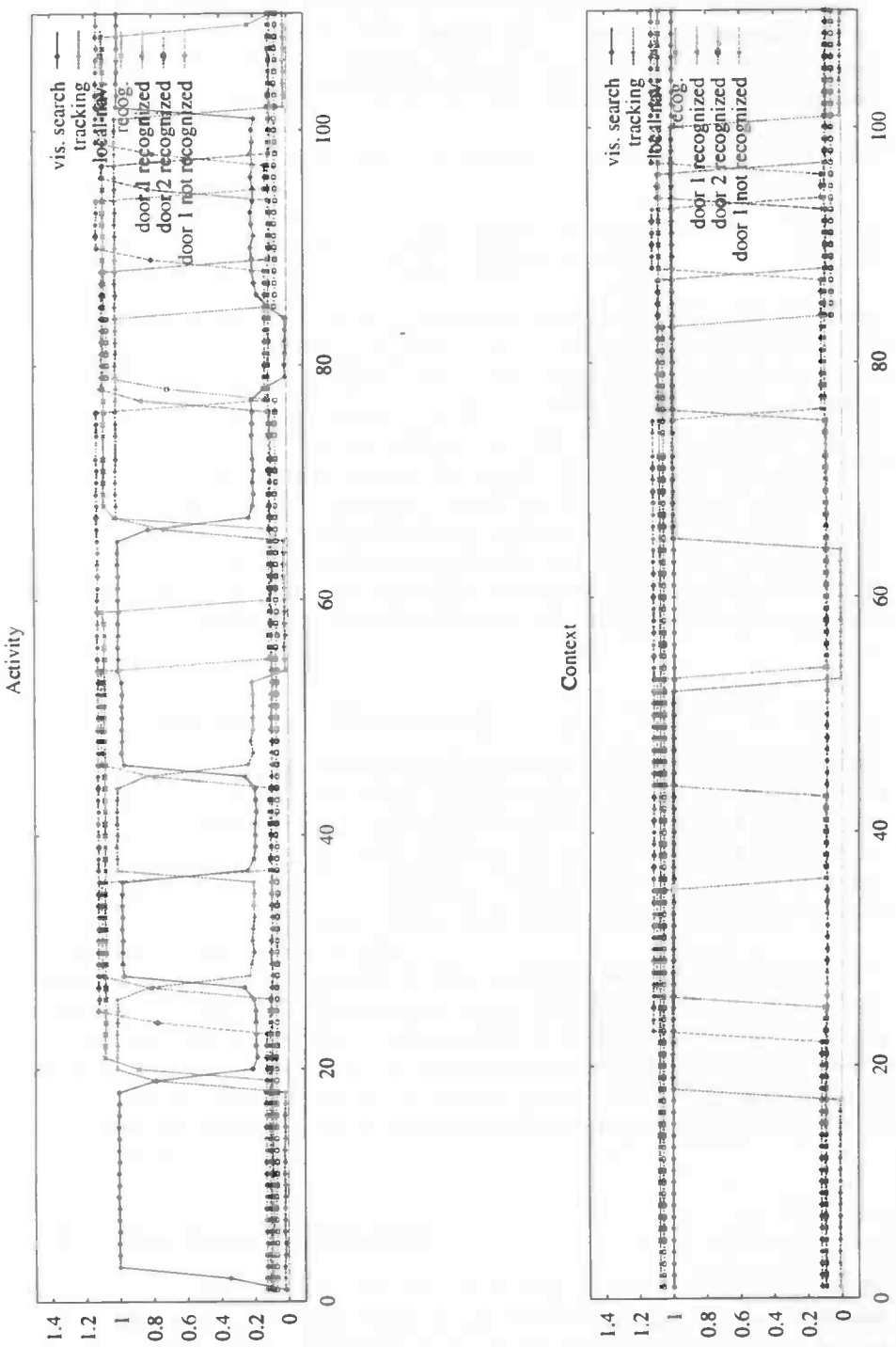


Figure 6.3: Plot of the activity and context values for the room scenario. Only the most relevant behavioral variables are plotted to keep the plot from becoming too crowded. The activation of the *Recognition* and *Tracking* behaviors after the *Visual Search* has found a door can clearly be seen.

Chapter 7

Autonomous Classifying

7.1 Unsupervised Learning

An autonomous robot has to be able to learn from its environment. In the previous application the classifier was used with pretrained classes. However, it might be needed to let the robot learn its own classes. Therefore one would need a classifier which is able to learn *unsupervised.*
What we need therefore is a kind of clustering method.

7.1.1 Maximum Likelihood Methods

Maximum Likelihood Methods [9, 1, 11] estimate the prototype vectors by maximizing the chance that the set of samples was drawn from the specific set of prototype vectors. Usually, this is done by estimating a prototype vector as an average of the samples, weighted by the the chance that a sample belongs to the class. As estimates for these chances frequency ratios are used. It also can be necessary to use sample covariance matrices and sample means, when the mixture density covariance matrix is not known. The calclulation of the maximum likelihood estimate can be actually quite computationally expensive. If one can make fairly good estimates for the unknown means, it can be done by an iterative, gradient ascent-like procedure. If such estimates can not be made, the solution becomes more complex; every possible solutions then has to be checked for its likelihood.

7.1.2 Unsupervised Bayesian Learning

Analogous to our Bayesian classifier we can use **Unsupervised Bayesian Learning**[9, 19]. We have a set H of N sample vectors $\mathbf{x}_i$. We want to estimate θ, the vector of prototype vectors. We shall use the samples to calculate the a posteriori density $p(\theta \mid H)$. Using Bayes' rule (eq. 2.11), we can write:

$$p(\theta \mid H) = \frac{p(H \mid \theta)p(\theta)}{\int p(H \mid \theta)p(\theta)d\theta} \tag{7.1}$$

where, because we assume independence of the samples,

$$p(H \mid \theta) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid \theta) \tag{7.2}$$

This equation can also be written in an recursive form, where H^n is the set of the first n samples:

$$p(\theta \mid H^n) = \frac{p(\mathbf{x}_n \mid \theta)p(\theta \mid H^{n-1})}{\int p(\mathbf{x}_n \mid \theta)p(\theta \mid H^{n-1})d\theta} \tag{7.3}$$

If the mixture density $p(\mathbf{x} \mid \theta)$ is identifiable, which means this density is not the same for different θ's, using equation 7.3 will sharpen $p(\theta \mid H^n)$ with every additional sample. It can be shown, that under fairly general conditions it converges to a dirac δ-function centered at the true value or θ. If the identifiability condition does not hold, this convergence can still happen, but it assures by no means the result can be used outside the sample set. In other words, $p(\mathbf{x} \mid H^n)$ may still converge to $p(\mathbf{x})$, but $p(\mathbf{x} \mid \omega_i, H^n)$ will in general not converge to $p(\mathbf{x} \mid \omega_i)$. Since we assume identifiability however, we will not consider this a major problem. This looks rather promising for our case, since it enables incremental learning of θ. There are, however, some problems associated with it. It is still computationally very complex, because all samples have to be processed for all classes.

7.1.3 Basic Isodata Clustering

Another method of clustering is **Basic Isodata or Nearest Neighbour Clustering**, which is very fast and simple. In this method, a set of prototype vectors is needed. The distance of this prototype vector to every sampel vector to be clustered is then determined. This distance can be the Euclidean distance, or another distance that is more convenient. Then, the vector is clustered in the cluster of the nearest prototype vector, and this prototype vector is shifted in the direction of the sample vector.
Although this is computationally a very attractive method, it still has some problems. We still need all prototype vectors beforehand (or at least an estimate of these vectors) and it is not possible to introduce new classes during learning. This also means that if a sample vector does not belong to the initial clusters, the prototype vector of the nearest cluster will still be shifted in the direction of this sample vector. If this occurs often enough, the resulting clusters will be useless.

7.2 On Line Learning

We want our robot to learn new classes as they appear, and we want to do that rather quickly. We don't have all our samples and the number of classes beforehand. Therefore, what we need is not just unsupervised learning, but *on line* learning.
Kohonen [12] used lateral inhibition in a neural network and was able to generate topologically correct mappings of the inputvectors to (only one or two) dimensional outputvectors. For instance, he created tonotopical maps when he

fed his network the ouputs of resonators with various frequencies sequentially. Grossberg [7] introduced a kind of verification step for the basic isodata clustering in his ART-1 and subsequent models. If a sample is too far away from every prototype, a new class is formed.
In Grossberg's models, this method is implemented in a two layer network. The weights going from the input layer to a specific unit in the output layer can be seen as a prototype vector belonging to that unit. The network contains a feedback loop, which enables it to let the most active unit in the second layer, which represents the 'winning' cluster, to send a pattern back to the first layer. The first layer compares the two patterns, and is able to set the second layer unit's activity to zero if the patterns are too far different. How big this critical diffence is can be configured through a parameter. The weights going from the first to the second layer are modified through Hebbian learning. The feedback weights from the second to the first layer are also modified through Hebbian Learning, which ensures they will converge to the input pattern if a unit in the second layer is active long enough. An Adaptive Resonance state of the network ensures that this is the case when a new cluster is formed or an old cluster is recognized.
To put it another way, the idea is to divide an N-dimensional space into regions. These regions have the already learned class prototypes as their centers. If a new sample is outside these regions, this means it must be a new class. We can then make a new region around this sample for the new class. We only need to specify the borders of the regions and the distance measure to be used.
This kind of clustering can function on-line, and is computationally feasible. We already have a distance measure 2.13 which worked out very well in the supervised experiments. Also, 5.1 shows that with these features, linear separable classes can be discriminated. The Mahanalobis distance seems a good candidate for a distance measure. The only thing left is to specify the borders.

7.2.1 Within Class Borders

If the classes are not very similar, and the vectors can be clustered in the neighborhood of the prototypes, one could use the mean distance of the images in a class to their prototype as a border measure. Of course, the mean distance has to be increased by a certain amount to include the far lying vectors. This amount has to be chosen somewhat arbitrarily; it is dependent on the structure of the clusters and vectors. In the experiments, this amount was two or three standard deviations. This approach can be succesfull if the within class distances are much smaller then the between class distances, but suffers from the fact that it introduces a parameter that can only be determined by trial and error; the amount with which the mean within class distance has to be increased.

7.2.2 Between Class Borders

If one can assume that the class prototypes have about the same distances to each other, one can use the mean between class distance for a classborder. If a featurevector is farther away from a prototype than the prototype's mean distance to the other prototypes, one can assume it does not belong to that class. This approach is prefereable over the use of borders based on within class distances because of the following reasons:

- Because of the transformation used to reduce dimensionality, it is highly possible that a new feature vector would lie well beyond the mean within class distance, while still being closer to the prototype than to any other prototype. With consequent updating and recomputing of the transformed prototypes, the new vector will eventually be within these borders, but at its first presentation this might not be the case.
- It is computationally more simple. One just has to compute the distances between the class prototypes, and it is not necessary to store all feature vectors.
- The determination of a arbitrary parameter, the amount with which the mean within class distance has to be increased, is avoided.

7.3 Real Life Experiment

A test was made in the vision lab. This room is rectangular and relatively small. The robot was positioned on one end of the room, so the image content was fairly constant over a large angle on one side (facing the room), while changing rapidly with the angle on the other sides (facing the walls). Starting with an empty classifier, Arnold made pictures at pan-angles (horizontal head angles) of -90.0, -45.0, 0.0, 45.0, and 90.0 degrees. The last two angles faced the room, while the first three faced the walls. These pictures were fed to the clustering algorithm. If a new class was made, an additional 4 pictures were taken and trained under that class, at -4.0, -2.0, 2.0 and 4.0 degrees of the original angle. This turned out to be necessary, because otherwise the determination of the Mahanalobis distance would not be reliable. This distance is dependent on the covariance matrix (see 2.3.1), which needs at least a couple of samples.
This sequence was done once, after which the pan-angles were decreased by 5, 10, 15 and 20 degrees, and the sequence was repeated. Both between- and within-class distances were used.

pass	degr.	class	degr.	class	degr.	class	degr.	class	degr.	class
1	-90.0	0	-45.0	1	0.0	0	45.0	2	90.0	2
2	-85.0	0	-40.0	1	5.0	0	50.0	2	95.0	2
3	-80.0	0	-35.0	1	10.0	0	55.0	2	100.0	0
4	-75.0	0	-30.0	0	15.0	0	60.0	2	105.0	1
5	-70.0	0	-25.0	0	20.0	0	65.0	2	110.0	0

Table 7.1: First unsupervised clustering test in the vision lab, using between class borders. 5 passes were made. With each pass, pictures were taken at pan angles 45 degrees apart. These angles were decreased by 5 degrees every pass.

Using between class distances, at the -90.0 and -45.0 degrees pictures new classes were created (Classes 0 and 1). At 0.0 degrees however, Class 0 was again recognised. This was due to the fact that the images of Class 1 mainly contained a locker, which consists of big surfaces separated by long edges, while the images of Class 0 contained much more fine detail (computer screens, tables with things

lying on them, chairs etc.). The images at 0.0 degrees had a similar content. If more classes would have been trained, the classifier would probably have been able to differentiate; later on this was tested. At -45.0 and -90.0 degrees a new class was made (Class 2).
At the following passes, the recognition (and updating) of Class 0 at 0.0 degrees proved to be a disadvantage. In principle, the classifier mostly was discriminating using the differences between images containing big surfaces (Class 2 also contained some of them) and images containig much detail. This resulted in Class 0 being recognised too often, as table 7.1 shows. The fact that Class 1 and Class 2 were only recognised in a fairly small range was not that surprising, because the image content changed significantly once outside these ranges.

pass	degr.	class	degr.	class	degr.	class	degr.	class	degr.	class
1	-90.0	0	-45.0	1	0.0	2	45.0	3	90.0	4
2	-85.0	0	-40.0	5	5.0	6	50.0	7	95.0	4
3	-80.0	8	-35.0	9	10.0	10	55.0	0	100.0	11
4	-75.0	12	-30.0	13	15.0	14	60.0	15	105.0	16
5	-70.0	17	-25.0	18	20.0	19	65.0	20	110.0	16

Table 7.2: First unsupervised clustering test in the vision lab, using whithin class borders. Again, 5 passes were made, and with each pass, pictures were taken at pan angles 45 degrees apart; these angles were again decreased by 5 degrees every pass.

Using Within class borders made the situation worse; at almost every angle a new class was made (see table 7.2). The classes had too much variation in themselves to be properly clustered using these borders. This is not so surprising; to set the amount the mean within class distance has to be increased is highly dependent on the structure of the classes. This parameter can not be set right beforehand, unless one knows much about the images and classes to be processed. Using within class borders proved to be a bad idea.

No so many new classes were created in the between class borders test. Of course, whether this is a disadvantage is dependent on the task to be performed with the classes; it might be that one only needs a class for a specific part of the room (for instance a door). In that case, the results can be very usefull. However, in some tasks one might need more classes. (For instance because there are many regions of interest in the room, or because the classes are going to be used to calibrate a navigation system).
The reason that not so much new classes were created is that the distances between the classes or the classes themselves are too big. If in the beginning some images are trained under the wrong classes, the class becomes so big that it contains almost every possible image. Furthermore, the distance to other classes can also become very large. This means almost all images will fall within the between class distance of some class. The discriminating power of the classifier greatly decreases. This problem could be solved if one has some initial classes trained, that are not much alike. Therefore, in a new test, four classes were trained first, at -180.0, -90.0, 0.0, and 90.0 degrees. After this, images were clustered at 45.0 degrees intervals. The starting point for these sequence was

increased with steps of 5.0 degrees. Only Between Class Borders were used; not only is this conceptually and computationally the preferred border strategy, but the previous test also showed that Within Class Borders performed very bad. The results can be seen in table 7.3.

Using some pretrained classes clearly made the clustering more reliable. Still

pass	degr.	class	degr.	class	degr.	class	degr.	class
1	-180.0	0 *	-135.0	0	-90.0	1 *	-45.0	4
2	-175.0	0	-130.0	6	-85.0	1	-40.0	4
3	-170.0	0	-125.0	6	-80.0	1	-35.0	4
4	-165.0	0	-120.0	6 -	-75.0	1	-30.0	4
5	-160.0	0	-115.0	1	-70.0	1	-25.0	2
1	0.0	2 *	45.0	5	90.0	3 *	135.0	3
2	5.0	2	50.0	5	95.0	3	140.0	3
3	10.0	2	55.0	7	100.0	8	145.0	9
4	15.0	0	60.0	5	105.0	3	150.0	0
5	20.0	2	65.0	5	110.0	0	155.0	9

Table 7.3: Second unsupervised clustering test in the vision lab, using between class borders. The procedure followed was similar to the previous two tests, except more pan-angles were used, and four classes were trained beforehand (marked with a *)

some errors occured. Class 0 was still recognized in the wrong places; again, the problem here is that the class can become too big if it is wrongly recognized (and updated) too often. Also, some new classes are created on spots an old class should be recognized (classes 7 and 8). These classes are subsequently not recognized any more.
The interpretation of other errors is not so straightforward. Class 6, for instance, consisted of images of a specific part of a desk. Although from the table it looks like an error, actually the creation and recognition of the class would be very good if used for a task in which fine discrimination would be needed.

The problem in using autonomous classifying is that the results are dependent on the task. As remarked, it might be an advantage or a disadvantage to cluster into many new classes. To make sure the classifier performs as needed, trials are necessary with or without pretrained classes to determine which configuration performs best.

Chapter 8

Summary and Concluding Remarks

In this project a system was developed that is able to classify visual contexts for an autonomous robot. It offers the capability to learn and recognize views in common indoor environments. The system is based on autocorrelation functions and Bayesian classifying. It functions without needing additional information about the contents of the images, and is most conveniently used in a supporting role, to augment other information. It can be used in this way to influence behavior selection and object redognition. For example, one can use an edge detector based scheme to detect all the doors around the robot. The context recognition system can then recognize the right door. The system can work in two modes. The classes to be recognized can be given beforehand; the system then determines the most likely class for an image. The classes can also be learned unsupervised; the system then makes new classes if the images are too different. The system was integrated with the behavioral architecture of an autonomous robot.

The image classifying is based on purely statistical methods. This turned out to be both an advantage and a disadvantage. The advantage is that it is general purpose, applicable to a wide range of tasks. No specific information about the task or the images is needed. Because of the use of image pyramids, it is reasonably translation invariant. The output of the system is very well suited to be used as supporting information. The disadvantage is that care has to be taken in using the system. Tasks must be structured. In the supervised mode, the system always classifies an image. This can give unreliable results if images are classified that have nothing to do with the task. Also, when images do have to do with the task, their content must not be too different from the trained images of their class. If a significant part of the image is filled with a new structure or object whose autocorrelation functions are different, this will influence the autocorrelation features calculated from the image. This occurs, for example, when in an image of a wall a poster is present, with much fine detail, while the trained images contained only bare wall. The fine detail of the painting will result in different autocorrelation values. This can severely disturb the feature vector used for classification.

In the unsupervised mode, new classes are made when the images are too different. However, how different the images need to be is of course highly dependent on the task. Therefore, to use the unsupervised mode reliably rough classes have to be pretrained.

The system need not be limited to function in a supporting role. If applied with care, it could also be used for comparing images, or recognizing objects. When used in that way, care has to be taken that the images used contain not much more than the object.
The idea of using visual context information to augment other information can be extended. It could be used for a range of other tasks, for example for calibrating navigation systems, for improving object recognition, or for helping a visual search system. However, classifying visual contexts is not an easy task. The particular approach followed here leaves room for some improvements.
Other information could be used (for example shapes or edges present in the image, image histograms, or depth information). Which information would be the best to use is very task dependent. The system would be less general purpose and more complex, although the use of the system would require less structured tasks.
Also, it might be interesting to use other features than autocorrelations functions. However, which features would be good to use is not so straightforward. The classification still needs to be translation invariant, and autocorrelation functions are very well suited for such a requirement.

Chapter 9

Documentation

9.1 QNX programs

9.1.1 recog_module

Recog_module must be configured with a file named *'recog.cfg'* located in the same directory. For its content see 9.3. When running as a Planet proces, recog_module is activated by sending a recog_msg (see recog_module.h) This message must contain the following entries:

label has to be R. This label tells recog that it is receiving a recog_msg, and that it has activate.

mode Operating mode. can be one of four:

- **C** Clustering. Clusters the image in the nearest class within its classborders. If it is outside the classborders, a new class is made.
- **L** Classifying. determines the class closest to the image.
- **S** : Trains number images under class klass (in the same recog_msg) Warning: take care not to use classes higher then UsedClasses.
- **U** : Trains number (in the same recog_msg) images under the next available class.

take_pic if 1, recog_module takes it's own picture; if 0, it uses a picture sent to it.

number The number of images to be trained.

klass The klass to be trained.

phi horizontal head angle to take picture.

use_phi must be 1 if phi is to be used

After becoming active and setting mode, recog either sends a pic_request over channel picrequests, or waits till it receives a picture, depending on the take_pic entry. If use_phi is zero, a taken picture will be taken with the current vertical and horizontal head angles, otherwise the vertical head angle (theta)

will be 0 and the horizontal head angle will be phi. The picture is returned over channel 'view'. If training, (number -1) / 2 pictures on either side of phi are taken and trained; otherwise, only the first image will be clustered or classified. To classify an image, at least three classes have to be trained !!!
After completion, recog returns a result_msg (see recog_module.h) over the channel 'result' with the following entries:

label is R

klass klass wich was recognised, clustered or trained.

neu is 1 when a new class was encountered during clustering.

mode must be Q if all went well

9.1.2 pic_stereo

9.1.3 train

Used to train a set of images under the classes in their files. Classes can be set by setcl. Configured through configuration file (see 9.3). Usage:

train <conf.file> <trainfiles>

conf.file configuration file

trainfiles file containing Vista files to be trained.

9.1.4 clust

Used to cluster a set of images. Also configured through a configuration file (see 9.3) Usage:

clust <conf.file> <file>

conf.file configuration file

file file containing Vista files to set cluster.

9.1.5 setcl

Used to give a vista image file a class attribute with a given value. Usage:

9.1.6 recog_dyn

9.2 Solaris programs

9.2.1 cluster_test

9.2.2 Utilities

setcl Used to give a vista image file a class attribute with a given value. Usage:

setcl [-f] <file> <class>

-f Optional; Makes setcl set all files in file to class

file Vista file(s) to set class

class Classnumber between 2 and 200

getcl Used to get the the class of a vista image file, if it exists. Usage :

getcl <file>

file Vista file to get class from. (must have class initially)

delcl Used to delete the class attribute in a vista image file. Usage:

delcl [-f] <file>

-f Optional: Makes delcl delete classes from all files in file

file Vista file to delete class from

setdim Used to give a vista image file the xdim and ydim attributes and give them values [1] Usage:

setdim [-f] <file> <xdim> <ydim>

-f Optional: Makes setcl set all files in file to class

file Vista file to set class

xdim xdim value

ydim ydim value

replace Used to replace an entry in one or more configuration files. The resulting files can also be written to another directory. Note: The configuration files are assumed to be called `conf1.cfg, conf2.cfg, ..., confN.cfg`, with N being the **<no.>** argument. Usage:

replace <no.> <inpath> <outpath> <entry> <value>

no. number of conf files to be handled.

inpath path to the input conf files.

outpath path to the output conf files.

entry slot name whose value is to be replaced.

value value to put in slot

[1] These attributes can be used to only classify outcroppings of the images, see ??

9.3 Configuration file

All recog programs are configured through a configuration file. In the planet modules, this file is called *'recog.cfg'* and must be located in the same directory as the program itself. For the other programs, the file must be given as the first command line option. All entries in this file are optional; slot names and values must be separated by '='; setting for Features till use_confidence have defaults. Note: Stereo must be mentioned before use_classifier

Features Feature extraction method to be used; can be: *ACF_3X3, ACF_5X5,ACV_3X3 or ACV_5X5*

Scales Number of scales in image pyramid; can be *1,3* or *6.*

ScaleDist Distance of scales in piramid; can be *OCTAVE* or *HALF_OCTAVE*

FilterType Type of filter to be used; can be *GAUSS* or *LAPLACE*

Classes Maximal number of classes (alternatively, classes - 1 is the highest classnumber). Must be between *2* and *200.* [2]

Stereo How to handle Stereo Images. Can be: *0* (Use only left image or mono image), *1* (make one feature vector from stereo image), or *2* (make two featurevectors from stereo image).

AddFeatures When using Stereo images, either add the two featurevectors (*yes*) or concatenate them (*no*).

Mode How to handle feature vectors. Can be: *complete* for complete featurevectors, *transformed* for transformed feature vectors.

Border Choice Which Border strategy to use. Can be: *none* (classify in closest class), *between* (use mean between class distance per class as classborder) or *within* (use mean within class distance as border).

Update How to update the classes. Can be: *all* (update all classes) or *new* (only update classes which have not been explicitly trained)

Subsample Amount of subsampling to be used on an image. Can be: *0, 1* (no subsampling performed) , *2, etc* (one time subsampling or more).

LeftOffset,RightOffset,TopOffset,BottomOffset When present, specify a region to be cropped out of the image to be classified. When zero, the whole image is used.

use_classifier Filename of classifier to be used

save_classifier Filename where classifier is to be saved

logfile Filename of logfile to be used; if not specified, no logfile will be used.

save_distances Filename to save distances during classification; if not specified, no distances will be saved.

save_features Filename to save features during classification; if not specified, no features will be saved.

[2] This entry doesn't have any influence on the classifying, unless one gets more classes. Default value is 42, but can be set as one wants.

Bibliography

[1] R.J.Barlow. *Statistics. A guide to the Use of Statistical Methods in the Physical Sciences.* John Wiley and Sons Ltd. Chichester, UK, 1989.

[2] Thomas Bergener, Carsten Bruckhoff, Percy Dahm, Herbert Janssen, Frank Foublin, and Rainer Menzner. *Arnold: An anthropomorphic autonomous robot for human environments.* In Soave '97, Selbstorganisation Von Adaptivem Verhalten, 1997.

[3] Thomas Bergener, Carsten Bruckhoff, Percy Dahm, Herbert Janssen, Frank Foublin, and Rainer Menzner. *Adapted to Human Environments: An Anthropomorphic Autonomouns Robot for Common Service Tasks.* Submitted to: Autonomous Robots, Kluwer Academic Publishers, Boston, 1998.

[4] Thomas Bergener, Percy Dahm. *A framework for dynamic man-machine interaction implemented on an autonomous mobile robot.* In ISIE '97, IEEE International symposium on Industrial Electronics, 1997.

[5] V. Braitenberg. *Vehicles. Experiments in Synthetic Psychology.* MIT Press, Cambridge, Mass., 1984.

[6] R.A.Brooks. *A robust layered control system for a mobile robot.* IEEE Journal of Robotics and Automation, RA-2 (1), 1986.

[7] G.A. Carpenter and S. Grossberg. *ART-2: Self-Organizing of stable category recoginition codes for analog input patterns.* Applied Optics, Vol. 26, p. 4921, 1987.

[8] E.R.Davies. *Machine Vision.* Academic Press, London, 1990.

[9] R.O.Duda and P.E.Hart *Pattern Classification and scene Analysis.* John Wiley and Sons, New York, 1973.

[10] R.A.Fisher. *The use of multiple measurements in taxonomic problems.* Annals of Eugenics 7, pages 179 - 188, 1936.

[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* Academic Press Inc., 1990

[12] Kohonen, T. *Self-organized formation of topologically correct feature maps..* Biological Cybernetics, Vol.43, pages 59 - 69, 1982.

[13] Kosko, B. *Neural Networks and fuzzy systems. A dynamical approach to Machine Intelligence.* Prentice Hall, Englewood cliffs, NJ.

[14] Martin Kreutz. *Bilderkennung mittels Autokorrelationsfunktionen 2. Ordnung in pyramidalen Bildstrukturen.* Master's Thesis, Institut für Neuroinformatik, Ruhr-Universität Bochum, 1994.

[15] M. Kreutz, B.Völpel, and H. Janssen. *Scale-invariant Image Recognition Based on Higher Order Autocorrelation Features.* Pattern Recognition, 29(1), 1996.

[16] T.Kurita, N.Otsu and T.Sato. *A Face Recognition Method Using Higher Order Local Autocorrelation and Multivariate Analysis.* In 11th International Conference on Pattern Recognition, volume 2, pages 213- 216, The Hague, 1992. IEEE

[17] T.Kurita and N. Otsu. *A New Scheme for Practical Flexible and Intelligent Vision Systems.* IAPR Workshop on CV, pages 431- 435, October 1988.

[18] J.A.McLaughlin and J. Raviv. *Nth-Order Autocorrelations in Pattern Recognition.* Information and Control, 12:121 142, 1968.

[19] K.V. Mardia, J.T.Kent, and J.M.Bibby. *Multivariate Analysis.* Academic Press, 7th Printing, 1989.

[20] H. Neven, A. Steinhage, C. Bruckhoff. *Dynamical Systems for vision-based autonomous mobile robots.* In P.Maes, M.Mataric, J.A.Meyer, J. Pollack, S.W.Wilson, editors, From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Cambridge, MA. The MIT Press / Bradford Books, 1996.

[21] Y.H. Pao *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley Publishing Company, New York, 1989.

[22] G. Schöner, M. Dose, C.Engels. *Dynamics of behavior: theory and applications for autonomous robot architectures.* Robotics and autonomous systems, 16:213-245, 1995.

[23] Axel Steinhage, Thomas Bergener. *Dynamical Systems for the Behavioral Organization of an Anthropomorphic Mobile Robot.* In Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE '97, pages SS7-SS12. IEEE publications, 1997.

[24] B. Völpel. *Repräsentation und Erkennung dreidimensionaler Umgebungen mit einem aktiven Stereokamerasystem.* Phd Thesis, VDI Verlag, 1996, Düsseldorf BRD.