UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

A METHOD TO DETERMINE A HUMANOID ROBOT POSITION BASED ON MACHINE LEARNING STRATEGIES

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
JOSÉ IGNACIO LÓPEZ PEÑA

TUTOR
DR. ANGEL FERNANDO KURI MORALES
POSGRADO EN CIENCA E INGENIERÍA DE LA COMPUTACIÓN

MÉXICO, D. F. ENERO 2013

Index

Abstract

The present work introduces an algorithm to determine a humanoid robot’s position in an environment of known characteristics. The algorithm is based on evolutionary method as a machine learning strategy and takes into account the characteristics and constraints to which a humanoid robot is subject. The method raises the possibility to store in the robot a set of databases holding the minimal information from the images that describe the navigation area or possible obstacles that it may face. The information in the database can be used later at run time by the robot to navigate.

1 Introduction

An open problem in robotics is the one dealing with the way mobile robots locate themselves inside a specific area. The problem itself is vital for a robot to achieve its goals. There are several ways to approach this problem. For example using landmarks as in [1] and [2] or estimating the robot's position based on the distance it has covered as in [3] and [4]. Most of these methods make use of several sensors of different types in the robot to gather information about its surrounding environment [5]. However, it is not always possible to use landmarks in the environment, or use any kind and number of sensors in the robot because of a restriction of its design. Moreover, there is a high probability that the data obtained from the measurements of the sensors or actuators contain noise that will lead to a wrong assumption of the robot’s physical location. All these situations may apply to all kinds of mobile robots, in particular to humanoid robots. A humanoid robot is a robot whose design mimics the human body. The intention of this work is to provide a method to determine the position of a humanoid robot.

1.1 Definition of the problem

In order to determine the position of a humanoid robot, it is necessary to explain first its characteristics and the constraints of which it is subject. First, unlike most common mobile robots, the humanoid robot to which we refer in this work, does not rely on wheels to move from one position to another, it is a biped robot. Second, the only available sensor in the robot to gather information about the surrounding environment is a camera located in its head. Third, it is assumed that the robot has limited resources of memory space and processing power[1] thus it is crucial to use its resources in an optimal way.

That said the problem that the present work aims to solve is to give a robot with such characteristics the ability to localize itself inside an area using an evolutionary method as a machine learning strategy. In particular, one of the main purposes of the present work is to be able to implement a reliable algorithm to determine the robot's position in an environment of whose characteristics are assumed to be known at the offset. In spite of its apparent artificiality, it is of high practical interest because it addresses a problem which is systematically found when the robot finds itself in a competitive arena or in an otherwise known surrounding. This is the case, for example, of cleaning robots, where the space to free from obstacles is well known; or of a surgical procedure where nanobots may be needed to identify their positions inside a well known environment, etc. In particular, such is the case in a traditional international tournament popularly known as "RoboCup" (for which see section 1.3). From the point of view of artificial intelligence and mechatronics, RoboCup is specifically aimed at the (apparently) ambitious goal of designing a team of robots able to defeat its human counterpart by the year 2050.

1.1.1 Mobile robot Position's Identification

The method proposed in this work makes use of the single sensor the robot has, a camera, to capture a collection of images that will be processed off line and then loaded into the robot's memory as a condensed database that will include information regarding the position where each image was taken along with the processed images. The robot will use this information to determine its own position by searching in the database the image it is getting later. Therefore, it is important to determine a way to identify an image as unique with the less amount of information, i.e. the optimal number of pixels and their position in the images. The collection of pixel points and their position is called a "sieve".

[1] The humanoids robots we used had a CMUcam3 [43] vision module. It is an ARM7TDMI based fully programmable embedded computer vision sensor that only has 64 KB of RAM memory and 128 KB in ROM for custom C code. The processing power of this sensor is limited compared to other options as a fit-PC2 [44], which is a small, light, fan-less computer that can run a full OS like Linux.

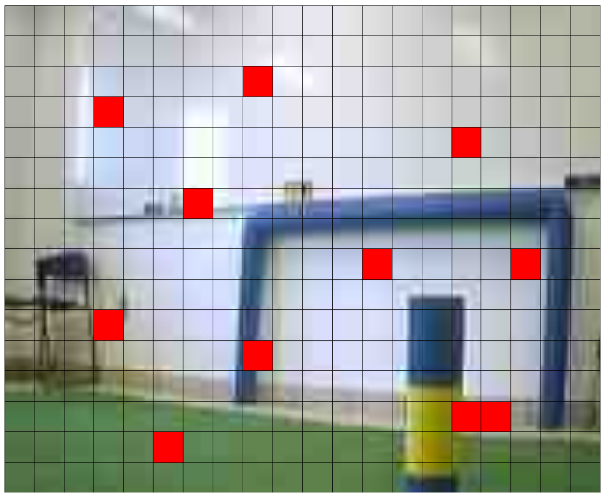


Fig. F1.1.2.1. A possible representation of a sieve.

1.1.2 Determination of the optimal sieve

To implement such database in the robot's memory it is necessary to reduce the number of elements in the database. This condition leads to the following questions. What is the smallest number of pixels which allows determine uniquely the position defining frame? And which are the positions of each and every one of these pixels?

The answer for these two questions cannot be simply obtained. On the one hand, it is obvious that a largest subset will ensure a more accurate identification of an image. On the other hand, with a small subset, the identification of the image will be as fast as possible. Evidently, these two objectives impose conflicting optimization goals that constitute a multi-objective optimization problem.

1.1.3 Multi-Objective optimization

We live in a world where most of the optimization problems have to deal with multiple objectives. This kind of problems involves two or more different functions, which typically are contradictory, and a tradeoff between the various objectives is always present. Therefore, there exists more than one optimal solution for these kind problems. In fact, one of the basic problems regarding this matter is how to adequately define an "optimum". Chapter 2 presents a quick overview of muti-objective optimization algorithms.

In what follows, there is a brief account of the various alternative approaches that have been explored in the recent past when trying to solve the problem of robot localization.

1.2 Current methods of robot localization

Self-localization is the task of estimating the pose (position and orientation) of a mobile robot. It is one of the fundamental problems in mobile robot navigation and many solutions have been introduced in the past. These solutions usually fall into one of the following categories [24]:

1. Behavior-based approaches that rely on the interaction of a robot with its environment.
2. Landmarks methods that rely on the recognition of landmarks to keep the robot localized geometrically.
3. Dense sensor matching methods that attempt to use the available sensors information to update the robot's pose.

In probabilistic terms, localization is the process of determining the probability of the robot being at a given pose [25] in a specific time. There are two classes of probabilistic localization. The first class uses an explicitly specified probability distribution across all possible positions; this type of localization is called the Markov localization. The second class uses a Gaussian probability density representation of robot position and scan matching for localization. This class of localization is called Kalman filter localization [5].

The ability to simultaneously localize a robot and at the same time accurately map its surroundings is known as SLAM (Simultaneous Localization And Mapping). This is considered a key prerequisite for an autonomous robot. SLAM addresses the problem of building a map of an environment from a sequence of landmark measurements obtained from a moving robot.

In a humanoid robot, the visual perception system plays a main role; because almost all other tasks the robot performs depend on it. The methods used to localize a humanoid robot based on computational vision gather information about the surrounding environment and detect known objects to estimate its position based on triangulation between these landmarks [13].

1.2.1 Current methods of object recognitions

A human being is capable to recognize a multitude of familiar and novel objects with little or no effort at all, regardless of the variations they may have. People can recognize objects from many different points of view, in different places, and of different sizes. Furthermore, they even recognize them if they are partially obstructed. While it may be obvious that people are capable of recognizing objects under many variations in conditions, this is still a challenge for computer vision systems.
Several methods have been proposed to recognize objects; some of them are based on feature descriptions of an image and the objects in it. These methods perform a search to find feasible matches between object features and image features.

1.2.1.1 Object recognition by indexing

Object recognition in computer vision refers to the problem of matching object features with those of an image. This matching process may be very complex due to the fact that an object has many features and an image may have features that do not necessarily belong to the object. One convenient way to represent the features of an image and their relations is through the use of graphs. On the other hand the indexing problem is the problem of recognizing few objects in a large database of objects while avoiding the use of the image-feature-to-object-feature matching paradigm. Horaud and Sossa proposed a method [35] where 3D objects and scene images are represented in terms of 2D characteristic views, which are then characterized with a number of weighted graphs. The graphs obtained from these objects will become the “model graphs”. These “model graphs” are characterized as polynomials to be able to organize them into hash tables to shape a database. Then the recognition of an object consists of determining whether a polynomial characterization of the graphs extracted from an image is present or absent in a database of model graphs. This kind of characterization constitutes a model for indexing because it rapidly states whether some sensed data equals some object’s data. Furthermore knowing that two polynomials are said to be equal if they have the same degree and if their coefficients are equal makes the problem of comparing two graphs equivalent to the problem of comparing only the coefficients of their associated polynomials. Recognition by indexing is therefore the process that attempts to rapidly extract from a large list of objects, those few objects that fit a group of image features while avoiding to establish image-feature-to-object-feature assignments.

1.2.1.2 Feature selection using a hybrid associative classifier with masking techniques

Aldape-Perez et al. introduced the method in 2006 [36]. The goal of the method is to improve the performance of current pattern classifiers by identifying an optimal subset of features and removing the redundant or irrelevant ones, without the need to compute a new classifier at each step. Its premise is that most of the times the initial set of selected features consists of a large number of potential attributes that constitute an obstacle not only to the accuracy but to the efficiency of algorithms. In high dimensional spaces these features often tend to be correlated. Many methods have been proposed to obtain the optimal subset of features, standing out the multilayer perceptron networks using the back propagation algorithm.

In this method the classification and extraction of features are performed in two phases. In the first phase the classification is done by a Hybrid Classification and Masking (HCM) technique. The classification occurs using a hybrid associative memory during the learning phase. Since one of the properties of the associative memories is to establish links between patterns and theirs classes, iterative complexity is eliminated. This means that one and only one classifier is computed during the whole classification process. In the second phase a mask search algorithm is applied to the patterns previously found in the first phase to recall them.

1.2.1.3 SIFT and SURF algorithms

The Scale-invariant feature transform or SIFT [28], finds the key points of the objects extracting them from a set of reference images. Objects are recognized in a new image by comparing each feature from the new image to key points previously found. The candidate matching features are selected based on the Euclidean distance.

The Speeded Up Robust Features or SURF [29], is a scale and rotation-invariant interest point detector and descriptor method partially inspired by the SIFT descriptor. A point of interest is one where combinations of measurements of intensity and its derivatives take on unusual values (for example, at corners) [30]. It determines the interest points using a Hessian matrix-based measure due to its stability and regularity. A descriptor describes the distribution of the intensity content within the interest point neighborhood, similar to the gradient information extracted by SIFT.

1.2.1.4 Graph transformation algorithm

This is a two-phase algorithm. It combines the local invariant features in the first phase with a structural matching method used in the verification phase. The features of both the image of the object to be recognized and the image where the object is present are found using any known method (e.g. SIFT). Once the features are found, the descriptors are compared to obtain an initial match if it exists. The comparison can be made using any correlating algorithm. For example, the one defined by equation (E1.2.1.4.1), where $c(x, y)$ is the correlation[2], $s(\xi,\eta)$ is the scene to be analyzed and $r^*(\xi - x,\eta - y)$ denotes the conjugated complex of the object to be detected.

$$c(x, y) = \iint_{-\infty}^{\infty} s(\xi,\eta) r^*(\xi - x,\eta - y) d\xi d\eta \qquad \text{(E1.2.1.4.1)}$$

Other algorithms that may be used for the correlation are described in [12]. Any match found in the first phase must be verified in the second. The verification is done using a graph transformation also described in [12]. The graphs produced by the features of each image are compared and the number of edges shared between the graphs determines if the object found in the first phase is valid or not.

1.2.2 Optimum Sieve Method

The Optimum Sieve method [26], [27] consists in finding the smallest set of pixels and their position (the “sieve”) that best identifies an image as unique in a set of images that represents the area a robot will navigate. In other words, this set of pixels must be a minimally sufficient representation of all the pixels in an image for all the images in the database. The purpose is to reduce the amount of the data of the image’s database so it can be stored it a robot’s memory and consequently increase the performance of image identification in a later stage. This method is one of the main motivations for this work and it will be discussed in detail throughout the entire document.

1.2.2.1 Image Identification

Once the *sieve* is determined, the image identification is performed just looking at the position of the pixels in the test image. Then their value is compared with the images in the database stored in the robot’s memory.

[2] **Correlation** refers to any of a broad class of statistical relationships involving dependence.

1.2.2.2 Image formats

All the images in the present work were initially captured as a standard JPEG image using the Fire-i™ Digital Camera [15] at a resolution of 640 × 480 pixels. However, the images were not used either with the same resolution or with the same format. They were scaled down to a resolution of 176 × 144 pixels and saved to the PPM file format to facilitate their analysis and handling. The image manipulation was done using the GIMP[3] [34] software.

1.3 RoboCup

The Robot World Cup Initiative or "RoboCup" for short is an international scientific initiative with the goal of advancing the state of the art of intelligent robots. The initiative and project dates since 1997, and originally used the game of soccer to promote AI intelligence and technology. As of today, RoboCup has expanded into other relevant application domains based on the needs of modern society covering the following categories: RoboCupSoocer, RoboCupRescue, Robocup@Home and RoboCupJunior. The ultimate goal of the RoboCup Initiative as mentioned above and as stated in the official website [6] is: "By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game; comply with the official rule of the FIFA, against the winner of the most recent World Cup."

1.4 Goals

The main goal of the present work is to make a humanoid robot able to determine its position using only a standard camera and an optimally reduced database. In order to achieve this goal an evolutionary method to find the optimal number of elements and their position in a sieve is used. It is, therefore, a machine learning strategy to setup an optimal database containing the minimum information needed to localize the robot in an area and recognize obstacles and enemies.

2 Multi-Objective Optimization Algorithms

Most, if not all, day-to-day problems in our world have to deal with multiple objectives. A typical way to deal with this kind of problems is to combine all the objectives into one single function and give each objective in it a specific weight [8]. This ap-

[3] GIMP is a multi-platform photo manipulation tool. GIMP is an acronym for the GNU Image Manipulation Program.

proach is inconvenient because it introduces the problem of how to weight each objective in the function. A different and sturdier approach is to model the problem as separate objective functions for each requirement (including restrictions that apply for all of them). Unlike single-objective optimization problems, there is no accepted definition of optimal in multi-objective optimization. One possible definition could be a way to find a good compromise or "trade-offs" rather than a single solution that simultaneously satisfies all, leading this to a complete set of valid solutions [10].

2.1 Pareto Front

In the simultaneous optimization of multiple objectives it is not always possible to achieve a single optimal solution with respect to all of them. If those objectives were independent there would be a whole set of solutions that would best suit all the objectives simultaneously. This set of solutions gives shape to what is called a Pareto front. Figure F2.1.1.1 shows an example of a Pareto front for two different objective functions f_1 and f_2. A and B are two different points of the Pareto Front. Doing a comparison between them in each objective function f_1 and f_2, we see that A is less than B in objective function f_1, but B is less than A in objective function f_2; therefore both solution vectors are non-dominated. Point C is also a feasible solution but it does not belong to the Pareto front as B dominates it in both objective functions f_1 and f_2.
Before giving the formal definition of the Pareto front, it is necessary to introduce first the concepts of Pareto dominance, Pareto optimality and Pareto optimal.

2.1.1 Pareto Dominance

A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (which dominance is denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.

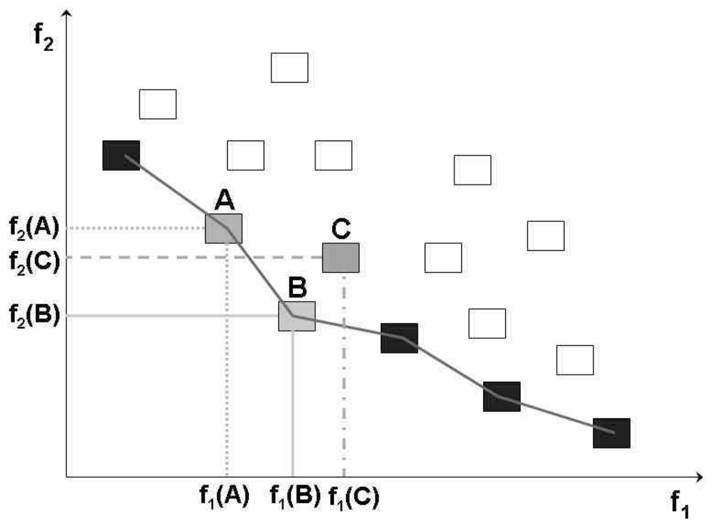


Fig. F2.1.1.1. Example of a Pareto front [41].

2.1.2 Pareto Optimality

In general, the solutions of a multi-objective optimization problem are related in two ways: either one of them dominates the other solutions or another dominates it (see Figure F2.1.1.1). A vector of decision variables $\vec{x}$ is a Pareto optimal if there is no other vector in the feasible region of the problem, which would decrease some criterion without causing a simultaneous increase in at least one other criterion. This concept does not lead to a single solution, but rather to a valid optimal set of solutions for the same problem, which is called the Pareto optimal set [8]. The vector $\vec{x}$, corresponds to the solutions included in the Pareto optimal set and are called non-dominated. The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the Pareto front.

2.1.3 Pareto Optimal Set

The Pareto optimal set $\left(\mathcal{P}^*\right)$ is defined as: $\mathcal{P}^* := \left\{x \in \Omega \mid \neg \exists x' \in F\left(x'\right) \preceq f\left(x\right)\right\}$

In other words, x is a candidate solution in the set of optimal solutions Ω for which it does not exists another candidate solution x' for all the objective functions that dominates at least one objective function with x.

2.1.4 Pareto Front

For a given Pareto optimal set $(\mathcal{P}^*)$, the Pareto front $(\mathcal{PF}^*)$ is then defined as the optimal solution from the set of all candidates solutions $\mathcal{PF}^*: \left\{ f_m(x) = \left[f_1(x), f_2(x), \ldots, f_M(x) \right] \mid x \in \mathcal{P} \right\}$

2.2 Types of Multi-Objective Algorithms

There are several ways to tackle a multi-objective optimization problem, In the beginning of this section one such method (called aggregating functions) was briefly mentioned. This and other methods will be briefly discussed in the rest of this section.

2.2.1 Aggregate objective function (AOF)

This is an intuitive approach to solve multi-objective problems. The basic idea is to combine all of the objectives into a single objective function, called the AOF (such as the well-known weighted linear sum of the objectives [19]). This objective function is optimized subject to constraints specifying how much of one objective must be penalized. These constraints frequently come in the form of inequalities $g_i(\vec{x}) \leq 0 \; i = 1,2,\ldots,m$ or equalities $h_j(\vec{x}) = 0 \; j = 1,2,\ldots,p$ for where $\vec{x} = \left[x_1, x_2, \ldots, x_n \right]$ is the vector of decision variables.

Often the aggregate objective function is not linear in the objectives expressing increasing marginal dissatisfaction with greater incremental sacrifices in the value of either objective. Furthermore, sometimes the aggregate objective function is additively separable, so that it is expressed as a weighted average of a non-linear function of one objective and a non-linear function of another objective [18]. The optimal solution will depend on the relative values of the weights specified. This method is subjective as it requires to be supplied with the weights to find a solution.

2.2.2 Multiple (AOF) methods

Unlike AOF a fair way to characterize multi-objective problems is by identifying multiple Pareto optimal candidate solutions. These candidate solutions can be found with the use of multiple AOFs where the solution of each AOF yields a Pareto point. These methods include the Normal Boundary Intersection (NBI) [19], Normal

Constraint (NC) [20], Successive Pareto Optimization (SPO) [21], and Directed Search Domain (DSD) [22].

2.2.3 Approximation by an even distribution of the Pareto points

The NC and DSD methods suggest two different filtering procedures to remove locally Pareto points. The AOFs are constructed with the aim of obtaining evenly distributed Pareto points that give a good impression (approximation) of the real set of Pareto points.

2.2.4 Other Methods

There are other methods to solve multi-objective optimization problems such as: particle swarm optimization, artificial immune systems, cultural algorithms, differential evolution, ant colony optimization, taboo search, among others that are described in [17].

2.2.5 Multi-Objective Evolutionary Algorithms

Unlike traditional optimization techniques that search for a solution using a single point, evolutionary algorithms (EAs) work with a simultaneous set of possible solutions called the population. Genetic Algorithms are a type of EAs which:

a) Explore a discrete problem space.
b) Test multiple points of the problem space simultaneously.
c) Encode possible solutions (points) of the problem space.
d) Evaluate every point independently.
e) Combine codes of the evaluated partial solutions to generate new exploration points.
f) Randomly alter selected elements of the codes of the points

These characteristics makes EAs suitable to solve multi-objective optimization problems as they can find several members of the Pareto optimal set in a single "run" of the algorithm [17]. C. Coello gives a definition of a Multi-Objective Evolutionary Algorithm in [17] which is included here:

Definition 1 (MOP Global Minimum): Given a function $\mathrm{f}: \Omega \subseteq \mathbb{R}^n \longrightarrow \mathbb{R}^k, \Omega \neq \emptyset$, $k \geq 2$ for $\mathbf{x} \in \Omega$ the set $\mathcal{PF} \triangleq \mathrm{f}(\mathbf{x}_i^*) > (-\infty, \dots, -\infty)$is called the global minimum if and only if $\forall\, \mathbf{x} \in \Omega$:$\mathrm{f}(\mathbf{x}_i^*) \preccurlyeq \mathrm{f}(\mathbf{x})$. Then, $\mathbf{x}_i^*, i = 1, \dots, n$ is the global minimum solution set (i.e., $\mathcal{P}^*$), f is the multiple objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution set is called the MOP global optimization problem.

Definition 2 (Evolutionary Algorithm): Let I be a non-empty set (the individual space), $\{\mu^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the parent population sizes), $\{\mu'^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the offspring population sizes), $\Phi : I \to \mathbb{R}$ a fitness function, $\iota : \bigcup_{i=1}^{\infty}\left(I^{\mu}\right)^{(i)} \to \{\text{true}, \text{false}\}$ (the termination criterion), $\chi \in \{\text{true}, \text{false}\}$, r a sequence $\left\{r^{(i)}\right\}$ of recombination operators $r^{(i)}: \ \mathbb{X}_r^{(i)} \to \mathcal{T}\left(\Omega_r^{(i)}, \mathcal{T}\left(I^{\mu^{(i)}}, I^{\mu'^{(i)}}\right)\right)$, m a sequence $\left\{m^{(i)}\right\}$ of mutation operators $m^{(i)}: \ \mathbb{X}_m^{(i)} \to \mathcal{T}\left(\Omega_m^{(i)}, \mathcal{T}\left(I^{\mu'^{(i)}}, I^{\mu'^{(i)}}\right)\right)$, s a sequence $\left\{s^{(i)}\right\}$ of selection operators $s^{(i)}: \ \mathbb{X}_s^{(i)} \times \mathcal{T}\left(I, \mathbb{R}\right) \to \ \mathcal{T}\left(\Omega_s^{(i)}, \mathcal{T}\left(\left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right), I^{\mu^{(i+1)}}\right)\right)$, $\Theta_r^{(i)} \in \mathbb{X}_r^{(i)}$ (the recombination parameters), $\Theta_m^{(i)} \in \mathbb{X}_m^{(i)}$ (the mutation parameters), and $\theta_s^{(i)} \in \mathbb{X}_s^{(i)}$ (the selection parameters). Then the algorithm shown in Figure F2.2.5.1 is called an Evolutionary Algorithm.

```
t := 0;
initialize $P(0) := \{a_1(0), \ldots, a_\mu(0)\} \in I^{\mu^{(0)}}$;
while ($\iota(\{P(0), \ldots, P(t)\}) \neq$ true) do
    recombine: $P'(t) := r^{(t)}_{\Theta_r^{(t)}}\left(P(t)\right)$;
    mutate: $P''(t) := m^{(t)}_{\Theta_m^{(t)}}\left(P'(t)\right)$;
    select:
      if $\chi$
        then $P(t+1) := s^{(t)}_{\left(\Theta_s^{(t)}, \Phi\right)}\left(P''(t)\right)$;
        else $P(t+1) := s^{(t)}_{\left(\Theta_s^{(t)}, \Phi\right)}\left(P''(t) \cup P(t)\right)$;
      fi
    t := t + 1;
od
```

Fig. F2.2.5.1. Evolutionary Algorithm Outline [17]

Definition 3 (Multi-Objective Evolutionary Algorithm): Let $\Phi : I \rightarrow \mathbb{R}^k$, (k ≥ 2, a multi-objective fitness function). If this multi-objective fitness function is substituted for the fitness function in Definition 1 then the algorithm shown in Figure F2.2.5.1 is called a Multi-objective Evolutionary Algorithm.

2.2.5.1 Types of MOEAs

Multi-Objective Evolutionary Algorithms can be classified into three major categories [8], [17] depending on what stage, search or decision, is carried out first.

1. A Priori techniques: These techniques make decisions before searching. They are based on non-Pareto approaches; they include lexicographic, linear fitness combination, and nonlinear fitness combination. The solution obtained with these techniques is a single point and therefore they do not incorporate the concept of Pareto optimal. However, they are easy to implement.
2. Progressive techniques or interactive computational steering: These techniques integrate search and decision-making. They use a ranking scheme to rank each individual in the population and a sharing function to preserve diversity.
3. A posteriori techniques: These techniques perform a search before making any decision. They are emphasized in performing a search as widespread as possible, to generate as many different elements of the Pareto optimal set as possible. The decision-making process takes place only after completing the search. Among these techniques are: independent sampling, criterion selection, aggregation selection, Pareto-based selection, Pareto rank- and niche-based selection, Pareto deme-based selection, Pareto elitist-based selection, etc.

In the following sections some of the first multi-objective evolutionary algorithms are mentioned.

2.2.5.2 VEGA

David Schaffer [47] proposed the Vector Evaluated Genetic Algorithm (VEGA) in 1985. It is considered the first implementation of a MOEA [17]. The idea of the algorithm is to evaluate a vector of objective functions of a multiobjective problem. The following description of the algorihtm is given in [17]: "The VEGA concept is that, for a problem with *k* objectives, *k* sub-populations of size *M/k* each would be generated (assuming a total population size of *M*). Each sub-population uses only one of the k objective functions for fitness assignment. The proportionate selection operator is used to generate the mating pool. These sub-populations are then shuffled together to obtain a new population of size M, on which the GA would apply the crossover and mutation operators in the usual way." These techniques do not incorporate directly the concept of Pareto optimum, are unable to find some portions of the Pareto front, and are only capable of handling a small number of objectives [8].

```
1. Procedure MOGA($N', g, f_k(\mathrm{x})$) $\triangleright$ $N'$ members evolved $g$ generations to
   solve $f_k(\mathrm{x})$
2. Initialize Population $\mathbb{P}'$
3. Evaluate Objective Values
4. Assign Rank based on Pareto Dominance
5. Compute Niche Count
6. Assign Linearly Scaled Fitness
7. Shared Fitness
8. for $i := 1$ to $g$ do
9.    Selection via Stochastic Universal Sampling
10.   Single Point Crossover
11.   Mutation
12.   Evaluate Objective Values
13.   Assign Rank Based on Pareto Dominance
14.   Compute Niche Count
15.   Assign Linearly Scaled Fitness
16.   Assign Shared Fitness
17. End for
18. End procedure
```

Fig. F2.2.5.3.1. MOGA algorithm [17]

2.2.5.3 MOGA

The Multi-Objective Genetic Algorithm (MOGA) was proposed by Carlos M. Fonseca and Peter J. Fleming in 1993 [48]. It is a variation of Goldberg's GA where each individual is ranked according to number of individuals in the current population by which it is dominated [8]. All individuals that are nondominated have the rank 1. The dominated individuals are penalized according the number of indivuals by which they are dominated [17], [45].

In the algorithm a fitness sharing function is used to maintain diversity, together with a mating restriction scheme that avoids crossover between very distant individuals in the search space. Figure F2.2.5.3.1 shows the MOGA algorithm as presented in [17].

```
1.  Procedure SPEA$(N', g, f_k(\mathrm{x}))$
2.  Initialize Population $\mathbb{P}'$
3.  Create empty external set $\mathbb{E}'(|\mathbb{E}'| < |\mathbb{P}'|)$
4.  for $i := 1$ to $g$ do
5.      $\mathbb{E}' = \mathbb{E}' \cup \mathcal{ND}(\mathbb{P}')$ ⊳ Copy members evaluating to be nondominat-
        ed of P to E
6.      $\mathbb{E}' = \mathcal{ND}(\mathbb{E})$ ⊳ Keep only member evaluating to nondominated
        vectors in E
7.      Prune $\mathbb{E}'$ using clustering if max capacity of $\mathbb{E}'$ is ex-
        ceeded
8.      $\forall_{i \in \mathbb{P}'}$ Evaluate($\mathbb{P}'_i$) ⊳ Evaluate fitness for all member of $\mathbb{E}'$
        and $\mathbb{P}'$
9.      $\forall_{i \in \mathbb{E}'}$ Evaluate($\mathbb{E}'_i$)
10.     $\mathcal{MP} \leftarrow \mathcal{T}(\mathbb{P}' \cup \mathbb{E}')$ ⊳ Use binary tournament selection with re-
        placement to select individuals from $\mathbb{P}' + \mathbb{E}'$ (multiset union)
        until the mating pool is full
11.     Apply crossover and mutation on $\mathcal{MP}$
12. End for
13. End procedure
```

Fig. F2.2.5.4.1. SPEA algorithm [17]

2.2.5.4 SPEA

The Strength Pareto Evolutionary Algorithm (SPEA) was introduced by Eckart Zitzler and Lothar Thiele [49]. This approach was conceived as a way of integrating different MOEAs. Its main characteristics [46] are:

- It incorporates elitism, a set of the optimal individuals generated are maintained besides the population.
- The set is used to evaluate the fitness of individuals according to the prevalence relationship.
- The population's diversity is preserved using a prevalence based mechanism.
- A clustering method is incorporated in order to reduce the Pareto set without losing its characteristics.

SPEA uses an external archive containing nondominated solutions previously found. At each generation, the nondominated individuals are copied to the external nondominated set. For each individual in the external set, a strength value is

computed. The strength is proportional to the number of solutions to which a certain individual dominates. The fitness of each member of the current population is computed according to the strengths of all external nondominated solutions that dominate it. The fitness assignment process of SPEA considers both closeness to the true Pareto front and even distribution of solutions at the same time.
Figure F2.2.5.4.1 shows the pseudo code of SPEA as presetned in [17].

```
1. Procedure NSGA-II $(\mathcal{N}, g, f_k(x_k))$ $\triangleright \mathcal{N}'$ members evolved $g$ generations to
   solve $f_k(x)$
2. Initialize Population $\mathbb{P}'$
3. Generate random population - size $\mathcal{N}'$
4. Evaluate Objective Values
5. Assign Rank (level) Based on Pareto dominance - sort
6. Generate Child Population
7. Binary Tournament Selection
8. Recombination and Mutation
9. for $i := 1$ to $g$ do
10.     for each Parent and Child in Population do
11.     Assign Rank (level) based on Pareto - sort
12.     Generate sets of nondominated vectors along PFknown
13.        Loop (inside) by adding solutions to next generation
           starting from the first front until $\mathcal{N}'$ individuals
           found determine crowding distance between points on
           each front
14.     end for
15.    Select points (elitist) on the lower front (with lower
       rank) and are outside a crowding distance
16.     Create next generation
17.     Binary Tournament Selection
18.     Recombination and Mutation
19.     end for
20. end procedure
```

Fig. F2.2.5.5.1. The NSGA-II algorithm [17]

2.2.5.5 NSGA-II

Deb, et al [23] proposed an algorithm called NSGA-II wich is an improved version of the NSGA. It performs sequentially the three basic tasks of fitness assignment, density estimation and archiving [8]. It uses simultaneously an elite preservation strategy and an explicit diversity preserving mechanism.

1. A child population is created using the parent population, both population are of size *N*.
2. Both populations are combined together to form a single population of size 2*N*.
3. The combined population is classified using non-dominated sorting.
4. Finally, the new population is filled with the individuals of the best fronts, until its size becomes equal to *N*. If the population becomes larger than *N*, a niching strategy is used to select the individuals of the last front.

   Niching refers to the clustering of individuals with similar characteristics inside the population such that the identification of the said clusters will promote and maintain a stable population [32], [33]. A way to do this is by adding a clustering distance to each member of the population. Labeling the individuals as per their cluster keeps the population diverse and helps the algorithm to explore the fitness landscape [17].

The pseudo code of the algorithm as presented in [17] is shown in figure F2.2.5.5.1:

2.2.5.6 RSPGAe – Reduced Pareto Set GA With Elitism

The RPSGAe [10] is a Genetic Algorithm that seeks to distribute uniformly the solutions across the Pareto's front. Besides, it reduces the set of solutions by clustering them in groups that maintain their characteristics. The algorithm is explained in detail in [8], [10] and its main steps are listed in Figure F2.2.5.6.1. First an internal population of size *N* is randomly created (step 1) then an empty external population is created (step 2). In each generation the following operators are applied: i) The population is evaluated (step 3a); ii) a clustering technique is applied to reduce the number of solutions in the Pareto front and the fitness of the internal population is obtained (step 3b); iii) a predetermined fixed number of the best individuals is copied to the external population (step 3c); iv) if the external population is not full the genetic operators selection (step 3c), crossover (step 3f) and mutation (3g) are applied to the internal population and the cycle is repeated. Otherwise, a clustering technique[4] is applied to the external population (step 3d-i), the individuals are then sorted and a predefined fixed number of the best ones are copied to the internal population replacing the individuals with the worst fitness.

[4] In MOEA clustering techniques are used to reduce large sets of solution candidates with a minimum loss of diversity leading to a better distribution of the solutions in the Pareto Front [46]. Clustering is one of the metrics used to compare different algorithms on how well distributed are the solutions in the PF [50]. One of these techniques used for clustering is the grid-mapping technique.

```
1)  Generate a random initial population (internal).
2)  Create an empty external population.
3)  While not Stop-Condition Do
    a) Evaluate internal population.
    b) Calculate the Fitness of the individuals using a cluster-
       ing technique.
    c) Copy the best individuals to the external population
    d) If the external population becomes full then
       i)  Apply the clustering to this population.
       ii) Copy the best individuals to the internal population.
       End If
    e) Select the individuals for reproduction.
    f) Apply crossover.
    g) Apply mutation.
4)  End While
```

Fig. F2.2.5.6.1. RPSGAe Algorithm [8]

3 World mapping

So far, we talked about how multiple optimization problems are solved. In addition, a brief introduction to the methods used for object recognition in computer vision was given as well as a glimpse to the optimum sieve method. However nothing has been said about the environment the robot is going to navigate or how the information (in our case the images of the environment), were acquired and optimized and how it is going to be used to localize the robot. These topics are going to be addressed in this section, starting with the definition of mapping and world map and how the environment is usually represented. Then the details of the environment used for the purpose of this work are given.

Knowing the environment is very helpful to find routes, i.e. paths the robot may use to navigate to different positions in the environment. Furthermore, if an optimal route is required then it is necessary to have a map, which will be defined later in this section. For a mobile robot, this is an important part of the navigation system. The problem of world mapping in robotics is that given some readings from the robot's sensors, it should find a map of the environment to localize itself as it moves in it. This can be seen as the chicken-or-egg problem: if there is a map, localization can be done; if there is localization, a map can be created.

Representation Classification	Description	Uses	Notes
Continuous representation	Used for an exact decomposition of the environment. This means that the features of the environment can be represented precisely in continuous space.	Mobile robots use continuous maps only in 2D representations.	An advantage is the high accuracy with respect the environment's configuration and the robot's position. This approach is usually combined with a closed-world assumption.
Decomposition strategies	This is an abstraction of the environment. Decomposition and a selection of environmental features take place.	It is useful when it is planned carefully to abstract the relevant useful features of the world.	A disadvantage is the lost of accuracy. Its advantage is the minimization of the map representation.

Table T3.1.1. Environment classification

3.1 Definition of world map

A world map in robotics is the representation of a physical environment in a way that a robot can understand. With this definition, a single environment could be represented in many different ways. Therefore the choice of the map's definition may be very broad. This leads to the following question: which representation is the best for the robot? To answer it, Roland Siegwart and Illah R. Nourbakhsh recommend in their book "Introduction to Autonomous Mobile Robots" that the following three relationships must be understood first [5]:

1. The precision of the map must appropriately match the precision with which the robot needs to achieve its goals.
2. The precision of the map and the type of features represented must match the precision and data types returned by the robot's sensors.

3. The complexity of the map representation has direct impact on the computational complexity of reasoning about mapping, localization, and navigation.

Having these requirements in mind the obvious conclusions is that each representation has an impact on the robot. Table T3.1.1 shows a classification of the environment representation along with current uses, advantages and disadvantages.

3.2 Mapping algorithms fundamentals

Once the way to represent the environment has been chosen, the new question to answer is: How does a robot obtain a map? Before answering the question, it is needless to say that creating a map is not an easy task. The readings from the robot's sensors may contain a lot of noise. In addition, the robot's motions may not be precise. This can also be seen as noise in the robot's movements. Many techniques have been developed to overcome this problems and Sebastian Thrun [15] has pointed out that there are already robust methods for mapping static, structured and limited size environments. However, the problem we want to solve is not static at all. Once the robot is in the area it is going to navigate, the environment changes dynamically as there are other agents interacting analogously inside it. Some techniques approach this problem by using short time windows where the environments are considered as static. Thrun also points out that as of today most state-of-the-art robotic mapping algorithms are probabilistic [15], meaning that they use probabilistic models of the environment. The reason behind this lies on what we previously pointed out: robotic mapping is characterized by the noise generated inside and outside the robot's sensors and the uncertainty produced by them on the robot's position.

3.2.1 Bayes rule

The Bayes rule is used as the starting point for many statistical mapping algorithms. It relates the chances of the occurrence of event x given the evidence y. It states that:

$$posterior = \frac{likelihood \times prior}{marginal\ likelihood} \qquad \text{(E3.2.1.1)}$$

and is defined by equation (E3.2.1.2)

$$p(x \mid y) = \eta p(y \mid x) p(x) \qquad \text{(E3.2.1.2)}$$

where $p(x \mid y)$ is called the posterior probability distribution over *X*. $p(x \mid y)$ specifies the probability of finding *y* given *x*. In other words, it makes possible to generate different measures *y* under different states *x*. $p(x)$ is the probability to be at a given state *x* before any measure, and it is called the prior probability distribution. η is a normalizer used to ensure that it is valid probability distribution. In robotics mapping all data from the sensors is obtained over time. The Bayes filter (see section 3.2.3) is a powerful tool for performing sequential estimations and it is commonly used to integrate this data.

3.2.2 Sequential estimation

To better understand the Bayes filter we point out that a sequential estimation is a method used to estimate a parameter by analyzing large enough data samples until significant results are observed or a certain degree of precision is reached.
To further illustrate this concept suppose that the data obtained from one of the robot's sensor needs to be analyzed. If the probabilistic distribution function of the measurements on that sensor is already known ($p(x)$), the mean can be obtained directly with:

$$\mu = \int x p(x) dx \qquad \text{(E3.2.2.1)}$$

In general, $p(x)$ is not known. Therefore to obtain the mean it is necessary to perform estimations with the data sampled by the sensor. If *n* samples are obtained $\{x_1, x_2, \ldots, x_n\}$ the mean can be obtained by equation E3.2.2.2.

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{(E3.2.2.2)}$$

With this approach, all *n* samples must be known before calculating the mean. However, a better approach is to update the mean as new samples are obtained. For only 1 sample the mean will be $m_1 = x_1$, doing this will also improve the previous estimation of the mean as new samples are acquired. Therefore the mean *m* in equation (E3.2.2.2) will become a function of the sample x_k and the value of the mean before the current sample m_{k-1} as it shown in (E3.2.2.3):

$$
\begin{aligned}
m_k &= \frac{1}{k}\sum_{i=1}^{k} x_i \\
&= \frac{1}{k}x_k + \frac{1}{k}\sum_{i=1}^{k-1} x_i \\
&= \frac{1}{k}x_k + \frac{1}{k}\frac{k-1}{k-1}\sum_{i=1}^{k-1} x_i \qquad \text{(E3.2.2.3)} \\
&= \frac{1}{k}x_k + \frac{k-1}{k}\left(\frac{1}{k-1}\sum_{i=1}^{k-1} x_i\right) \\
&= \frac{1}{k}x_k + \frac{k-1}{k}m_{k-1}
\end{aligned}
$$

This is known as the sequential mean of a random variable [38] and is described by equation (E3.2.2.4).

$$
m_t = \frac{1}{t}x_t + \frac{t-1}{t}m_{t-1} \qquad \text{(E3.2.2.4)}
$$

where t is the time where the sample is obtained.
Posterior probability distribution can also be sequentially estimated as actions are taken and as observations are made [38]. The produced actions and the observed states over time can be modeled and the model can be used to improve the estimates. However we also have to bear in mind that noise will always be present in the observations as in the state, and it has to be included in the model.

3.2.3 Bayes Filter

The Bayes filter is used as a recursive estimator. It consists of two steps. The first one called Prediction update, and the second one called the Measurement update. The Prediction update represents how the current state is expected to be, given the estimate of the previous state and what actions were applied. It is given by equation (E3.2.3.1).

$$
pu(x_t) = \int p(x_t \mid x_{t-1}, a_t) \cdot mu(x_{t-1})dx_{t-1} \qquad \text{(E3.2.3.1)}
$$

where x_t is the current state, x_{t-1} is the previous state, a_t is the action taken at time t and the function $mu(x_{t-1})$ is called the Measurement update.

The Measurement update represents how the observations over time, z_t, change the posterior into the prior probability. It comes directly form Bayes rule (E3.2.1.1.1) and is expressed by equation (E3.2.3.2)

$$mu(x_t) = \eta p(z_t \mid x_t) pu(x_t) \qquad \text{(E3.2.3.2)}$$

Where η is a normalizer and $pu(x_t)$ is the prediction update.

By combining equations (E3.2.3.1) and (E3.2.3.2) in a single equation (E3.2.3.3), a generic Bayes filter for simple estimation problems is obtained. With equation (E3.2.3.3) a posterior probability over the state x_t can be obtain recursively.

$$p(x_t \mid z^t) = \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid z^{t-1}) dx_{t-1} \text{(E3.2.3.3)}$$

where z^t represents the data obtained from one of the robot's sensors up to time *t*, i.e $z^t = \{z_1, z_2, \ldots, z_t\}$, x_t is current state and x_{t-1} is previous state.

As the Bayes filter is recursive, it needs to be initialized with an initial probability at time $t = 0$. The initial probability is $p(x_0 \mid z^0) = p(x_0)$. In [39] Thrun gives an over-view of how $p(x_0)$ should be initialized when x_0 is unknown. The Bayes filter also requires that the state x_t contains all unknown quantities that may influence sensor measurements at multiple points in time. In the context of robotic mapping, there are typically two such quantities: the map and the robot's pose in the environment. Both of them influence sensor measurements over time. Hence, when using probabilistic techniques, the mapping problem is truly one where both the map and the robot pose have to be estimated together [15]. Using *m* to denote the map and *s* for the robot's pose the following Bayes filter is described by equation (E3.2.1.3.4). Notice that *s* and *m* comprise the full state vector *x*, i.e. $x_t = (s_t, m_t)^T$

$$p(s_t, m_t \mid z^t) = \eta p(z_t \mid s_t, m_t) \iint p(s_t, m_t \mid s_{t-1}, m_{t-1}) p(s_{t-1}, m_{t-1} \mid z^{t-1}) ds_{t-1} dm_{t-1} \qquad \text{(E3.2.3.4)}$$

A detailed discussion about the Bayes rule and the Bayes filter can be found in [39].

3.2.4 Kalman Filter

Rudolph Emil Kalman introduced the Kalman filter in the1950s. It was primary used as a technique for filtering and predicting in linear systems [39]. Kalman filters are

Bayes filters [42] that represent the posterior probability $p(s_t, m_t \mid z^t)$ with Gausians[5]. Here we will present a brief description of the Kalman Filter as explained in [15] and [39]. Mapping algorithms based on Kalman filters are often referred to as SLAM algorithms [15]. The mapping relies on the following assumptions: the motion model and the perceptual model must be linear with added Gaussian noise. A linear motion model implies that the pose of the robot and the map at a given time depend linearly on the previous pose and map. However, the Kalman filter is implemented for continuous states. Therefore it is not applicable to discrete or hybrid state spaces. Hence, a modified version was introduced in 1960 called the discrete Kalman Filter. The discrete Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the "a priori" estimates for the next time step. The measurement update equations are responsible for the feedback for incorporating new measurements into the a priori estimate to obtain an improved a posteriori estimate. As pointed out previously, the Kalman filter addresses the problem of trying to estimate the state of a discrete-time controlled process governed by a linear model. In cases where a non-linear model is present, the model needs to be approximated by a linear function before using the Kalman filter. Such implementation is referred to as an extended Kalman filter or EKF.

3.2.5 Markov localization

Markov localization is the straightforward application of Bayes filters to the localization problem. It addresses the global localization problem, the position tracking problem, and the kidnapped robot problem in static environments. Markov localization uses an explicitly specified probability distribution across all possible robot positions. The robot's belief[6] state is represented as a separate probability assignment for every possible robot pose in the robot's map. The data from the robot's sensor measure-

---

[5] A **Gaussian function** is a function of the form $f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$. The parameter a determines the height of the curve, b is the position of its center, and c determines the width. Gaussians functions are used to describe normal distributions.

[6] The robot's **belief** about its position on the environment.

ments is used to update each possible robot position in the belief state individually using the Bayes formula. Therefore, every time a new measure is sensed, the probability of every possible pose of the robot in the map has to be updated as well. A more detailed discussion of these topics is found in [5], [15], [39], [40].

3.3 Environment set up

So far the concept of map in robotics and some of the algorithms that have been used for mapping have been introduced. In this section we will describe how the environment was setup for this work.

The environment consisted of an area of 2.1 × 2.1 mts^2 divided into 49 square tiles each one of 0.30 × 0.30 mts2, each tile mapped to a Cartesian coordinate in a (x, y) plane. The following three tasks were performed to set the environment:

1) The landscape was segmented in three coordinates:
    i. A horizontal coordinate
    ii. A vertical coordinate
    iii. An observation angle (x, y, θ)
2) A picture was taken on each possible (x, y, θ) coordinate combination i.e. images were taken from 360º around the robot at each coordinate with a 30º step for the observation angle. At each coordinate (x, y) 12 images were captured (as shown in Figure F3.3.1) yielding a total of 588 images.
3) Each one of the images was digitalized using a standardized RGB format with a total of 640 × 480 pixels per image.
4) Each one of the images was manipulated in GIMP to resize them using the Sinc (Lanczos 3) interpolation [37] to a resolution of 176 × 144 pixels. All images were saved in the PPM format (Portable Pixel Map [9]), which maintains constant file size for all images. To reduce the amount of rough data the images were then sampled at regular interval every 16 pixels by our algorithm.

These images formed a database that included information about the position of the image. This information consists of the x, y coordinates and the angle θ of the image. All of these images will be used as offline training data so the robot can know its position by looking only into the database for an image that match with what its sensor is getting at some future time. It is true that there may be cases where some ambiguity will occur. For example, when the robot is to close to a wall and it is facing to it or it is to close to a post or simply by looking directly to the floor. In those circumstances, a possible solution is to make the robot turn its head and camera to a different angle to take a new picture.

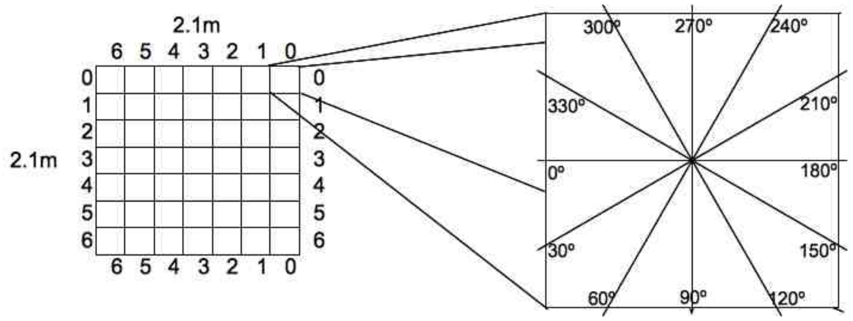


Fig. F3.3.1. Illustration of the sampled area.

3.3.1 The Portable Pixel Map Format

The PPM format is a color image file format that can contain one or multiple images in a single ppm file that can be ASCII or binary. In the file there are no data delimiters or padding before, after, or between images. The file consists of two parts. In the first part lies the general description of the file. Among other things, the first part of the file contains the type of the file, the height and width of the images and the maximum color value. The second part of the file is a raster that represents one or multiple images, every width pixels in the raster represents a row of the image and the raster will have height number of rows in it to represent a single image.

Each pixel in the row is the value of the red, green, and blue samples of the pixel in this order. Each sample can be either 1 or 2 bytes of size depending on the maximum value reported in the first part of the file.

For this work, a binary representation of the file was used, leading to 54 bytes (from byte 0x0000 to 0x0035) that represent the description of the file. The second part, starts from byte 55 (0x0036 onwards) and it is a raster that represents the images. As we used images of size 176 × 144 pixels, every 528 bytes in the raster represented a row of pixels of the image in the RGB color space. The reason behind using this format is that it is easier to process the pixel information as they all had the same amount of information.

3.3.2 Data set up

As pointed in the previous section, in th PPM format, each pixel value in the raster that represent the image is composed by 3 bytes, one byte for color red, gree and blue. Therefore, the total amount of bytes in only one picture of size 176 × 144 pixels is 76,032 ((176 × 3) × 144), hence the total amount of pixels in the database for the 588 images is 44,706,816 (76,032 × 588). This simple exercise gives an idea of the huge amount of information that needs to be processed and the implication on the time need to perform the computations. In order to further reduce the amount of data, a pixel out of 16 was selected leaving with only 1,584 pixels per image to work with. The database of images was created reading each of the PPM images files, discarding all the information using only the information of the pixels values.

4 Determination of the Sieve

The method consists of the extraction of a fixed number of pixels from an image captured by the robot along with their position (in the image) to perform a search in a database that will contain all images loaded into the robot's memory. With this set of pixels, a single image must be identified from all the images in the set, and the amount of pixels in the set must have to be the minimum. In figure F4.1 we illustrate an image and its possible representations by two different sieves. The main motivations to reduce the amount of information in the database are that it must fit into the robot's memory and that this information is to be processed in real time. The first issue is vital if there are limited resources of memory space and processing power in the robot. At first glance, to overcome this problem the robot's memory can be increased as well as its processing power, but there is a better choice. The database size may be reduced in a way that it only holds a minimum amount of data required to identify an image. To achieve the reduction in size, a number of pixels *p*, expected to be less than the total number of pixels that conforms an image (i.e. 1,584 in this case) must be found. Additionally the number of pixels *p* should be such that an image should not be wrongly identified. It is obvious that if the number of selected pixels in the sieve reaches the total number of pixels in the image, the probability to identify an image as a different one will be low. On the other hand, if the number of pixels is minimum, e.g. one pixel, the probability to misidentify an image in the database will be high. Therefore, two things have to be minimized:

a) The number of pixels *p* in the sieve required to identify an image,

b) The *image identification error*, ε_I .

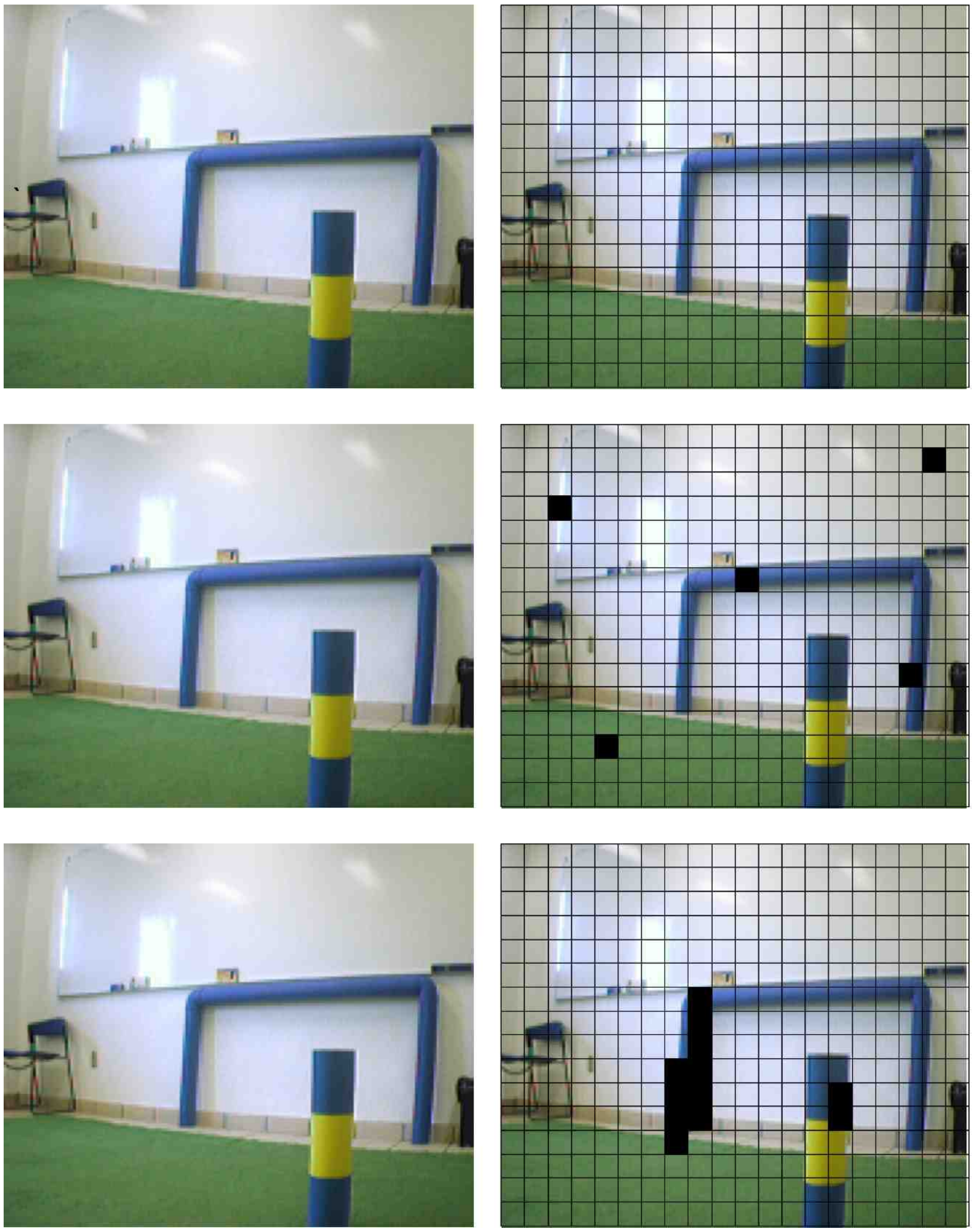
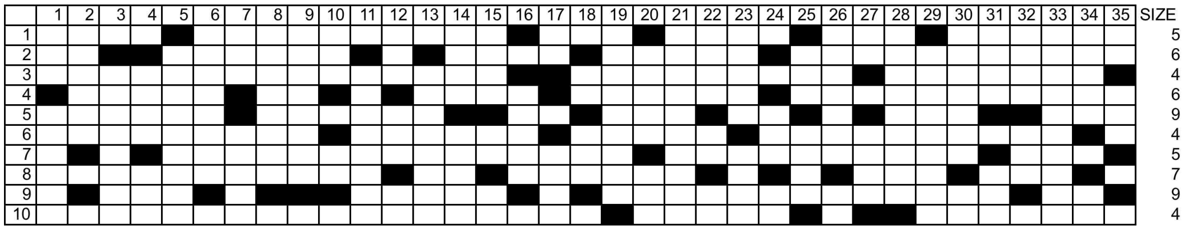


Fig. F4.1. An image of the environment and its possible representation by a set of pixels

This is a typical multi-objective optimization problem with conflicting goals. In section 2.2 we mentioned that there are different ways to approach these kinds of problems, e.g., by aggregating the objectives into a single objective function. A different approach is by using a Multi-Objective Evolutionary Algorithm (MOEA).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	SIZE
1					■											■				■					■				■							5
2			■	■							■		■					■						■												6
3																■	■										■								■	4
4	■						■			■		■					■							■												6
5							■							■	■			■				■			■		■				■	■				9
6										■							■						■											■		4
7		■		■																■											■				■	5
8												■			■							■		■		■				■				■		7
9		■				■		■	■	■						■		■														■			■	9
10																			■						■		■	■								4

Fig. F4.2. A sample view of 10 individuals for images of 35 pixels

The algorithm we propose uses a population of *n* individuals. Each individual genome is the size of the number of pixels in the image, i.e. 1,584. Every allele in the genome maps directly to a position of a pixel in the image. The set of alleles with a value of 1 represent the pixels that will work as a sieve, which will be used to select the pixels of all the images in the same position. In other words a bit turned on in the genome means that the pixel in that position will be part of the sieve. The number of pixels that will be part of the sieve is selected randomly. Figure F4.2 shows a set of 10 individuals created for hypothetical images of 35 pixels each one. Every dark rectangle in the image represents the position of the pixel selected to be part of the sieve. Each individual may have a similar or different sieve sizes as it is also shown in the figure. To evaluate the candidate sieve, a random image is selected from the set of images. The pixels in the sieve are then compared with the remaining images. Every time that the value of a pixel is in another image, a counter for the candidate sieve is increased. This counter will provide the number of repetitions found across all the images. The number of repetitions is called the *pixel identification error,* ε. The number of pixels repeated per image is called *the image identification error*, ε_I. To find ε_I, the pixel identification error ε is divided by the total number of images (588) and the number of pixels that form the sieve. The result is then multiplied by the number of pixels that correspond to a single image as shown in equation (E4.1).

$$\varepsilon_I = \frac{\varepsilon}{T_i s} N \qquad \text{(E4.1)}$$

where: ε_I is *the image identification error*, ε is *pixel identification error*, T_i is the number of images, s are the pixels in the sieve and N is the number of pixels in an image.

The population in the MOEA will produce a set of points that will be part of the solution space. If there is a subset of points in the solution space that dominates all the other solutions, then this subset will outline a Pareto front. This process is repeated each generation. If the number of pixels in the *sieve* falls in a predefined threshold, the algorithm has reached the optimal set of points in the sieve, which allow us to satisfy objectives a) minimize the size of the sieve s and b) minimize ε_I.

4.1 Adjustments performed to the RPSGAe algorithm

The MOEA used to solve this problem was the RPSGAe algorithm implemented in Java and introduced in section 2.2.5.3. Several runs were performed and the algorithm took more than 40 minutes to find a Pareto front. A closer analysis of the implementation revealed that the clustering procedure was consuming most of the processing time. As there were only two objective needed to be optimized, it was decided that the clustering module was unnecessary and therefore it was eliminated from the implementation. This simple modification speeded up the processing time to just a few minutes. It is important to point out that the RPSGAe algorithm leaves open the implementation of the genetic operators: selection, crossover and mutation. Therefore the genetic operators were implemented as in the Eclectic Genetic Algorithm [31] to include: Deterministic Coupling, Full Elitism and Annular Crossover. By doing so we can take advantage of a high degree of elitism and at the same time give variety to the genetic pool [31].

5 Obstacles in the field of view

With this method, the robot expects to find a perfect landscape in the sense that it ought to unequivocally determine at least one of the identifying sieves. There are two possible reasons for which an image may not be identified: a) There is too much noise in the image or b) There is an object (originally not present) which is obstructing the field of view of the robot. When this does occur, it must be that something happening in its environment modified the landscape in its field of view. In this hypothetical case, (and discarding the possibility of excessive noise in the environment) one must conclude that a new object (not present in the original scenery) is causing the ambiguity of position problem. In our case, rather than becoming a nuisance, this fact will allow us to a) Determine the presence of an extraneous object and b) Determine the precise nature of such object. If we assume a finite collection of extraneous objects we will be able to characterize them much the same way we were able to characterize the environment. Following this line of reasoning, we can introduce new data to determine what caused the change in the landscape and where this change occurred.

5.1 Enemy identification

Once the problem of identifying the robot's position has been dealt with, there is a second possible application of the method. Namely, it does allow us to take one-step further and identify the strange objects in the field of view. In the context of a soccer match such as the ones in RoboCup, the possible obstructions (and the consequent inability to identify an image from its sieve) must be due to the presence of an alien object not present in the course of image acquisition. These objects are hereinafter called "enemies". To be able to identify an enemy it is necessary to take pictures of each enemy to create a new set of images that will form an enemy database. Then we can identify them by applying morphologic operators, which make such enemy images position and scale independent. At that point, the non-identification of a robot's position is an advantage. Since it implies the detection of the presence of an undesirable (in the sense that a robot playing soccer must avoid it) obstacle. Furthermore, it allows the robot to identify the type and distance of the enemy relative to the present position. It is important to clarify that the enemy identification was not done as part of the present work; it was included here only as an example of a different application that the sieve method may have.

6 Validation of the sieve

To demonstrate the effectiveness of the Optimum Sieve method proposed in this work a statistical validation method was performed using two similarity measures:

1. The Hamming distance
2. The Euclidean distance

This section describes the similarity measures and the data used in the validation.

6.1 Data setup for the validation

To validate this method a new database of images was set up. The validation method applied to this database used both:

a) The complete set of pixels (1,584) and
b) The reduced set of pixels found by the MOEA.

The new database was setup by 28 images acquired from a unique point located in the center of the area showed in figure F3.4.1. The images have a separation of 5 degrees between them and sweep a range from zero to 30 degrees in the horizontal and from zero to 20 degrees in the vertical as oppose to the original set of images that had a separation of 30 degrees between them only in the horizontal. The robot's camera was facing forward and parallel to the ground.

6.2 Similarity measures

Given the nature of the problem there should be a way that can help determine if an image the robot is sensing at a specific time is similar to another image in the robot's database. If the two images are alike, then the probability that the robot is in the same position as the image in the database is high, and it will be low if the two images are different. Therefore, the robot is set in a different position. To be able to measure how different is an image from another a similarity measure between images is needed. In the present work, two different measures were used. One was the Hamming distance (between two sets of pixels). The other was the Euclidean (which determines how "far" or "similar" are the components of a pixel relative to another one).

6.2.1 Minkowski metric

Given two points $x = (x_1, x_2)$ and $y = (y_1, y_2)$, a general measure of their closeness is the p-Minkowski [11] metric given by

$$d^p(x,y) = \left[|x_1 - y_1|^p + |x_2 - y_2|^p\right]^{\frac{1}{p}} \qquad \text{(E6.2.1.1)}$$

where p is a positive real number such that $p \geq 1$. For $p=1$, $d(x,y)$ is the Hamming distance. For $p=2$, $d_2(x,y)$ is the well known Euclidean distance. For $p=\infty$ equation (E6.2.1.1) reduces to

$$d^\infty(x,y) = \max\left[|x_1 - y_1|, |x_2 - y_2|\right] \qquad \text{(E.6.2.1.2)}$$

6.2.1.1 Hamming distance

A particular case of the above is the so-called Hamming distance, which is defined as:

$$d = \left(\sum \left(I(x,y) - I'(x,y)\right)\right) \qquad \text{(E.6.2.1.1.1)}$$

The Hamming distance [14] determines the number of positions at which the corresponding symbols of two or more hypothetical messages are different. In our case, we focus on the differences between any two pixels. The higher the number of different positions, the more different they are.

Pixel A	1	0	1	1	1	0	0	1	0		
Pixel B	1	1	1	0	1	1	0	1	0		
	0	1	0	1	0	1	0	0	0	=	3

Fig. F6.2.1.1.1. Example of the Hamming distance between two binary strings

6.2.1.2 Euclidean distance between pixels

From the Minkowski metric shown in section 6.2.1, the Euclidean distance can be defined as:

$$d = \left(\sum \left(I(x,y) - I'(x,y)\right)^2\right)^{\frac{1}{2}} \qquad \text{(E6.2.1.2.1)}$$

To calculate the distance between two pixels, each pixel was considered as a three dimensional vector of values (R, G, B). Therefore, the distance between two pixels is obtained with equation (E6.2.1.2.2).

$$d = \left((r - r')^2 + (g - g')^2 + (b - b')^2\right)^{\frac{1}{2}} \qquad \text{(E6.2.1.2.2)}$$

It is important to notice that the pixels used to calculate the distance share the same position in their corresponding image. That is, to get the distance between pixel A and pixel B both must have the same position in image 1 and image 2.

The distance between the pixels of two different images is obtained one by one and then added to obtain the aggregated pixel distance. This distance is divided by the total number of pixels of the images. At the end we will have the average pixel distance between two images. This average pixel distance is use to determine how similar are the images compared. A shorter distance will correspond to similar images a greater distance will correspond to dissimilar images.

After calculating the distance between every pixel for each pair of images in the database, the average distance between two images was obtained with equation (E6.2.1.2.3)

$$\overline{D} = \frac{1}{N}\sum_{i=0}^{N} d_i \qquad \text{(E.6.2.1.2.3)}$$

where N is the total number of pixels of an image.

6.2.1.3 Euclidean distance between images

A more typical way to calculate the distance between two images is to consider each image as a vector. The distance is obtained using equation (E6.2.1.3.1)

$$d = \left(\sum_{i=0}^{numpix} \left(\left(r_i - r_i'\right)^2 + \left(g_i - g_i'\right)^2 + \left(b_i - b_i'\right)^2 \right) \right)^{\frac{1}{2}} \qquad \text{(E6.2.1.3.1)}$$

6.3 Statistical validation

The total number of comparisons performed to calculate the distances was obtained with equation (E6.3.1)

$$Comparisons = \frac{T_i}{2}\left(T_i - 1\right) \qquad \text{(E6.3.1)}$$

where *Ti* is the total number of images in the dataset.

For a database with 28 images, 378 comparisons were made. The comparisons were performed exclusively between different images, as there was no reason to compare an image with itself, as it is already known that the distance of two images alike is zero. Additionally images that were already compared were not compared again i.e. if *image_a* vs. *image_b* were compare previously there was no need to compare *image_b* against *image_a* as it will yield the same result.

7 Results

7.1 MOEA results

For a population of 10 individuals several runs were performed to find the optimal sieve for the 588-image database. Figure F4.2 is a sample view of the genetic coding for 10 hypothetical individuals. Figure F7.1.1 shows the Pareto fronts result from these runs. The graphic shows the number of errors obtained for a specific size of the sieve. The reader can notice that as the number of points in the sieve decreases the number of errors increases and likewise as the number of points in the sieve increases the number of errors decreases. The proposed algorithm does not implement the self-adapting feature present in the original EGA therefore the following parameters were used: Population: 10, Generations: 50, Crossover factor: 0.8, Mutation factor: 0.01 and a random seed: 5123412.

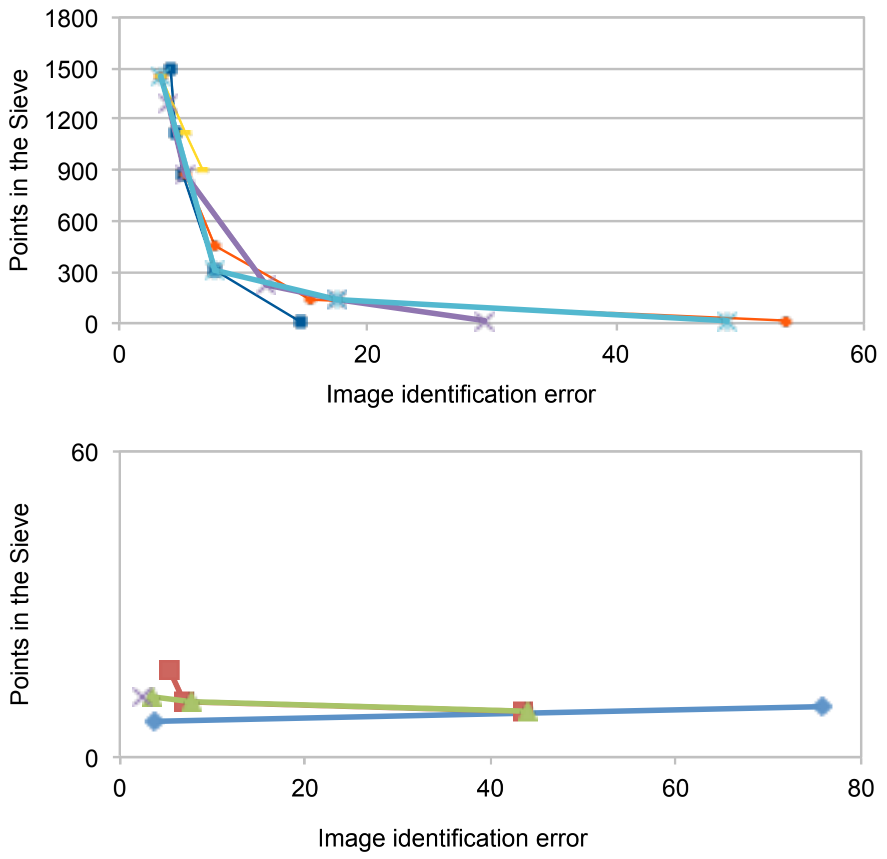


Fig. F7.1.1. Pareto fronts found for the original 588-image database

Table T7.1.1 and Table T7.1.2 show the numerical results for just one of the runs of the Pareto front from the first 3 and the last 3 generations.

Generation 0	
Errors	Points in the Sieve
4.173461	1502
4.527774	1127
5.046563	878
7.7246842	313
14.57172	5

Generation 1	
Errors	Points the Sieve
3.3837917	1449
5.4062386	870
7.7692122	455
15.411243	135
53.789436	15

Generation 2	
Errors	Points in the Sieve
3.3837917	1449
5.244737	1117
6.6968594	903

Table T7.1.1. Results for the first 3 generations

Generation 26	
Errors	Points in the Sieve
5.2382	17
6.8688245	11
43.4753	9

Generation 27	
Errors	Points in the Sieve
3.3730834	12
7.60477	11
44.07496	9

Generation 28	
Errors	Points in the Sieve
2.4735944	12

Table T7.1.2. Results for the last 3 generations

From these runs, the best suitable solution at generation 28 with a single element in the Pareto front corresponds to the data in Table T7.1.3 which shows the 12 pixels sieve found by the algorithm that produce an error of 2.4735944.

N	PIXEL	R	G	B
0	1	161	159	82
1	2	244	255	248
2	5	239	251	250
3	7	243	255	249
4	8	244	254	249
5	13	244	255	248
6	18	244	255	248
7	182	244	255	248
8	748	45	41	51
9	1219	136	179	161
10	1369	246	253	248
11	1465	44	46	44

Table T7.1.3. The sieve found for the 588 images database (RGB values are representative).

7.1.1 Determination of the sieve for the new database

Although an optimal sieve was known for the original database, a new optimal sieve was required for the new database of 28 images described in section 6.1 to validate the method. These data will be used as a training data as we will obtain a new sieve for them. Then they will be compared against each other. As pointed out previously, unlike the EGA, the proposed algorithm does not self-adapt its parameters. Therefore the following parameters were used in the MOEA to obtain the optimal sieve of pixels that describes an image for the new database: Population: 50, Generations: 100, Crossover rate: 0.6 and Mutation rate: 0.01.
The sieve found for this new set of images had a size of 405 pixels. As can be noted, the size of the sieve is considerably larger than the one for the original set of images. It is important to remember that the separation in the new set of images was smaller than the original set of images (it had 5º of separation between each image as oppose to the 30º). Therefore the images are similar to each other than the images in the original set.

7.2 Validation of Results

As pointed out in section 6.1 the validation of the resulting sieve was derived from the two similarity measures (Hamming and Euclidean distance) for the complete set of pixels (1,584). To present the results as clearly as possible, the statistical analysis using the 1,584 pixels of the images will be presented first. Then the same analysis will be shown using the 405 pixels sieve. It is also important to clarify that the validation was performed on the new set of 28 images and not on the original set of 588 images.

7.2.1 Image Similarities using 1,584 pixels

The maximum average Euclidian distance found between two images was 117.46. Its corresponding Hamming distance was 11.32 Figure F7.2.1.1.

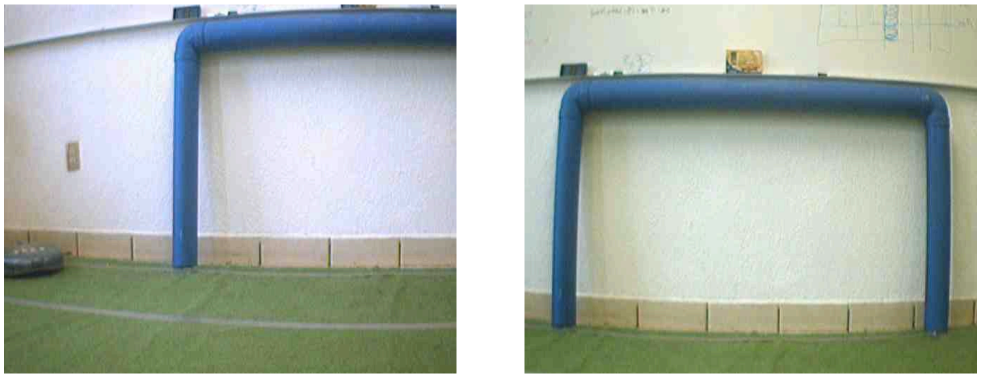


Fig. F7.2.1.1. These two images have the maximum average distance of 117.46 using 1,584 pixels per image.

The minimum Euclidian distance found was 27.35 and its corresponding Hamming distance was 8.05 Figure F7.2.1.2.

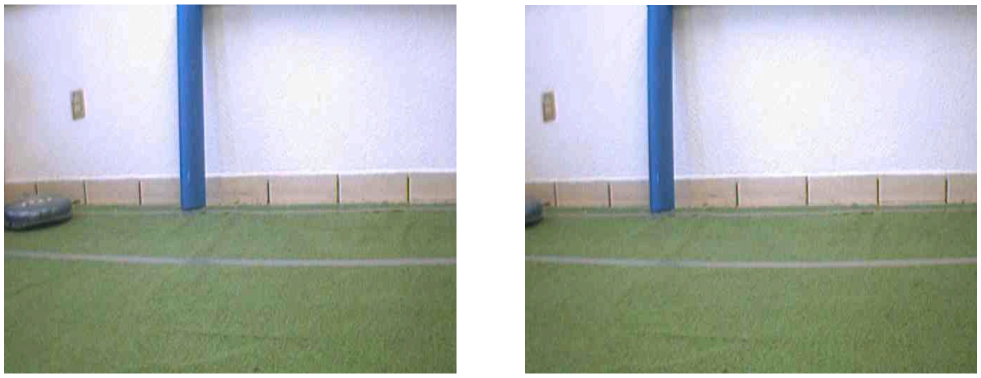


Fig. F7.2.1.2. These images have the minimum average distance of 27.35 using 1,584 pixels per image.

The average Hamming distance of the complete dataset is 10.67 and its standard deviation is 0.93 as shown in Fig. F7.2.1.3.

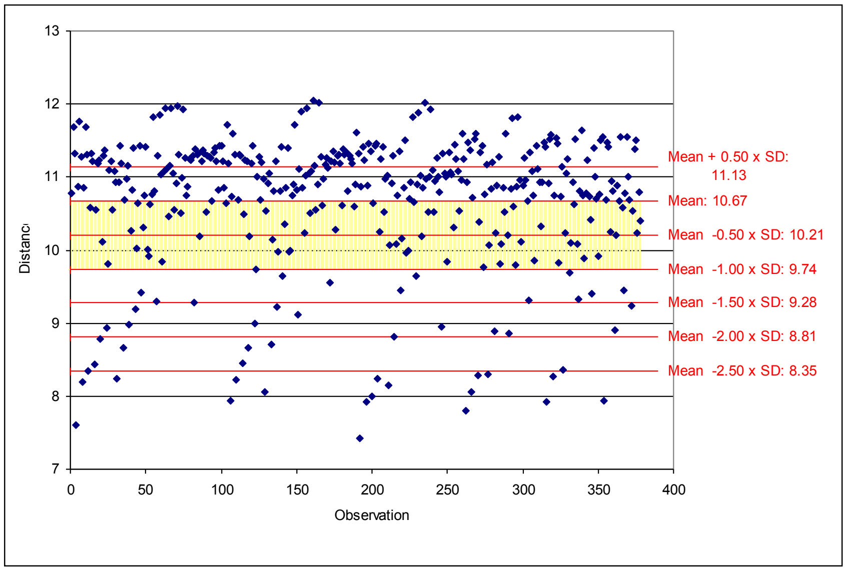


Fig. F7.2.1.3. Hamming distance of the set of 28 images compared using the 1,584 pixels of the image.

The average Euclidean distance is 83.22 and the standard deviation is 21.75 as shown in Fig. F7.2.1.4.

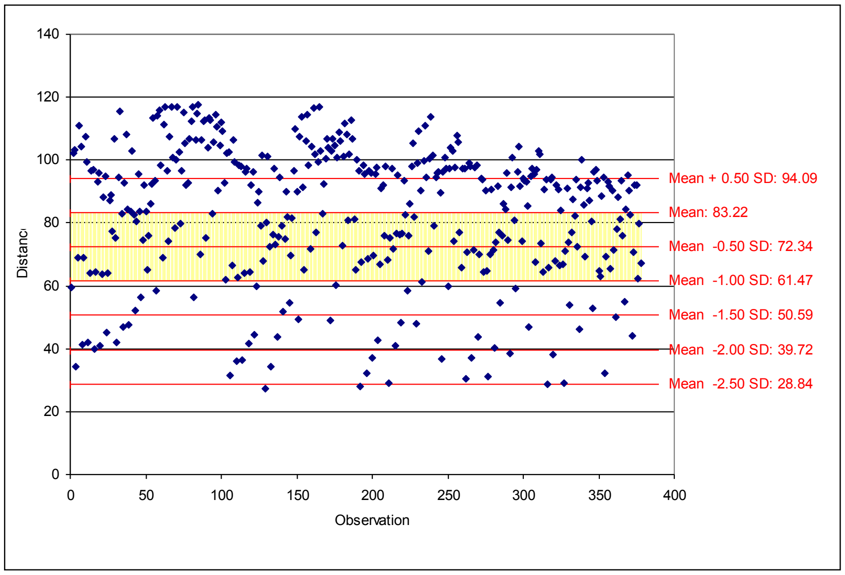


Fig. F7.2.1.4. Euclidean distance of the 28 images compared using the 1,584 pixels of the image.

From figures F7.2.1.3 and F7.2.1.4 it can be seen that two images with a distance below the mean are similar to some degree. Moreover, images extraordinarily similar are those whose distance is two times the standard deviation below the mean. This observation is important because it can allow us to establish a degree of certainty when one image is compared with another. The results obtained from the comparisons of the images with the 1,584 pixels are important because they establish a metric with which we are going to compare the images using the sieve.

7.2.2 Image Similarities using a pixel sieve of 405 pixels

When using the 405-pixel sieve, the maximum Euclidian distance was 121.16 and its corresponding Hamming distance was 11.54 Figure F7.2.2.1.

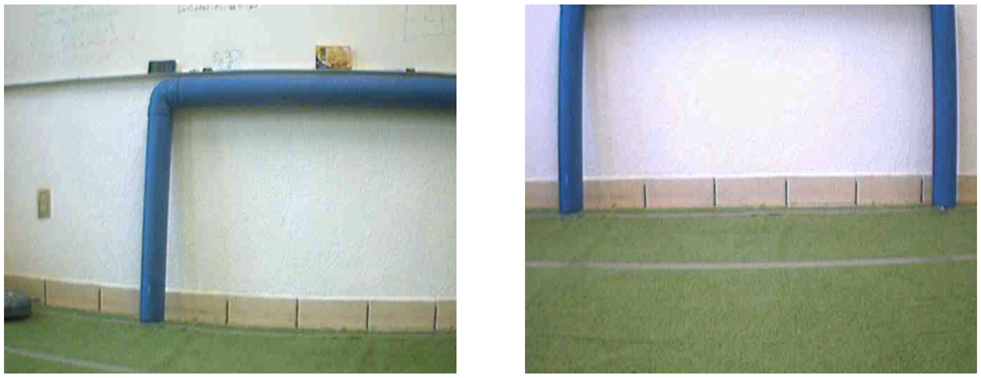


Fig. F7.2.2.1. These two images have the maximum average Euclidean distance of 121.16 using a 405-pixel sieve.

The minimum Euclidian distance was 27.86 and its corresponding Hamming distance was 7.85 Figure F7.2.2.2.

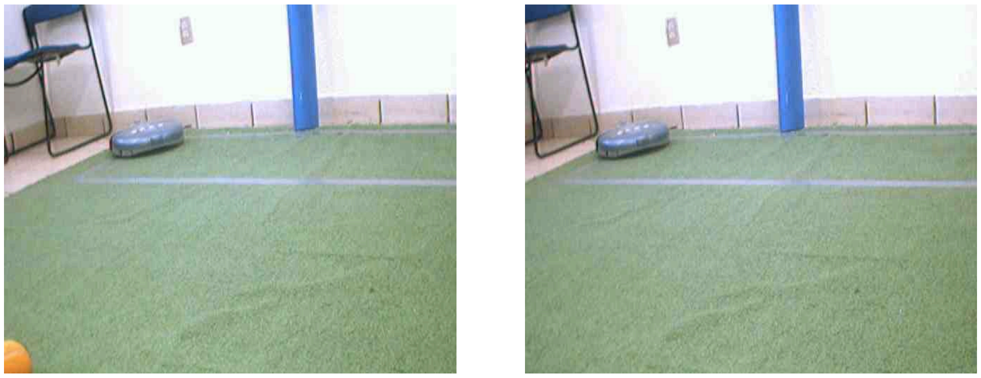


Fig. F7.2.2.2. These two images have the minimum average Euclidean distance of 27.86 using the 405-pixel sieve.

The average Hamming distance is 10.69 and the standard deviation is 0.92 as shown in Figure F7.2.2.3.

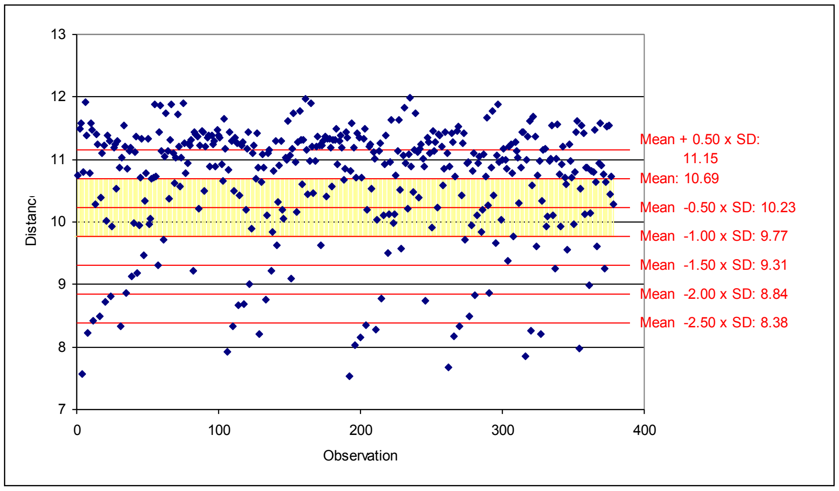


Fig. F7.2.2.3. Hamming distance of the images compared using the pixel sieve.

The average Euclidean distance is 84.69 and the standard deviation is 22.43 as shown in Figure F7.2.2.4.

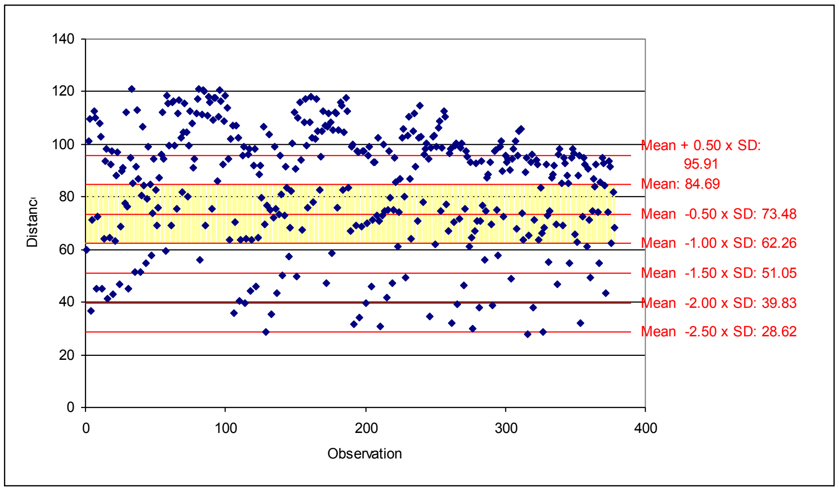


Fig. F7.2.2.4. Euclidean distance of the images compared using the pixel sieve.

The reader can notice that figures F7.2.2.3 and F7.2.2.4 are similar to figures F7.2.1.3 and F7.2.1.4 and they display the same behavior. Images described by the sieve whose distance is below the mean are similar. In addition, as in section 7.2.1, images whose distance is two times the standard deviation below the mean are also extraordinarily similar.

7.2.3 Validation using control and test images

With the results obtained from the validation tests described in sections 7.2.1 and 7.2.2 we concluded that the images with a short distance value are similar in some degree that can be established using the standard deviation and most importantly that we can use only the points in the sieve instead of the whole set of pixels to compare the images. This last conclusion is important as the information to be stored in the robot is decreased and also the time needed to compare the images is also reduced. For instance, the size of the original 588-image database used in this work is of 2.8 MB, after the sieve was found a new image database was created sampling only the pixels in the sieve. The size of this new database was of 22 KB. With this in mind we perform a new test. We took new pictures of the environment and compared them against the 588-image database using the 12-pixel sieve previously found and reported in section 7.1. The reason behind using the 588-image database is that this set of images covers a wider view of the environment. The new images were compared using the method described in section 7.2.2. It is important to notice that the images were taken long after the original set of images and the environment changed considerably since then. As result the test images had new objects, some were moved from their original location, and other objects were removed. This may be considered as near real case scenario. The hypothesis we want to validate is to be able to recognize the location where a picture was taken by comparing an image with the set of 588 images using only the sieve found Figure F7.2.3.1 shows the result of comparing 4 images taken from a known position using the 1,584 pixels. The accuracy percentage of the identification of the position for these images is of 83% using the Hamming distance, 92% using the Euclidean distance between pixels and 83% using the Euclidean distance between images. The same was done using the 12-pixels sieve found. Figure F7.2.3.2 shows the positions result from the comparisons. In this case the accuracy percentage decreased in all cases to 67% using the Hamming distance, to 83% using the Euclidean distance between pixels and to 67% using the Euclidean distance between images. Based only on these percentages[7] we may say that the Euclidean distance between pixels is the best option to decide the position of the robot.

[7] These percentages were obtained from the average of the times x, y and θ were successfully found.

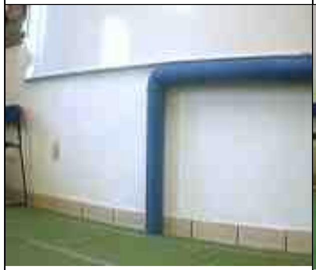
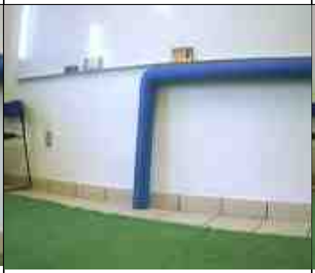
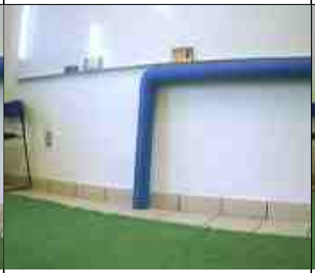
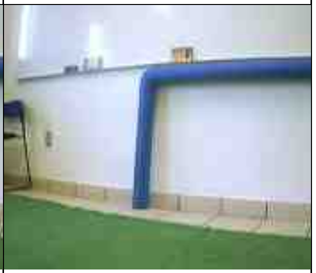
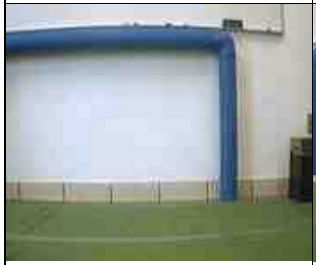
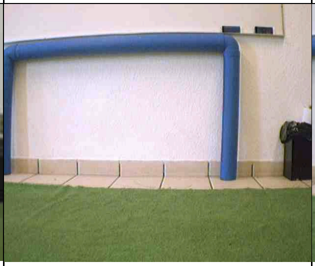
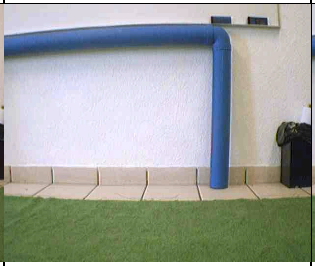
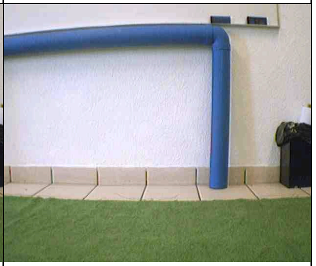
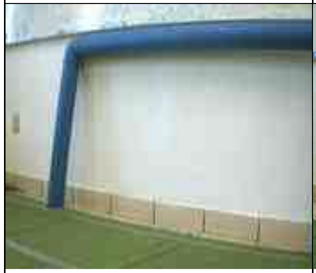
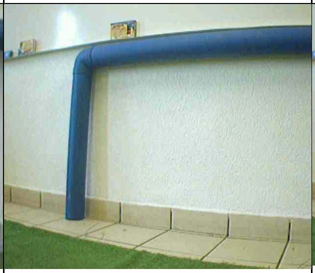
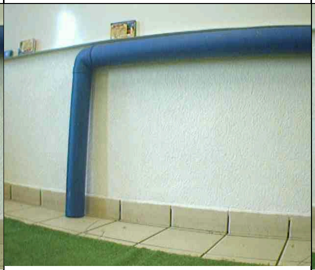
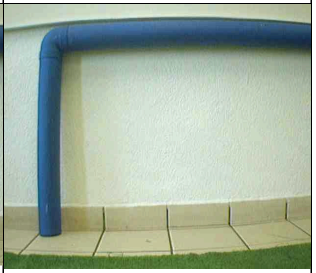
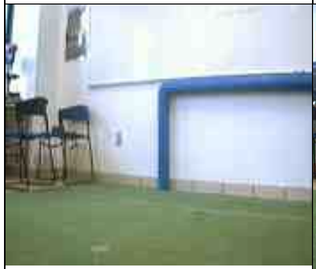
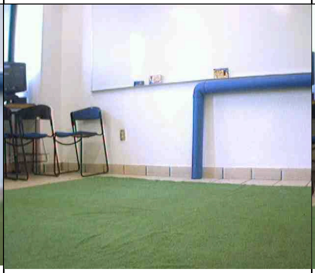
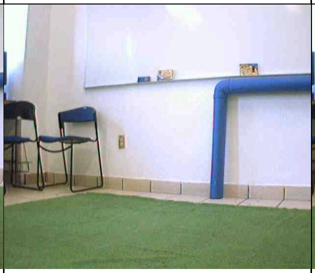
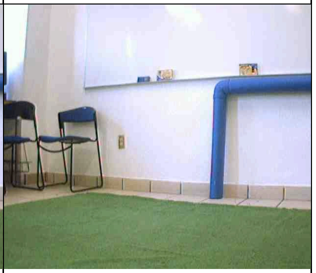
Control images taken from a known position	Position found with the Hamming distance.	Position found with the Euclidean distance between pixels	Position found with the Euclidean distance between images
			
4x, 3y, 60θ	4x, 3y, 60θ	4x, 3y, 60θ	4x, 3y, 60θ
			
4x, 3y, 90θ	3x, 3y, 90θ	4x, 3y, 90θ	4x, 3y, 90θ
			
6x, 3y, 60θ	6x, 3y, 60θ	6x, 3y, 60θ	6x, 6y, 90θ
			
2x, 2y, 60θ	1x, 2y, 60θ	2x, 4y, 60θ	2x, 4y, 60θ

Fig. F7.2.3.1. Positions found using all the 1,584 pixels of the on control images.

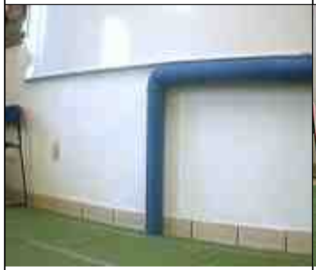
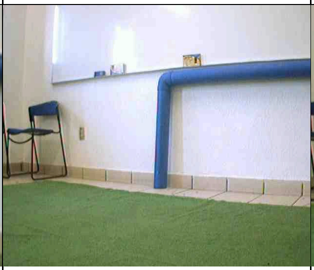
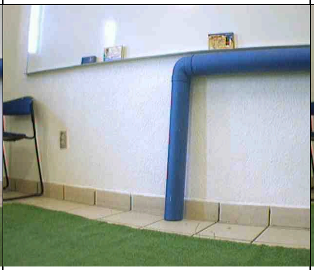
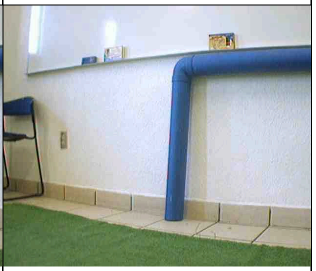
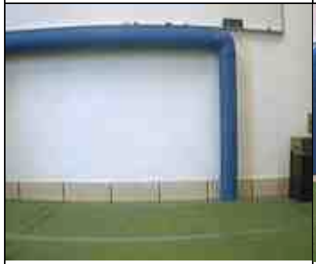
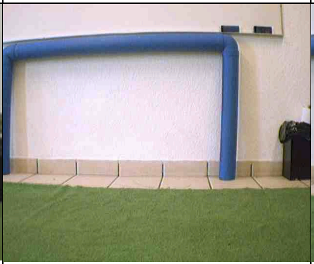
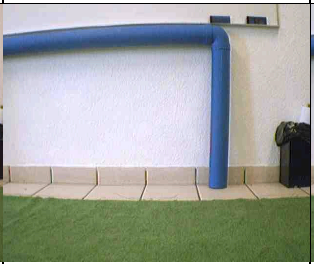
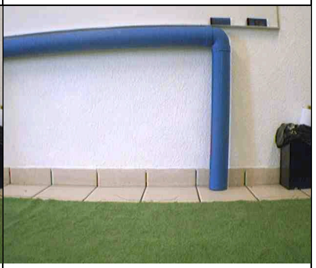
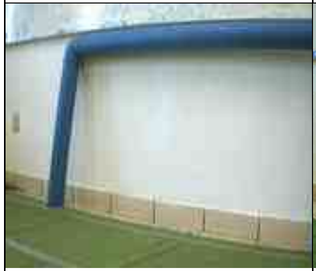
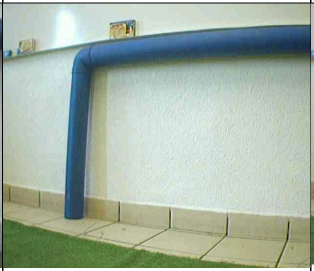
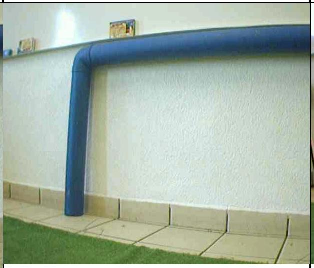
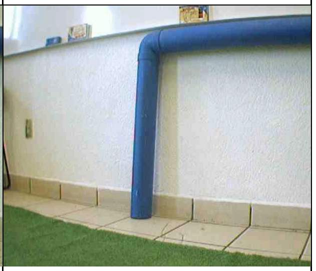
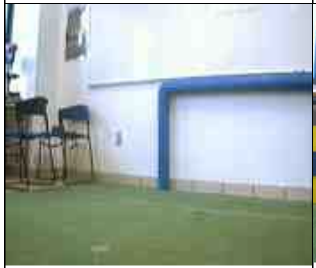
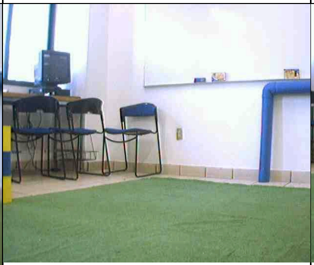
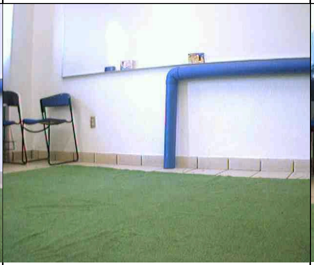
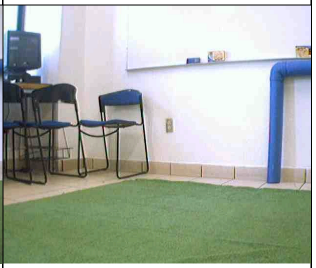
Control images taken from a known position	Position found with the Hamming distance.	Position found with the Euclidean distance between pixels	Position found with the Euclidean distance between images
			
4x, 3y, 60θ	4x, 2y, 60θ	5x, 4y, 60θ	5x, 4y, 60θ
			
4x, 3y, 90θ	3x, 3y, 90θ	4x, 3y, 90θ	4x, 3y, 90θ
			
6x, 3y, 60θ	6x, 3y, 60θ	6x, 3y, 60θ	6x, 4y, 60θ
			
2x, 2y, 60θ	0x, 4y, 60θ	2x, 2y, 60θ	2x, 5y, 60θ

Fig. F7.2.3.2. Positions found using the 12-pixel sieve on control images.

Conclusions

The results obtained in the validation procedures show that the sieve has the same statistical behavior as the complete number of pixels that define an image. The accuracy percentages shown at the end of section 7.2.3 where we performed the image identification using the 1,584 pixels and the pixels in the sieve are an example of this behavior. The decrease in the accuracy percentage when using the sieve is due the image identification error introduced by the sieve that is roughly of 2.5. The accuracy will be higher if a new sieve with a smaller image identification error is found and used instead. This, as in all multi-objective problems, may present a tradeoff between the size of the sieve and the image identification error and will impact directly in the size of the database to be stored in the robots memory. Based on the previous analysis a robot can determine if an image acquired with its camera is similar to an image in its local database with a degree of certainty based on how far an image is from another. In the present work three different ways to measure the similarity between the images were used. These similarity measures can be used to decide the position of the robot. Hence it is viable to provide a database containing just the essential information needed by a robot to locate itself in a known area by searching in it the pixels determined by the proposed algorithm. In addition, the method allows not only to identify the robot's position but also alien objects within the robot's line of sight. Therefore, we could, in principle, determine where an obstacle stands for the robot to take possible evasive actions. In the case of unrecognized images, this methodology can be applied to identify foreign objects in a scene, assuming that the forms of the foreign objects are already known with or without the use of scale invariant operators to describe such objects. This could be a starting point for future works based on this method. It is a fact that the method requires that the area in which the robot will navigate is known beforehand. Mainly because of this it is necessary to acquire the images for the offline training. This opens the opportunity to find new ways to obtain such images in a better way.

8 Appendix A. The Eclectic Genetic Algorithm

As it was mentioned in section 4.1, some of the operators in the MOEA used in this work were modified to use some of the operators of the Eclectic Genetic Algorithm (EGA). To give a better understanding to the reader on how these operators work in the EGA, a modified version of the original article [31] is included in the current appendix.

Over the years, the area of Genetic Algorithms has seen the rise of several strategies to overcome issues as a) whether it will converge to an acceptable solution and, b) How to choose the parameters of the GA (such as crossover probability, population size, etc.) to achieve efficiency. The EGA addresses the said issues by taking the "best" out of all strategies found in Genetic Algorithms approaching what, in the literature, has been called an Idealized GA (IGA). An IGA should have:

a) Full elitism over a set of size *n* of the last population.
b) A deterministic selection scheme (as opposed to the traditional proportional selection operator).
c) Annular crossover.
d) Selective invocation of a Random-Mutation Hill Climbing (RMHC).
e) Population self-adaptation of the following parameters: *n* (the number of offspring), P_c (probability of crossover), P_m (probability of mutation), PH (probability of RMHC's invocation).

8.1 Full Elitism

Full elitism means that a copy of the best *n* individuals is kept up to generation *k*. In other words, given that *nk* individuals have been tested by generation *k*, the population will consist of the best *n* up to that point.

Therefore, the population is forced to lean towards a focalized set of hyperplanes. The obvious risk is that the search space gets restricted in a way that hampers the GA excessively. To avoid this, the selection operator is modified as described in section 9.2.

8.2 Deterministic Selection

Deterministic selection does not rely on the individual fitness to determine the most desirable descendants. It rather, emphasizes a genetic variety by imposing a strategy that enforces crossover of predefined individuals. In this model a new strategy is adopted to select and individual *i* deterministically and cross it with individual *n-i+1*. This selection scheme is called the *Vasconcelos Model* (VM). At a first glance, this strategy seems to destroy the good traits of the best individual by deliberately crossing it with the worst individual. However, when taken in conjunction with full elitism it leads to the implicit analysis of a wider variety of schemas (i.e. it maximizes the exploration of the solution landscape). The exploration of such vast landscape is focused via the full elitism implicit in the model.

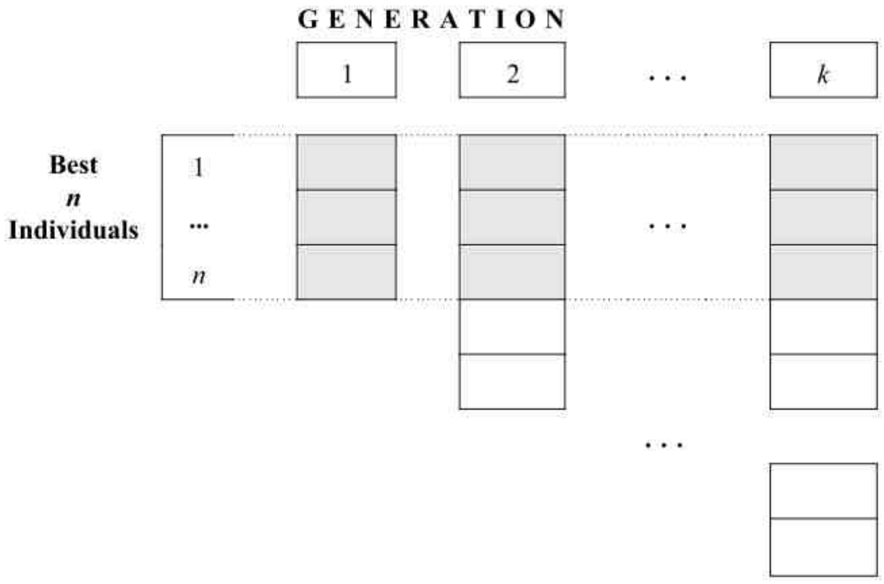


Fig. F8.1.1. Full Elitism.

The desired schemas remain present as full elitism is being use and, as before, crossing over dissimilar individuals avoid hitchhiking. Additionally, the crossover rate has to be such that the time for crossover that combines two desired schemas is small with respect to the discovery time for such schemas. Full elitism guarantees that, even in the Vasconcelos strategy, the worst individual for population *k* is in the top *1/k* percentage. For example, if *n* = 50 a look at the 20th generation, the individuals in such population will be among the best 5% of the total individuals analyzed. This characteristic is incrementally exposed as the GA proceeds.

8.3 Annular Crossover

In annular crossover, the genome (as shown in figure Fig.9.3.1.) is no longer seen as a linear collection of bits but rather as a ring whose leftmost bit is contiguous to its rightmost bit.

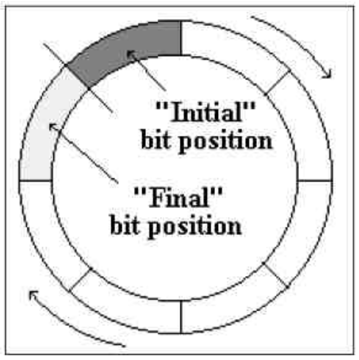


Fig. F8.3.1. Annular Genome.

When applying annular crossover, there are two parameters to consider for each interchange:

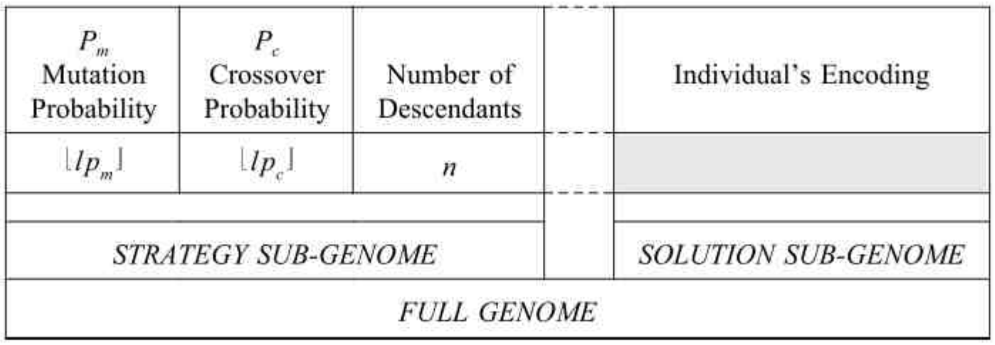
a) The starting crossover locus. That is, where the segment to be extracted from the individual starts.
b) The length of the semi-ring. That is, how many genes of the individual are going to be extracted. Clearly, for a genome of length *l* there are *l* possible locus and *l*-1 possible lengths. When extracting the first individual's genes, however, we must no longer concern ourselves with position encoding dependencies.

8.4 Self-Adaptation.

Although this feature was not added in the MOEA implemented for this work as mentioned in sections 4.1, it was included to show the EGA characteristics completely. When running a GA there are several parameters that are to be set a priori: Three of these are the most common:

a) The mutation rate (P_m)
b) The crossover rate (P_c), and
c) The size of the population (N).

In many cases the user tries to fine tune these parameters by making a number of runs on different "representative" case problems**.** In a self-adaptive GA the three parameters are included as an extension of the genome in such a way that the parameters evolve along with the individual. The idea behind self-adaptation is that not only does the GA explore the solution landscape but the parameter landscape as well. In this way, the genome is divided in two sub-genomes: a strategy genome and a solution genome. Both sub-genomes are subject to the genetic operators. We should consider the parameter sub-genome as a set of three sub-genomes which are functionally independent. The self-adaptive genome is shown in figure Fig 9.4.1.



P_m Mutation Probability	P_c Crossover Probability	Number of Descendants		Individual's Encoding
$\lfloor lp_m \rfloor$	$\lfloor lp_c \rfloor$	n		▨
STRATEGY SUB-GENOME				*SOLUTION SUB-GENOME*
FULL GENOME				

Fig. F8.4.1. Self-Adaptive Genome.

Notice that the size of the population N is implicitly dealt with by considering not the population's size itself, but rather the number of descendants produced by the genetic operators.

In this case of self-adaptation the way the operators affect the performance of the GA takes into consideration the population (for any given generation) as a whole. In a sense, this self-adaptive GA is aimed at improving the mean values of the population rather than the values of each individual.

8.4.1 Probability of mutation

P_m is encoded in every individual. The mutation rate for the whole population in the k-th generation g_k is calculated as follows:

$$\left(P_m\right) = \frac{1}{N}\sum_{i=1}^{N}\left(P_m\right)_i \qquad \text{(E8.4.1.1)}$$

Therefore, the mutation operator's rate is fixed for all the individuals during g_k.

8.4.2 Probability of Crossover

P_c is, as before, encoded in the genome of every individual. Now, in generation g_k the crossover rate is given by

$$(P_c)_k = \frac{1}{N}\sum_{i=1}^{N}(P_c)_i \qquad \text{(E8.4.2.1)}$$

The crossover operator's rate is fixed for all the individuals during g_k.

8.4.3 Number of Descendants

The number of descendants, *n,* is also encoded in the genome of every individual and the number of descendants n_k in the *k*-th generation is given by

$$n_k = \frac{1}{N}\sum_{i=1}^{N} n_i \qquad \text{(E8.4.3.1)}$$

The number of descendants is fixed for all the individuals during g_k.
Using a self-adaptive strategy as the one mentioned here frees the user from the initial arbitrary parameter selection, to a certain extent. It is usual to set upper bounds on the possible values encoded in P_m, P_c and n. In that sense, there is still an arbitrary selection of initial parameter values. However, individuals which represent the better parameters for the particular problem are allowed to learn from the problem that is being solved.
The EGA approaches the behavior of an idealized Genetic Algorithm. Furthermore, in including a self-adaptive behavior it modifies its behavior without impairing the desirable characteristics.

8.5 Adaptive Hill Climbing.

Finally as part the EGA an RMHC is used when the GA is very close to find the solution. That is, the EGA consists of a self-adaptive GA plus a HC. The HC is activated also using a self-adaptive mechanism.

a) The first step is to define two bounds:
   1. The minimum Hill Climber percentage ($\eta\lambda$).
   2. The maximum Hill Climber percentage ($\eta\mu$).

where

$$0 < \eta_\lambda < \eta_\mu \leq 1 \qquad \text{(E8.5.1)}$$

The HC algorithm will then be activated, at least, η_λ of the time and, at most, η_μ of the time.

b) As a second step, we must define two other bounds:
   1. The minimum number of function evaluations to be performed by the HC upon invocation (N_λ).
   2. The maximum number of such function evaluations (N_μ).

These two bounds are given as a percentage of the population's size *N*. The actual percentage of HC function evaluations (relative to *N*) is included in the strategy sub-genome and is subject to the genetic operators. The actual number of evaluations up-on HC invocation (N_η) in generation k is given by

$$(N_\eta)_k = \frac{1}{N}\sum_{i=1}^{N}(N_\eta)_i \qquad \text{(E8.5.2)}$$

c) At generation g k the hill climber's effectiveness is evaluated from

$$\eta_\phi = \frac{1}{N}\sum_{i=1}^{N}(\iota_\eta)_i \qquad \text{(E8.5.3)}$$

where

$$\iota_\eta = \begin{cases} 1 & \text{if individual was found by the HC} \\ 0 & \text{otherwise} \end{cases} \qquad \text{(E8.5.4)}$$

To be able to determine ι_η's value we must include a new element in the ι_η genome. This element is of type boolean. It will be set when the individual has been found by the HC and reset otherwise. The genome for the EGA is shown in figure Fig.9.5.1.

d) Denoting the probability of invoking the HC by PH, we have:

$$P_H = \begin{cases} \eta_\lambda & \text{if } \eta_\phi < \eta_\lambda \\ \eta_\phi & \text{if } \eta_\lambda \le \eta_\phi \le \eta_\mu \\ \eta_\mu & \text{if } \eta_\phi > \eta_\mu \end{cases} \qquad \text{(E8.5.5)}$$

P_m Mutation Probability	P_c Crossover Probability	N Number of Descendants	ι_η Originated by HC	HC Function Evaluations	Individual's Encoding
$\lfloor lp_m \rfloor$	$\lfloor lp_c \rfloor$	n	η	N_η	▨
STRATEGY					*SOLUTION*

Fig. F8.5.1. Self-Adaptive Genome for Eclectic Genetic Algorithm.

e) Generate a random number ρ_K, where $0 < \rho_K \le 1$. Prepare to invoke the HC if $P_H \le \rho_K$.

f) Once the HC is scheduled to start, the string upon which it will operate is selected randomly from the first five in the population. Recall that the individuals in the population are ordered from best (individual 1) to worst (individual N). Therefore, to select the string ι_η upon for the HC to operate, we make

$$\iota_\eta = \lfloor R \times 5 \rfloor + 1 \qquad \text{(E8.5.6)}$$

where R is a random number uniformly distributed and $0 < R \le 1$.

The result of the strategy just outlined is to guarantee that the HC will be active whenever it has proved to be effective. The probability that the HC will override the GA proper is bounded above by η_μ. This prevents the HC from taking over the whole process. On the other hand, the HC will be invoked with $P_H \ge \eta_\lambda$ which, in turn, avoids the possibility that due to poor performance of the HC at some generation gk, the HC is shut out for the rest of the process. The EGA incorporates an HC process that is self-adaptive on two accounts:

a) Because its activity is determined by its effectiveness, and
b) Because the adequate number of function evaluations is evolved as the GA unfolds.

It is important to clarify that the algorithm proposed in this work is not an extension of the EGA to solve multi-objective problems, it rather leverages some of the operators found in EGA: selection, crossover and mutation, and incorporates it in the RSPGA algorithm to found the sieve.

9 References

1. Betke Margrit y Gurvits Leonid. Mobile Robot Localization Using Landmarks. IEEE Transactions on robotics and automation, vol. 13, no. 2, 251-263. April 1997
2. Boley, D.L., Steinmetz, E.S., Sutherland, K.T. Robot localization from landmarks using recursive total least squares. Robotics and Automation, 1996. Proceedings., 1996 vol 2. 1381 – 1386
3. Seong Jin Kim; Byung Kook Kim; , "Dynamic localization based on EKF for indoor mobile robots using discontinuous ultrasonic distance measurements," Control Automation and Systems (ICCAS), 2010 International Conference on, pp.1912-1917, 27-30 Oct. 2010
4. Hua Wu; Shi-Yin Qin; , "A new method of distance estimation for robot localization in real environment based on manifold learning," Wavelet Analysis and Pattern Recognition, 2007. ICWAPR ’07. International Conference on , vol.2, no., pp.585-590, 2-4 Nov. 2007 doi: 10.1109/ICWAPR.2007.4420737
5. Roland Siegwart and Illah R. Nourbakhsh. Introduction to Autonomous Mobile Robots. MIT Press, 2004. ch. 4, Perception, pp. 89-91
6. RoboCup Objective http://www.robocup.org/about-robocup/objective/
7. Mitchell, M., An Introduction to Genetic Algorithms, MIT Press, 1998, ch. 4, Theoretical Foundations of Genetic Algorithms, pp. 96- 99.
8. C. Zopounidis and P.M. Pardalos (eds.), Handbook of Multicriteria Analysis, Applied Optimization, Chapter 10 On Multi-Objective Evolutionary Algorithms, 287-310 103, DOI 10.1007/978-3-540-92828-7_10, © Springer-Verlag Berlin Heidelberg 2010
9. Portable Pixel Map specification. http://netpbm.sourceforge.net/doc/ppm.html
10. A. Gaspar-Cunha RPSGAe – Reduced Pareto Set Genetic Algorithm: A Multi-objective Genetic Algorithm with Elitism. Workshop on Multiple Objective Metahueristics, Carre de Sciences, Paris France. November 2002.
11. J. Koplowitz and G.T. Toussaint, "A unified theory of coding schemes for the efficient transmission of line drawings," Proc. of 1976, IEEE Conf. on Communications and Power, Montreal, October 1976, pp. 205-208.
12. W. E. Aguilar Martinez, “Reconocimiento de objetos basado en la correspondencia estructural de caracteristicas locales”, Mexico, 2006. 140 p. Tesis Maestria (Maestria en Ciencia e Ingenieria de la Computacion)-UNAM, Facultad de Ciencias

13. Juan Manuel Ibarra Zannatha, Rafael Cisneros Limón, Ángel David Gómez Sánchez, Eric Hernández Castillo, Luis Enrique Figueroa Medina, and Felipe de Jesús Khamal Lara Leyva. Monocular visual self-localization for humanoid soccer robots. Proc of CONIELECOMP 2011, 21st International Conference on Electronics, Communications and Computers. pp. 100-107. Cholula, Puebla, México. February 28th–March 2nd 2011.
14. Hamming, R. W.: Error Detecting and Error Correcting Codes. Bell System Technical Journal. 26(2), 147–160 (1950)
15. S. Thrun, "Robotic Mapping: A Survey," Exploring Artificial Intelligence in the New Millennium, G. Lakemeyer and B. Nevel, eds., Morgan Kaufmann, 2002.
16. Fire-i™ Digital Camera http://www.unibrain.com/products/visionimg/fire_i_dc.htm
17. Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, Second Edition, Springer, New York, ISBN 978-0-387-33254-3, September 2007.
18. K. Deb. Multi-objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester, 2001.
19. Das, I.; Dennis, J.E. 1998: Normal-boundary intersection: a new method for generating the Pareto surface in nonlin- ear multicriteria optimization problems. SIAM J. Optim. 8, pp. 631–657
20. Messac, A., Ismail-Yahaya, A.,Mattson, C.A. The normalized normal constraint method for generating the Pareto frontier. Structural and Multidisciplinary Optimization. 2003-07-01, Springer Berlin / Heidelberg, pp. 86 – 98, Issue: 2 Doi: 10.1007/s00158-002-0276-1
21. Mueller-Gritschneder, Daniel; Graeb, Helmut; Schlichtmann, Ulf (2009). "A Successive Approach to Compute the Bounded Pareto Front of Practical Multiobjective Optimization Problems". SIAM Journal on Optimization 20 (2), pp 915–934.
22. T. Erfani and S. Utyuzhnikov. Directed search domain: a method for even generation of the Pareto frontier in multiobjective optimization. Engineering Optimization, 43(5). pp. 467–484, DOI:10.1080/0305215X.2010.497185, 2010.
23. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation, 6(2):182–197, April 2002.

24. Gutmann, J.-S.; Burgard, W.; Fox, D.; Konolige, K.; , "An experimental comparison of localization methods," Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on , vol.2, no., pp.736-743 vol.2, 13-17 Oct 1998 doi: 10.1109/IROS.1998.727280
25. Gutmann, J.-S.; Fox, D.; , "An experimental comparison of localization methods continued," Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on vol.1, no., pp. 454- 459 vol.1, 2002 doi: 10.1109/IRDS.2002.1041432
26. Kuri-Morales, A., López-Peña, I., "A Novel Method to Determine a Robot's Position Based on Machine Learning", MICAI 2011, IEEE Press, 97-101, Editor(s)): Batyrshin, I., Sidorov, G., ISBN: 9780769546056, 26/11/2011.
27. Kuri-Morales, A., López-Peña, I., "Experimental validation of an evolutionary method to identify a mobile robot's position", MCPR 2012, Springer Berlin / Heidelberg,, 236-245, Editor(s)): Carrasco-Ochoa, J., Martínez-Trinidad, J., Olvera López, J., Boyer, Kim, ISBN: 978-3-642-31148-2, 27/06/2012..
28. D. Lowe. Object recognition from local scale-invariant features. In ICCV, 1999.
29. H. Bay, T. Tuytelaars, L.V. Gool, SURF: speeded up robust features, in: European Conference on Computer Vision, vol. 1, 2006, pp. 404–417.
30. Forsyth, D.A., Ponce, J. Computer Vision: A Modern Approach, Second Edition, Prentice Hall, 2011, ch 21, Recognition By Relations Between Templates, pp. 627-671
31. Kuri-Morales, A., Villegas, C., "A Universal Eclectic Genetic Algorithm for Constrained Optimization", 6th European Congress on Intelligent Techniques and Soft Computing, Aachen Germany, Vol. 1, pp. 518-522, ISBN: 3-89653-500-5, 07/09/1998..
32. Samir W. Mahfound. Niching Methods for Genetic Algorithms. PhD. thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1995.
33. Jeffrey Horn. The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.
34. GNU Image Manipulation Program (GIMP) http://www.gimp.org/
35. Radu Horaud, Humberto Sossa, Polyhedral object recognition by indexing, Pattern Recognition, Volume 28, Issue 12, December 1995, Pages 1855-1870, ISSN 0031-3203, 10.1016/0031-3203(95)00048-8.

36. M. Aldape-Perez, C. Yanez-Marquez, and L. O. Lopez Leyva. 2006. Feature Selection Using a Hybrid Associative Classifier with Masking Techniques. In Proceedings of the Fifth Mexican International Conference on Artificial Intelligence (MICAI '06). IEEE Computer Society, Washington, DC, USA.
37. W. Burger, M.J. Burge. Digital Image Processing: An Algorithmic Introduction using Java. 2008. Springer-Verlag New York. ISBN 978-1-84628-379-6. November 2007
38. Goodrich, MA. A Tutorial on Simple Particle Filters. October 2006.
39. S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, Massachusetts, USA, 2005.
40. G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina, Department of Computer Science, 1995.
41. A. Ochoa, A. Hernández, L. Cruz, J. Ponce, F. Montes, Liang Li, Lenka Janecek. Artificial Societies and Social Simulation Using Ant Colony, Particle Swarm Optimization and Cultural Algorithms. Book Article. New Achievements in Evolutionary Computation, 2010. http://www.intechopen.com/books/export/citation/ProCite/new-achievements-in-evolutionary-computation/artificial-societies-and-social-simulation-using-ant-colony-particle-swarm-optimization-and-cultural. [accessed 2012-10-5]
42. Fox, V.; Hightower, J.; Lin Liao; Schulz, D.; Borriello, G.; "Bayesian filtering for location estimation," *Pervasive Computing, IEEE* , vol.2, no.3, pp. 24- 33, July-Sept. 2003
43. CMUcam3 http://www.cmucam.org/projects/cmucam3/
44. Fit-PC http://www.fit-pc.com/web/fit-pc/
45. Carlos A. Coello Coello. A short tutorial on evolutionary multiobjective optimization. Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, pages 21–40, 2001, Springer-Verlag, Berlin, ISBN: 3-540-41745-1. http://citeseer.ist.psu.edu/coellocoello01short.html
46. T. Weise, Global Optimization Algorithms – Theory and Application. Thomas Weise, 2008, Online available at http://www.it-weise.de/ [accessed 2012-11-6]
47. J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pages 93–100. Lawrence Erlbaum, 1985.

48. Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, Proceedings of the Fifth International Conference on Genetic Algorithms, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.
49. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 3(4):257–271, November 1999.
50. X. Li, J. Zheng, J. Xue. A diversity metric for multi-objective evolutionary algorithms. Proceedings of Int. Conf. on Neur-Computing, 2005:68-73