We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

5

Soft Computing Applications in Robotic Vision Systems

Victor Ayala-Ramirez, Raul E. Sanchez-Yanez, Carlos H. Garcia-Capulin and Francisco J. Montecillo-Puente
Universidad de Guanajuato FIMEE
Mexico

1. Introduction

1.1 Soft Computing

Soft computing is a collection of intelligent techniques working in a complementary way to build robust systems at low cost. Soft computing includes techniques such as neural networks, fuzzy logic, evolutionary computation (including genetic algorithms) and probabilistic reasoning (Wang and Tang, 1997). These techniques are capable of dealing with imprecision, uncertainty, ambiguity, partial truth, machine learning and optimization issues we usually face in real world problems.

Soft computing addresses problem solving tasks in a complementary approach more than in a competitive one. Main advantages of soft computing are: i) its rich knowledge representation (both at signal and pattern level), ii) its flexible knowledge acquisition process (including machine learning and learning from human experts) and iii) its flexible knowledge processing. These advantages let us to build intelligent systems with a high machine intelligence quotient at low cost. Soft computing systems have already been applied in industrial sectors like aerospace, communications systems, robotics and automation and transport systems (Dote and Ovaska, 2001).

1.2 Robotic Vision

Vision, as an exteroceptive sensor, enables autonomous systems to complete complex tasks where environment information is needed. Robotic vision is used in a set of robotic tasks like local and global map building, reactive navigation, topological navigation, object tracking, visual servoing and active sensing among others (de Souza and Kak, 2002). Most of these tasks include a pattern recognition component. In each of these tasks the robot needs to process large amounts of data at a fast rate in order to satisfy real time operation constraints (Barnes and Liu, 2002). Another fact to take into account is the presence of different perturbations in the signals acquired by the robot. At each run, the robot acquires essentially different information even if real life test conditions are very similar. For example, in outdoor environments, sun and clouds can provoke very significant illumination changes in the images, difficulting then to achieve the expected performance of the vision algorithms. To cope with these uncertainties, soft computing techniques have been used because its robustness when facing this kind of scenarios.

Open Access Database www.i-techonline.com

Source: Scene Reconstruction, Pose Estimation and Tracking, Book edited by: Rustam Stolkin,
ISBN 978-3-902613-06-6, pp.530, I-Tech, Vienna, Austria, June 2007

1.3 Soft Computing Applications in Robotics and Vision

Soft computing has been widely used in robotics and vision applications. Fuzzy logic is mainly used in robot control and in the pattern recognition issues arising from robotic tasks. Robot control is particularly addressed by fuzzy logic because we can specify the desired behavior for a system in terms of rules. For example (Saffioti, 1997) presents how to apply fuzzy logic in robot navigation. Another example of fuzzy logic for autonomous vehicle navigation is the FUZZY-NAV project (Pan et al., 1995). Fuzzy pattern recognition uses patterns and models where a given degree of uncertainty, imprecision and inaccuracy is included in the form of associative rules. For example, in human robot interaction, gesture and faces need to be recognized. These kinds of objects are very difficult to characterize in terms of features or relations in a statistical way. That is what makes interesting to use fuzzy systems to incorporate uncertainty handling. (Buschka et al., 2000) have proposed the detection of fuzzy landmarks for map construction. Similar applications have been also proposed by (Bloch and Saffioti, 2002; Gasós and Saffioti, 1999) where map building is addressed.

Neural networks are useful when we have only some examples of the behavior we want to incorporate on a system. In robotics, NAVLAB is a project where neural networks were used to steer an autonomous vehicle (Pomerleau, 1994). Another application for neural networks in robotics concerns denoising techniques for images or even in control applications where, from a set of input-output pairs, neural networks are capable of approximating control surfaces whose behavior we try to emulate.

Genetic algorithms are well suited for optimization problems where we have some cues about desired performance that we can encode in a fitness function. This kind of scenario arises when detecting landmarks or artificial shapes in the robot environment. Another application for genetic algorithms in the robotics domain concerns the path planification issues where the trajectory search space can be verified faster than by brute force approaches or randomized searches.

Detailed discussions on soft computing approaches are given in a number of texts. Neuro-fuzzy algorithms are presented in (Pal and Mitra, 1999), (Mitra and Hayashi, 2000) and (Buckley and Hayashi, 1994). (Herrera and Verdegay, 1996) also include the GA and their relation with other soft computing algorithms. The intelligent systems development is covered in (Ovaska, 2004) and (Abraham et al., 2002).

In this work, we present three robotic vision applications where soft computing techniques are a crucial component for the success of our application. Firstly, we will present a fuzzy color tracking system where color is represented by means of membership functions and fuzzy rules to aggregate color information in the CIELab space. A second system presented here concerns a genetic algorithm based approach for the detection of parametric shapes in images acquired by a mobile robot. A third example includes the hybridization of two soft computing techniques, we present a geno-fuzzy controller used for the servo-control of a pan and tilt camera. For all three methods we present the specific soft computing aspects of their implementation both in simulation platforms and in a robotic platform named XidooBot, a P3AT robot (Fig. 1.).

Fig. 1. XidooBot, a Pioneer P3AT robot used as our experimental test bed.

2. Fuzzy Color Tracking

2.1 Tracking System Components

A high level task for autonomous robot navigation is visual object tracking. This capability is used to avoid collisions or to self-localize by using visual landmarks. Almost all visual tracking systems follow the block diagram shown in Fig. 2. These systems process the visual information acquired by a camera in order to locate a target in the image.

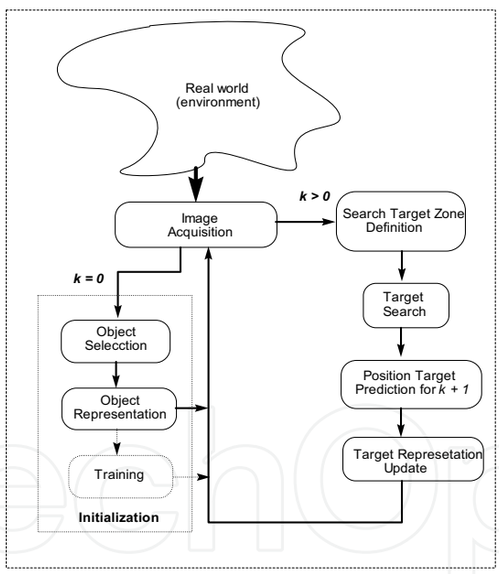


Fig. 2. Block diagram for a general system for target tracking. We use a fuzzy logic-based approach to model the color of the target.

The initialization phase requires using a model to represent the target. The search step requires defining similarity measures to detect it along the visual sequence. Generally, the target representation defines the way in which the comparisons are made. There are several cues to represent the target, among them color is one that has been successfully used on real

time applications (Nummiaro et al., 2003; Argyros and Louriakis, 2004). Robustness of a visual object tracking system relies in the target representation. In this way, we have combined color and fuzzy logic to represent targets (Vertan et al., 2000; Montecillo-Puente et al., 2003), and here we present a fuzzy color tracking system. We use fuzzy logic in order to separate color components and color attributes, like illumination (Keller and Matsakis, 1999). Our main concern is to solve the illumination problems because in real applications illumination changes very often and that appears as if the target was changing its visual appearance.

2.2 Fuzzy Color for Object Representation

Color is one of the features most oftenly used to represent objects. But this feature has some inherent problems, mainly the representation of color in an optimal way. By optimal way we mean to be capable of distinguish between different and similar colors, i.e. red and blue or light red and dark red, respectively. Due to changes in illumination it is possible that a color passes from a light one to dark one, so in real time tracking it is necessary to update the actual representation for color. In this way, the well known problem of saturation also arises due to illumination conditions. These are the topics covered in this section. First we define the fuzzy color and then we describe the procedure to update the fuzzy color.

2.2.1 Fuzzy Color

In order to represent a target by color, we assume that it is monochromatic, that is, it is composed by a set of visually homogeneous pixels, i.e. the target appearance is composed of pixel with intensity values very close in a given color space. The goal is to represent this set of color pixels visually homogeneous, in some way. That is a common situation in real applications due to illumination sources and video cameras noise. In order to represent these color pixels, it is necessary to select a color space. We have selected the *CIELab* color space because in such model visually similar colors have close color coordinates; additionally it possesses a luminance component. The two chromatic components of this space are named, a and b, and the luminance component, L (Braum et al., 1998). Fuzzy color is the assignation of convenient fuzzy sets to each color component. The procedure to define them is as follows:

1. Assume we have a set of visually homogeneous pixels in the RGB color space, p_i with color components p_i^R, p_i^G and p_i^B.
2. Convert all pixels, p_i, to the *CIELab* color space obtaining color components p_i^L, p_i^a and p_i^b.
3. Compute the normalized histogram to each component.
4. Adjust a membership function to each histogram. We may use triangular, trapezoidal or Gaussian ones.
5. The membership functions define the fuzzy sets attached to the set of pixels. That is μ_L, μ_a and μ_b are the membership functions for the components L, a and b

The fuzzy representation of the set of color pixels is given by these membership functions. For deciding if a particular pixel p belongs to the set of pixels (or target) we evaluate the following fuzzy rule

$$\text{if } (R_L \text{ and } R_a \text{ and } R_b) \text{ then } p \text{ is the target} \tag{1}$$

where R_L, R_a and R_b are defined as

$$\begin{aligned} &R_L : L_p \text{ belongs to color component } L \text{ of the target} \\ &R_a : a_p \text{ belongs to color component } a \text{ of the target} \\ &R_b : b_p \text{ belongs to color component } b \text{ of the target} \end{aligned} \tag{2}$$

and L_p, a_p and b_p are the *CIELab* components of the pixel p. Let be C_L, C_a and C_b the membership values of R_L, R_a and R_b for the pixel p, respectively. That is, $C_L = \mu_L(L_p)$, $C_a = \mu_a(a_p)$ and $C_b = \mu_b(b_p)$. Finally, we define truth value for the rule (1), C_p, which expresses how much the pixel p belongs to the set of pixels (or target), as

$$C_p = \min(C_L, C_a, C_b) \tag{3}$$

For the case in which the target is non-monochromatic, e.g. the target is composed by dark red and yellow, we apply the above procedure for defining fuzzy sets to each set of colored pixels. That is a problem because, in general, we do not know how many colors there are and obviously we do not know also the set of colored pixels corresponding to them. We can use in this case some method for determining the number of colors and the set of pixels attached to it, i.e. the Mean Shift procedure (Comaniciu et al., 2000). Once we have the number of colors and the corresponding set of pixels attached to each of them. We apply the above procedure for each color. So we define, for a non-monochromatic target composed by k different colors, its representation as follows:

$$\begin{aligned} &if\ (R_L^1 \ and\ R_a^1 \ and\ R_b^1) \\ &or\ ...\ or\ (R_L^i \ and\ R_a^i \ and\ R_b^i) \\ &or\ ...\ or\ (R_L^k \ and\ R_a^k \ and\ R_b^k) \end{aligned} \tag{4}$$

where R^i_L, R^i_a and R^i_b are defined as (2) for the color i.
So the truth value for a pixel p, with L_p, a_p and b_p *CIELab* components, now is given by

$$C_p = \max(\min(C_L^1, C_a^1, C_b^1), ..., \min(C_L^i, C_a^i, C_b^i), ..., \min(C_L^k, C_a^k, C_b^k)) \tag{5}$$

where the expression $\min(C^i_L, C^i_a, C^i_b)$ represents the truth value of the pixel p for the color i. Then, we define on C_p a fuzzy set with a triangular membership function spanned over [0, 1]. Finally, we form the region of the object by applying an α-cut to all pixels into the search region.

2.2.2 Fuzzy Color Update

In real conditions the target changes its color components, mainly due to illumination variations. Some times it is not a crucial problem because the fuzzy representation of color absorbs them, for example in indoor environments. But some times there are big changes in color components what makes impossible to detect the target, specifically in outdoor environments. Generally, a big change does not occur instantaneously. So we can use these gradual changes to update the membership functions of the color components.

To update the membership functions we illustrate our procedure by using triangular ones as in Fig. 3. These functions are determined by three parameters p_0, p_1, and p_2. These parameters are known after target color modelling. Now we focus only on p_1, assuming that distances from p_1 to p_o and from p_1 to p_2 are constants. Let be R_α the set of pixels which are classified as the target, with $p_i \in R_\alpha$. We could compute the mean of the color components on this set, that is

$$\bar{I}_L = \frac{1}{N_{R_\alpha}} \sum_{p_i \in R_\alpha} p_i^L \tag{6}$$

$$\bar{I}_a = \frac{1}{N_{R_\alpha}} \sum_{p_i \in R_\alpha} p_i^a \tag{7}$$

$$\bar{I}_b = \frac{1}{N_{R_\alpha}} \sum_{p_i \in R_\alpha} p_i^b \tag{8}$$

where $N_{R\alpha}$ is the cardinality of R_α. We change the centers of the memberships by adding

$$\Delta p_1^L = \gamma(p_1^L - \bar{I}_L) \tag{9}$$

$$\Delta p_1^a = \gamma(p_1^a - \bar{I}_a) \tag{10}$$

$$\Delta p_1^b = \gamma(p_1^b - \bar{I}_b) \tag{11}$$

with p_1^L, p_1^a and p_1^b being the membership centers for the L, a and b components, respectively and γ is a smoothing constant.

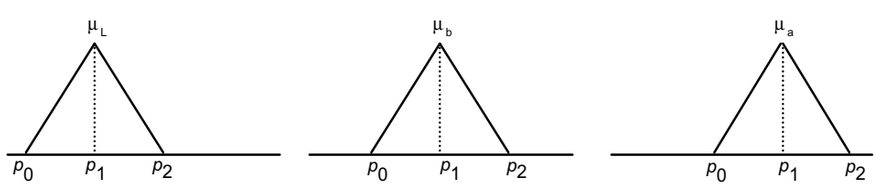


Fig. 3. Triangular membership functions.

2.3 Performance Evaluation

In order to evaluate the fuzzy representation we have performed the following test: In an OpenGL environment simulator, we have set a red ball with a varying light source, then we have moved the ball along a circular path and we have made controlled illumination changes in our environment. In this test we know the ground truth of the center position of the ball in each image. We save an image sequence and the ball position at each image of this sequence. If the error position in each image is small and the number of pixels composing the detected object is almost constant, we can say that our fuzzy color representation is good. In order to show that, we apply a fuzzy color tracking having the test image sequence described previously as input. In Fig. 4, we present an image and the

corresponding detected color blob. In Fig. 5, some frames of the sequence are shown and also the detected blob that satisfies the fuzzy color model. In Fig. 6, we present the position error between the exact ball position in the test image and the position detected by our color tracking system. We observe that maximum error is between 4 pixels. We can consider this error as a low one, that is, our tracking method has a good accuracy.

Fig. 4. A pair of images showing the ball in the simulated environment and the region where the object is detected by our tracking system satisfying the fuzzy color constraints.

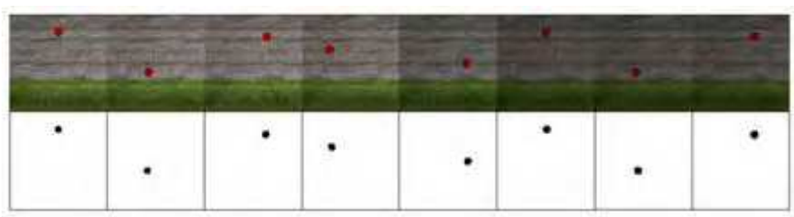


Fig. 5. Test image sequence and the detected region using a fuzzy color representation.

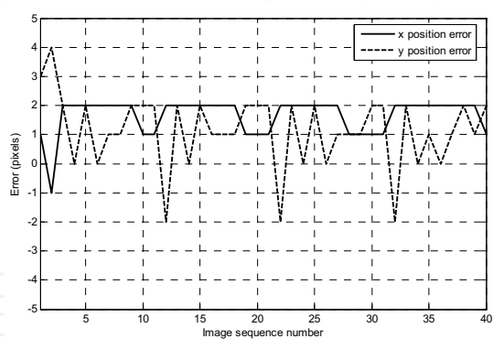


Fig. 6. Graph of the error position between the ground truth image position for the target and the detected position by the fuzzy color tracking system.

In Fig. 7, we show a graph of the number of pixels correctly detected along the frames of the sequence. We can observe that, between frames 30 and 40, there is a transient in the number of pixel detected as belonging to the object. That is caused by the fuzzy color adaptation. An explanation for this behavior is related to the edges of the ball. When the image is lighter, the ball has a good contrast against the background. In the other hand, when light turns dark that effect diminishes. An outdoor sequence taken from the described system

implemented in XidooBot, our robotic platform is shown in Fig. 8. The computing time for a tracking cycle is 0.08 seg, that results in a 12.5 Hz frame rate.

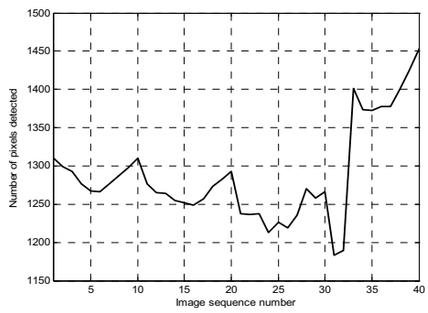


Fig. 7. Graph of number of pixels detected.

Fig. 8. Some frames of an outdoor image sequence where a girl is kicking a yellow soccer ball.

3. Object Recognition using Genetic Algorithms

Genetic algorithms (GA) are pseudo-random search techniques inspired from evolutionary processes. We start from an initial set of feasible solutions for a problem and by mimicking natural evolution, best solution individuals survive and they are the basis of new populations of solutions. This evolutionary cycle is repeated until a solution satisfying problem constraints emerge. GAs are optimization methods; they are useful when we need to search through a large number of feasible solutions. Solutions are evaluated to determine

which ones are the best suited to the problem by using a fitness function that encodes the knowledge we have about the nature of the solution of our problem. In robotics, GA have found application in path planning problems for a robotic arm (Ahuactzin et al., 1993), path planning for mobile robots (Gerke, 1999) and for estimating the position of a mobile robots (Kang et al. 1995). Specifically, genetic algorithms are useful to find a good solution in large search spaces because they can avoid local minima by using genetic operators like mutation, that help the GA to probe in practically every region of the search space. Other GA-based applications in robotics are in visual landmark detection tasks (Hao and Yang, 2003, Mata et al., 2003).

In our work, we present a GA-based circle detector. Our system uses a three edge point circle representation that enables the system to reduce the search space by eliminating unfeasible circle locations in our image. This approach results in a sub-pixellic circle detector that can detect circles in real images even when the circular object has a significative occluded portion. For robotic applications, these circles could be issued from circular landmarks or even being a part of the landmark. After benchmarking our algorithm with synthetic images, we have tested our algorithm on real world images. We present the results of both cases. The latter implementation has been tested on a mobile robot platform XidooBot for circular landmarks detection on the robot environment.

3.1 Circle Detection using GAs

Shape detection is needed in robotic vision tasks like object tracking, visual servoing or landmark recognition. In addition to color and texture, shape is an important cue for modelling objects in scenes of the robot workspace. Object location techniques are solved using two types of methods: i) deterministic methods like Hough transform, e.g. (Yuen et al., 1990), geometric hashing and template or model matching, e.g. (Iivarinen et al., 1997; Jones et al., 1990) and ii) stochastic tecniques, including RANSAC (Fischer and Bolles, 1981), simulated annealing and genetic algorithms (Roth and Levine, 1994).

Using GAs to detect shapes in an image involves mainly the making of design choices for the solution elements in a genetic algorithms framework. We work on images containing one or several circles. The circles are searched through the edge image obtained from an image pre-processing step. A classical Sobel edge detector was used for this purpose. In the following paragraphs we show how to pose the circle detection problem in terms of a genetic algorithm approach.

3.1.1 Individual Representation

Each individual C uses three edge points as chromosomes. Edge points are represented by their relative index in a list V of all the edge points resulting from the edge extraction step. Each individual represents then a feasible circle where their (x_0, y_0, r) parameters are defined as follows:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{12}$$

with:

$$x_0 = \frac{\begin{vmatrix} x_j^2 + y_j^2 - (x_i^2 + y_i^2) & 2(y_j - y_i) \\ x_k^2 + y_k^2 - (x_i^2 + y_i^2) & 2(y_k - y_i) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{13}$$

$$y_0 = \frac{\begin{vmatrix} 2(x_j - x_i) & x_j^2 + y_j^2 - (x_i^2 + y_i^2) \\ 2(x_k - x_i) & x_k^2 + y_k^2 - (x_i^2 + y_i^2) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{14}$$

and

$$r = \sqrt{(x - x_0) + (y - y_0)} \tag{15}$$

We can then represent the shape parameters (for the circle, $[x_0, y_0, r]$) as a transformation T of the edge vector indexes i, j, k.

$$[x_0, y_0, r] = T(i, j, k) \tag{16}$$

This approach enables us to sweep a continuous space for the shape parameters while keeping a binary string for the GA individual. We can then reduce the search space by eliminating unfeasible solutions.

3.1.2 Fitness Evaluation

Each individual has a fitness proportional to the number of actual edge points matching the locus generated by the parameters of the shape (x_0, y_0, r). In our practical implementation, we can not test for every point in the feasible circle so we perform a uniform sampling along the circumference. If we take N_s points, we construct an array of points $S_i = (x_i, y_i)$. Their coordinates are given by:

$$x_i = x_0 + r \cdot \cos\frac{2\pi\ i}{N_s} \tag{17}$$

$$y_i = y_0 + r \cdot \sin\frac{2\pi\ i}{N_s} \tag{18}$$

Fitness function *F(C)* accumulates the number of expected edge points (i.e. the points in the set S) that actually are present in the edge image. That is:

$$F(C) = \frac{\sum_{i=0}^{N_s - 1} E(x_i, y_i)}{N_s} \tag{19}$$

We use also some other factors to favor the context of specific applications for detection, including completeness of the circumference or a given size for the circles.

3.2 Performance Evaluation

We have carried up three tests to evaluate the performance of our approach to circle detection. Firstly, we have generated 10 synthetic grayscale images with only one circle in them and where the ground truth of the circle parameters was known *a priori.* Our method has been run 100 times on each image and the results were recorded in Table 1. We can see that our algorithm detects the circle parameters with sub-pixellic accuracy (lower than 0.194 pixels for the center coordinates and radius length). Our method is robust with respect to translation and scale. Average elapsed time to detect a circle in an image containing exactly one circle is 5 ms.

We have also studied the behavior of the elapsed time for detection of our algorithm with respect to the circle radius. As expected, time seems to grow at a quadratic rate with respect to the radius of the circle. That can be seen in Fig. 9.

Img.	Time	Position	$\bar{x}$	$\bar{y}$	$\bar{r}$	Error	$\lvert e_x \rvert$	$\lvert e_y \rvert$	$\lvert e_r \rvert$
1	0.003	Real	90.00	198.00	10.00	avg	0.046	0.023	0.023
		Detected	89.95	197.98	10.02	max	0.046	0.023	0.023
2	0.001	Real	35.00	28.00	17.00	avg	0.052	0.042	0.052
		Detected	34.95	27.96	17.05	max	0.056	0.056	0.056
3	0.000	Real	167.00	14.00	10.00	avg	0.069	0.023	0.069
		Detected	167.07	13.98	10.07	max	0.103	0.035	0.104
4	0.004	Real	38.00	221.00	28.00	avg	0.003	0.005	0.013
		Detected	38.00	220.99	28.01	max	0.005	0.009	0.018
5	0.025	Real	76.00	44.00	28.00	avg	0.001	0.005	0.013
		Detected	76.00	44.01	28.01	max	0.009	0.022	0.018
6	0.000	Real	184.00	82.00	24.00	avg	0.110	0.077	0.118
		Detected	184.11	81.92	23.88	max	0.144	0.126	0.126
7	0.001	Real	133.00	38.00	10.00	avg	0.014	0.018	0.035
		Detected	133.01	38.02	10.03	max	0.044	0.059	0.050
8	0.001	Real	233.00	108.00	11.00	avg	0.101	0.117	0.171
		Detected	233.10	107.88	11.17	max	0.194	0.160	0.194
9	0.484	Real	120.00	126.00	69.00	avg	0.001	0.001	0.009
		Detected	120.00	126.00	69.01	max	0.029	0.009	0.019
10	0.001	Real	217.00	82.00	20.00	avg	0.068	0.150	0.086
		Detected	216.93	81.85	20.09	Max	0.068	0.150	0.086

Table 1. Circle detection results on synthetic images containing a single circle.

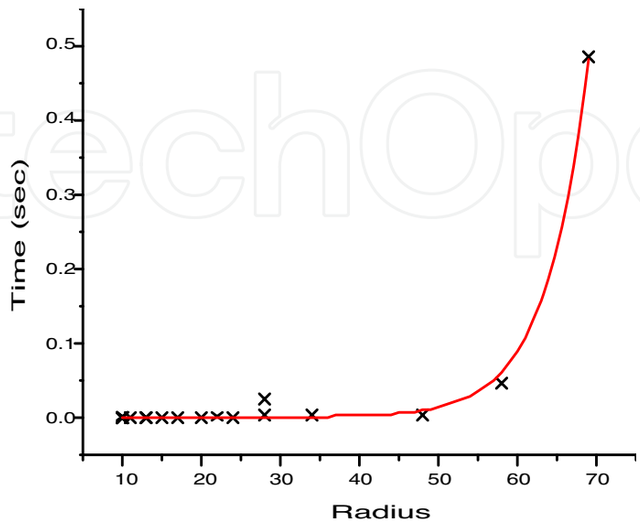


Fig. 9. Time needed to find a single circle in an image against its radius.

3.3 Application to Robotic Vision Tasks

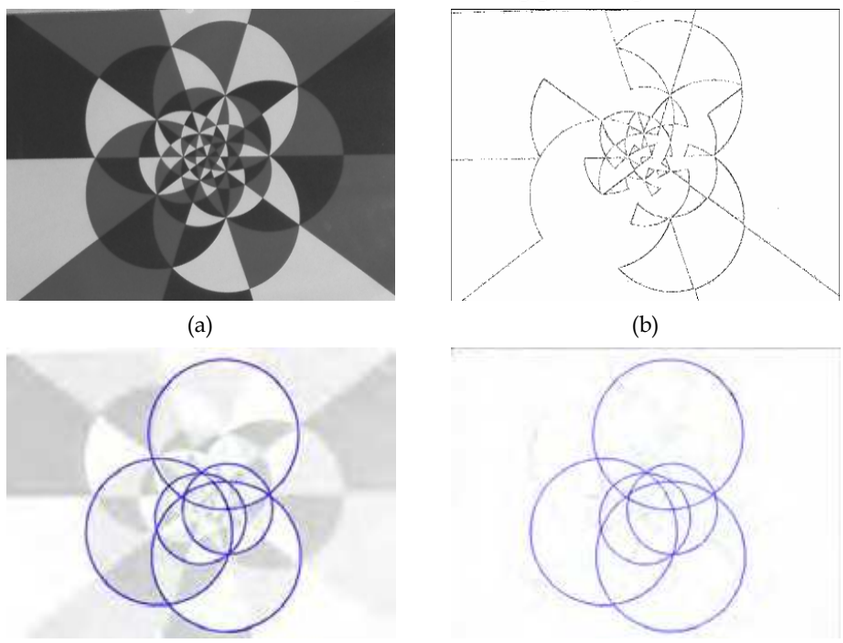
We applied the circle detection method for a robotic task. We have used it for the detection of artificial landmarks containing circular forms. Our interest is to develop a system that uses elliptical shapes (now only covering circular shapes) for using them as landmarks in a topological navigation task. Such a system will be similar to other systems that use quadrangular planar landmarks (essentially posters) for the same task. We consider only quadrangular and circular shapes as structurally salient in a semi-estructured environment. The entire circular landmark recognition process uses two processing loops. The circle detection task achieves a processing rate of about 5 Hz and it interacts with the color tracking system already described that runs at about 13 Hz. The circle detection is launched at the beginning of the navigation task and then it is relaunched only when the robot fails to perceive the detected landmark. This situation arises when the robot is in transit from a topological place to another. Fig. 10 shows some typical images acquired by our mobile robot with circular landmarks on them. An example of an image with multiple circles and the results of the detection process are shown in Fig. 11.

(a) (b) (c) (d)

Fig. 10. Typical scenarios where circular landmarks are useful.

4. Geno-Fuzzy Visual Servoing

We present a genetic algorithm optimization approach for a visual servoing system using a fuzzy controller for an active camera. Visual task for the camera is to center an object with a known model in the field of view of the camera. Our system implements the two-dimensional controller by multiplexing a fuzzy controller for only one motion axis of the camera. We have simulated our system and obtained the controller response to different inputs. We have studied four cases for comparison purposes: a proportional controller, a trial and error tuned fuzzy controller, a fuzzy controller using a genetically-optimized rule base and another one with a database optimization using genetic algorithms.



(a) (b)

(c) (d)

Fig. 11. Results of the circle detection process in a poster with several circles: (a) the original image, b) the edge image of (a), (c) the best 5 circles found by our algorithm overlaid on (a), and (d) the detection results overlaid on the edge image.

We take advantage of the collaborative approach of soft computing for problem solving by combining genetic algorithms and fuzzy logic in a visual servoing controller. In the following paragraphs, we will describe current approaches and applications for hybridization of fuzzy and genetic techniques (so named geno-fuzzy techniques) and some of their applications.

4.1 Visual Servoing

Visual servoing is a maturing approach for controlling robots. It uses a visual task specification instead of using a Cartesian coordinate system previously taught to the robot

(Corke and Hutchinson, 2000). Most robotic systems are instructed interactively to reach some important points for a particular task. Robot task consists in the optimization of the path for all the points not already taught. Using visual servoing, information acquired by the visual sensors of the robot is used to control robot motion in manipulators or mobile robots. Flexibility of robot use is increased in this way, in particular when the robot has to interact with some other objects in its workspace (parts to handle, obstacles to avoid, etc.) A visual servoing system includes techniques from computer vision, robotics and control, and could be considered as a fusion of these disciplines.

According to Corke and Hutchinson, visual servoing systems can be classified in two types: i) image-based visual servoing (IBVS), where error is measured directly on the image and mapped into actuator control signals, and ii) position-based visual servoing (PBVS), where vision techniques are used to reconstruct the 3D environment where the robot evolves and then an actuator control is computed from the error obtained from such an information.

For the PBVS systems, a calibration step needing vision techniques and geometric models is required. In IBVS systems, control computations involve computation of the system Jacobian matrix $\mathbf{J}_v$, a linear transformation that maps the end effector velocity $\dot{\mathbf{r}}$ into the motion of some image feature $\dot{\mathbf{f}}$:

$$\dot{\mathbf{f}} = \mathbf{J}_v(\mathbf{r})\dot{\mathbf{r}} \tag{20}$$

Simplest approach to visual servoing uses the control law that assumes a square and non-singular Jacobian matrix:

$$\mathbf{u} = \mathbf{J}_v^{-1}(\mathbf{r})\dot{\mathbf{f}} \tag{21}$$

A soft computing approach has been the use of fuzzy logic and neural networks to avoid the computation of the Jacobian matrix, as done in (Suh and Kim, 2000) (Stanley et al., 2001), where a fuzzy rule optimization is performed by training a neural network. In this work, we propose to use geno-fuzzy learning techniques to optimize an image-based fuzzy visual servocontroller.

4.2 Geno-Fuzzy Techniques

A highly useful soft computing technique for implementing controllers is fuzzy logic. As pointed out in (Klir and Yuan, 1995), fuzzy logic controllers have advantages over traditional controllers when i) the system to be controlled is complex, ii) the system has been traditionally controlled by human experts, or iii) when human input is needed in the controller model. A fuzzy controller is composed of several basic elements. The fuzzifier translates numerical input into fuzzy values for linguistic input variables. A fuzzy knowledge base is composed of two parts: i) a database, where information about fuzzy membership functions for the input and output linguistic variables used by the system are stored and, ii) a rule base, where rules that determine the controller behavior are stored. This knowledge is used by the fuzzy inference engine to compute a fuzzy output at each instant. Fuzzy output is converted into a numerical output value by means of a defuzziffier.

It is also known that the main drawback of fuzzy controllers is their need of a more complex tuning procedure than for conventional controllers. Building the knowledge base of a fuzzy

system can be done by four methods (McNeill and Thro, 1994; Cordón and Herrera, 1995): i) Synthesis of expert knowledge, ii) trial and error synthesis procedures, used in this work for obtaining an initial or primitive model, iii) synthesis from numerical evidence, and iv) use of machine learning techniques. In this work, a genetic algorithm-based approach is used for tuning the primitive model of the fuzzy visual servocontroller.

Genetic programming, particularly, genetic algorithms, are used to optimize a fitness function by mimicking natural evolution for organisms. Individuals for this evolution are computational representations of potential solutions for the problem to be solved. Each individual is represented as a binary string also known as a computational chromosome. The entire set of individuals examined at a time is called the population.

Geno-fuzzy systems, as is called the combination of genetic algorithms and fuzzy logic controllers, are feasible because in one hand, the behavior of a fuzzy controller is determined by a set of parameters included in the controller knowledge base. Optimal parameter search for the fuzzy controller defines a complex search space. In the other hand, this type of search spaces can be handled efficiently using genetic algorithms. Therefore, fuzzy logic and genetic algorithms could be used to design and optimize fuzzy controllers by formulating optimal parameter search as a genetic algorithm problem.

Pioneering work on geno-fuzzy systems has been done by Karr. He has been the first one to propose fuzzy set parameter tuning for a fuzzy controller (Karr, 1991). (Herrera and Cordón, 1997) have proposed another methodology for genetic algorithm based optimization of a fuzzy controller and its application to the inverted pendulum problem. In robotics, geno-fuzzy techniques have been applied for the control of manipulators as in the work of (Jin, 1998) and a hierarchical fuzzy controller for the navigation of an outdoor mobile robot proposed by (Hagras et al., 2001). Geno-fuzzy systems enable us to develop automatic design methods for fuzzy controllers. This procedure can be applied for automatic design or optimization methods.

4.3. Fuzzy Visual Servoing

We have implemented a visual servoing system using a fuzzy controller to map image features into camera control commands. In a first step, we have synthesized the controller by trial and error and we have compared its performance against a proportional controller for a target recentering task. In order to cope with complexity in the fuzzy controller rule set, we propose a multiplexed fuzzy controller. Secondly, we have optimized the fuzzy controller by using genetic algorithms. We have analyzed two cases: i) Adaptation of the fuzzy rule set, and ii) scaling of a constant gain in inputs and output of the fuzzy controller.

We consider a recentering task for an active camera to follow a given target with a known model moving on a vertical plane. Visual servoing is needed to perform this task. Fig. 12 depicts the implemented system.

In such an IBVS system, the position estimation is computed directly from the last image acquired by the robot. Position error is then computed from the comparison of the reference image I_r and the current image I_c. We also compute an estimate for target velocity. These variables are then fuzzified and input to a fuzzy controller. The controller computes actual commands to be sent to the camera in order to center the target in the camera image. Inputs to our fuzzy controller are positioning error $\mathbf{e}$ and current target velocity $\mathbf{v}$. Both variables are vector quantities with x and y components. Output variable for our controller is the velocity correction V_o. The components of this velocity vector are pan displacement velocity

V_{ox} and tilt velocity V_{oy} for the camera. The structure selected for our controller implementation was a multiplexed one. We have chosen to decouple the x-y controller into an x-controller and a y-controller. Each of them controls only one degree of freedom of the active camera. At each iteration of the control loop, we compute the x-controller output and then the y-controller output. By making this design choice, we have reduced complexity without sacrifying too much accuracy in the controller performance. Structure for our fuzzy controller is shown in Fig. 13.

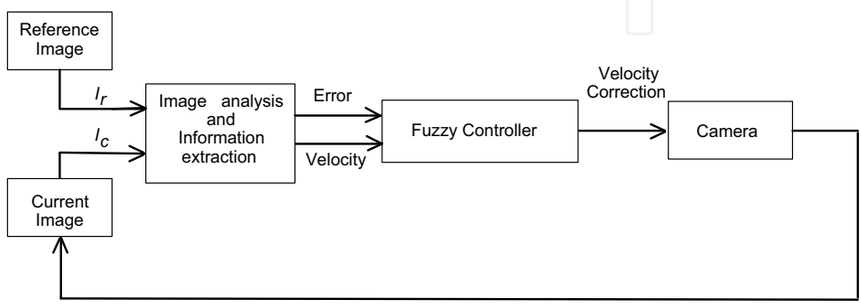


Fig. 12. Fuzzy visual servoing loop.

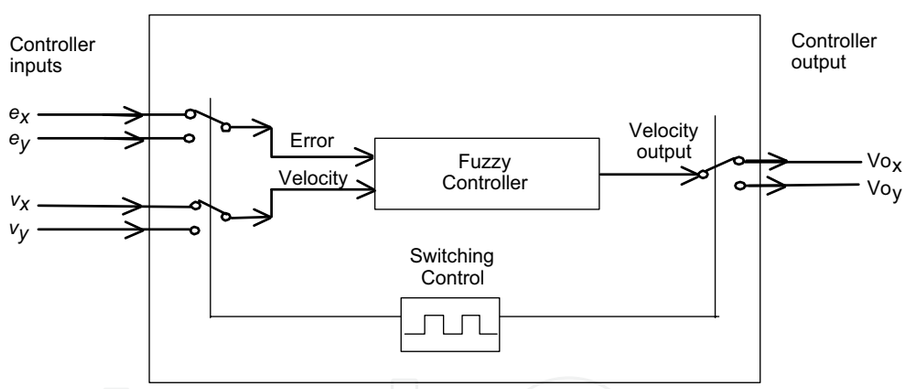


Fig. 13. Multiplexed fuzzy visual servocontroller.

Our system is modelled by a fuzzy logic controller with input universe U, output universe V, and a set of IF-THEN rules that determine a mapping $U \in R^n \rightarrow V \in R$. Every rule of this set has the form:

$$R_i :: \text{IF } (x_1 \text{ is } \mathrm{F}_{1i}) \text{ and } (x_2 \text{ is } \mathrm{F}_{2i}) \text{ THEN } (y \text{ is } \mathrm{G}_i) \tag{22}$$

where F_{ji} and G_i are fuzzy sets of the input and output linguistic variables, respectively (Wang, 1994). We use the singleton fuzzifier and the COA (Center Of Average) method for the defuzzification step. We code the knowledge into a BIOFAM (Binary Input-Output Fuzzy Associative Memory) matrix where the inputs are the error between the center

coordinate of the camera image and the position of the center of the object, and the velocity of the object being tracked for a given direction, and the output is the correction needed on the camera position.

4.4 Hybridization Approaches for Geno-Fuzzy Controllers

Learning strategies for fuzzy controllers by using genetic algorithms are classified using three main approaches (Cordón et al., 2001): i) Michigan approach, where optimization is carried on particular elements of the fuzzy controller specification, ii) Pittsburgh approach, when the system as a whole is optimized, and iii) Iterative Rule Learning (IRL) approach, where the system is synthesized by optimizing independent rules that combine fuzzy sets from a given rule repository, defined either explicitly or implicitly.

Michigan approach encodes each rule in a complete chromosome. Only best rules are kept at each iteration as elite members of the genetic algorithm population. As pointed out by (Ishibuchi et al., 20000), the optimization of the fuzzy rule-based system is indirectly performed by searching for good fuzzy rules. Fuzzy rules have generally a pre-defined structure and a pre-defined set of fuzzy concepts to be used in these rules.

Pittsburgh approach (Ishibuchi et al., 1999) involves encoding a complete or partial rule set into one computational individual. Optimization by using genetic algorithms is equivalent then to find fuzzy rule-based systems with high performance indexes.

In the IRL approach (Cordón et al., 2001), knowledge acquisition is done by acquiring concepts from a repository of fuzzy sets related with input and output linguistic variables. Structure is not pre-defined for these rules and even fuzzy sets associated with the linguistic labels can be chosen in a flexible way from a repository of known fuzzy sets.

A common strategy for all three approaches described above is to adapt the fuzzy controller database. In this technique, fuzzy membership functions (shape and parameters) are individually adapted. In this work, we have tested the database adaptation of a fuzzy controller and the Pittsburgh approach for learning a complete rule base as described below.

In order to optimize our controller, we need to define a fitness function. This fitness function usually consists in the evaluation of the controller performance for an interval of time. Fig. 14 shows a block diagram of the optimization procedure. In our implementation, the fitness function is a measure of the controller performance.

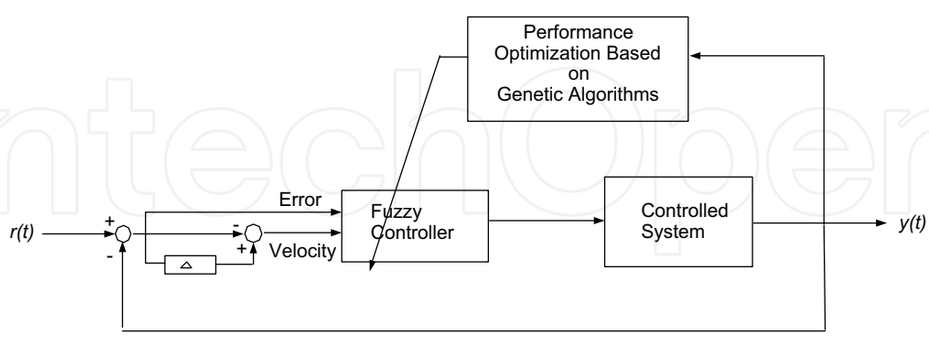


Fig. 14. Optimization of a fuzzy visual servocontroller by using genetic algorithms.

Several performance criteria can be applied in this context. A fitness function that takes into account several parameters at the same time according to the controller characteristics to be improved can be suggested. Some authors (Zhimin et al., 2000) have been concerned with proposing methods for tuning multiple characteristics of the controller. In conformity with the nature of our problem, our controller performance is measured using the difference between the position of the center of the object to be tracked and the center of the image frame, during an interval of time. In order to measure this performance, we use a least square method to compute a figure of merit q as follows:

$$q = \sqrt{\sum_{t=1}^{t_{\max}} \sum_{i=1}^{2} (X_i(t) - X_{d,i}(t))^2} \tag{23}$$

where $X_1(t)$ is the x coordinate of the camera at instant t, $X_2(t)$ is the y coordinate of the camera at same instant and $X_{d,i}(t)$ is the true target position for X_i at instant t.

Another issue to be considered is the length of the chromosome in use. For example, this length is critical for the execution time of the computer implementation for the mutation operation. In this work, we propose two optimization strategies using genetic algorithms: i) Genetic adaptation of the rule base, and ii) genetic adaptation of the database by means of the scaling functions.

Pittsburgh approach for complete rule base adaptation

We specify our fuzzy controller by using a decision table approach. This approach enables the entire rule base to be encoded in a single entity or chromosome. Coding takes place as follows. We start at the BIOFAM position (1, 1) and each row in the matrix is scanned from left to right. Then, the rows are linked together. In order to avoid the generation of a larger chromosome each rule is coded into a genetic alphabet using positive integer numbers in the [0, n-1] range, where n is the number of fuzzy sets of the output variable. The coding process is shown in Fig. 15.

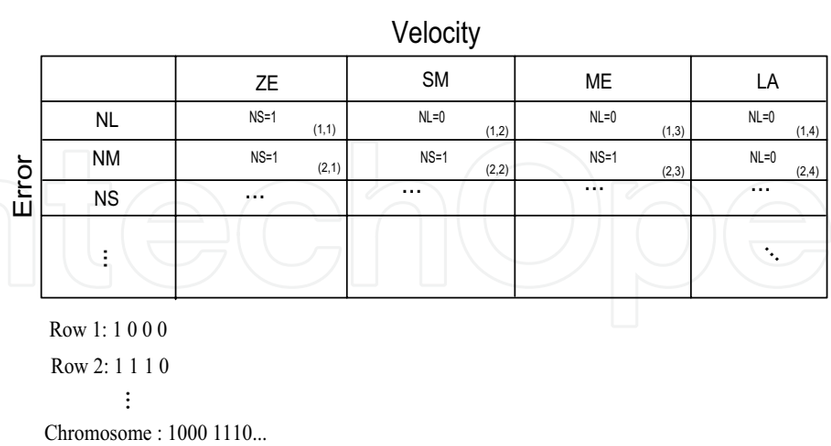


Fig. 15. Rule base encoding scheme.

Genetic database adaptation approach

The second optimization method proposed is the coding of the database using scaling functions. This method was chosen because the controller behavior can be completely modified with just three real numbers. These numbers are a scaling factor for the input variables or, from the control point of view, a gain. An integer representation for each data is used for coding these real numbers. Each number is mapped into an integer value range from -32768 to 32767. The gain range is [0, 2], therefore, the precision for mapping into an integer representation is 3.0517578×10^{-5}. In this form, each real number represents a specific gain for each section of the controller. Thus, each gain value is coded into an integer interval, coded in a binary form. Any real number is then represented by a binary string with 16 bits. There exist 6 possible gain constants to be modified in the controller. Choosing the values to be modified or grouped is an important decision to make. Working with the 6 gains will lead to a larger chromosome and as a consequence, to a more complex search space, and accordingly, the algorithm will take a longer time to find a good solution.

4.5 Tests and Results

We have developed a graphical simulation environment in C language. In this environment, we simulate the *x-y* plane where the target moves and the field of view of the camera. The visual task is to center an object in the field of view of the camera. In our work, this object is a ball of uniform intensity. We have already developed some libraries to manage fuzzy models. We use these libraries to implement the closed loop simulation of our controllers. The fuzzy visual servoing algorithm is executed for all the duration of the time interval to be simulated. Simulation will be stopped also if the tracked object goes beyond the limits of our world simulation.

We have studied four different controllers for the visual servoing task, namely:

Case I: A proportional controller, used only for comparison purposes, implemented as proposed by (Corke, 1996).

Case II: A fuzzy controller tuned by hand (Perez-Garcia et al., 2003) using a self-developed integrated development environment for fuzzy models.

Case III: A genetically-optimized fuzzy controller, using a controller rule base adaptation approach (Pittsburgh approach).

Case IV: A genetically-optimized fuzzy controller, using a controller database adaptation approach.

We simulated different motion patterns for the target on the simulation environment. For each controller, we have applied some motion patterns for the target to be recentered. Results for all different cases will be compared in order to evaluate the controller performance when different design strategies are used.

Case I: Proportional controller.

Some authors, like (Corke, 1996), have studied problems arising in visual servoing. He has pointed out that a proportional controller will exhibit poor performance when used for visual servoing tasks. Main problem is originated by the sampling frequency rate that is too low. For a real time system, target tracking frequency rates are between 2.5 and 12 Hz. When we use a personal computer, tracking execution loop runs at about 4.0 Hz. In order to make the simulation more realistic, we have modelled our camera plant as a first-order system including some time delay T and some inertia a. The camera model used is:

$$C(s) = \frac{1}{Ts + a} \tag{24}$$

According to control theory, K, the proportional gain has to be large enough to compensate system errors. In a servo-controller, there is a conflict because a large value for K can cause the target to get out of the camera view. When dealing with dynamic systems, Corke proposes to use small values for K in order to avoid system instabilities. We have computed the response for a proportional controller with K =0.15, a common value in visual servoing implementations.

Case II: Fuzzy controller tuned by trial and error.

In this section, we will present simulation results when a fuzzy controller is used for the visual servoing task. We have tuned the fuzzy visual servocontroller by a trial and error process. We tested different configuration parameters for all the components of our system and different motion laws for the target. We have reached a final configuration where position error of the target with respect to the center of the acquired image is minimized. Input and output ranges were chosen taking into account specification of a Sony EVI-D30 pan and tilt camera to make a realistic simulation.

We show the final configuration for input and output variables in Fig. 16. We can see that the input variable Error has seven linguistic variables. This fact enables us to achieve a better accuracy in the controller output without having a big number of rules in the BIOFAM for the controller shown in Table 2. Labels are as follows: NL= Negative Large, NM= Negative Medium, NS= Negative Small, ZE= Zero, PS= Positive Small, PM= Positive Medium, PL= Positive Large, SM= Small, ME= Medium and LA= Large.

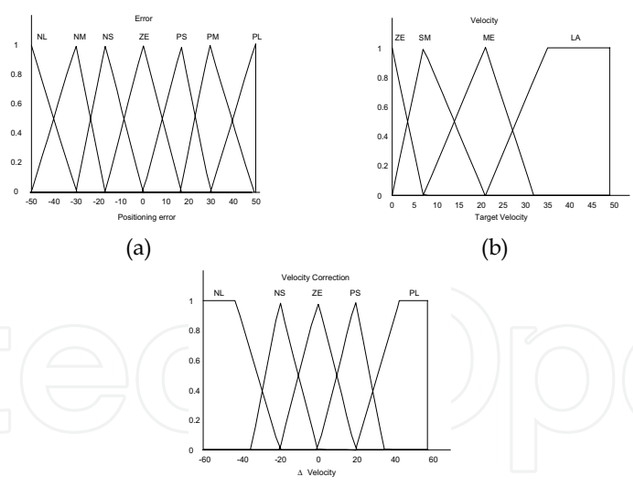


Fig. 16. Membership functions for input and output linguistic variables of the fuzzy controller tuned by trial and error.

Velocity / Error	ZE	SM	ME	LA
NL	NS	NL	NL	NL
NM	NS	NS	NS	NL
NS	NS	NS	NL	NL
ZE	ZE	ZE	ZE	ZE
PS	PS	PS	PL	PL
PM	PS	PS	PS	PL
PL	PS	PL	PL	PL

Table 2. BIOFAM matrix for the fuzzy controller. See text for labels meaning.

We have obtained the controller response to step and ramp inputs. These responses were computed when the controller was using the BIOFAM shown in Table 2 and the input and output variables defined as in Fig. 16. Results are shown in Table 3. We can see that the fuzzy controller outperforms the proportional controller for step input. The fuzzy servocontroller presents an underdamped response and a shorter transient time than the proportional controller. The former presents an overdamped response. When a ramp input signal is used, the proportional controller presents a classical delayed action of the input signal. Otherwise, the fuzzy controller shows an almost zero error. This behavior can be explained by the fact that we have some prediction step because object velocity is an input to the fuzzy controller. This fact enables us to estimate the new positions where the target features could appear.

Case III: Genetically optimized fuzzy controller using Pittsburgh approach

Here we present our results for a fuzzy visual servocontroller optimized by using the Pittsburgh approach over the complete rule set. Figs. 17 (a) and (c) show the error response to step and ramp inputs of the fuzzy visual servocontroller after the rule set had been optimized. Genetic optimization of the rule base slightly improves the response of the fuzzy visual servoing system originally tuned by trial and error. The optimized rule set has a better performance when a similar input to the learned one is fed to the controller but it degrades its performance when a different type of input signal is used.

Case IV: Database adaptation by using a linear scaling function

For this kind of optimization, we have used a gain constant for each input and another for the output of the fuzzy visual controller. As we have only one multiplexed controller for both x and y axes, we have decided to optimize in parallel the same controller. As for the others cases, the optimized servocontroller has been fed with step and ramp inputs. Error responses to these inputs are shown in Figs. 17 (b) and (d), respectively. As we can note there, there are significant improvements in the visual servocontroller response when compared with the other cases. Transient response to step inputs is shorter than for the other cases and the static error for ramp input vanishes, something very difficult to achieve using conventional techniques as pointed out by (Corke, 1996).

Performance evaluation comparison for visual servocontrollers

We have compared the four controllers and the results are summarized in Table 3. We have computed maximum error in pixels for step and ramp inputs. In the case of step input, we have also computed the settling time in seconds and for the ramp, we have computed the steady state static error. We can see that the non optimized fuzzy controller clearly

outperforms the proportional controller when both standard inputs are applied. Comparing the fuzzy controller performance to the rule base-optimized one, we find that maximum error for the last decreases for step input. In the ramp input case, optimized controller maximum error increases slightly but steady state static error is negligible. The fuzzy visual servocontroller optimized by using data-base adaptation outperforms all other controllers in all cases except for the settling time when a step input is applied. In this case, the non-optimized fuzzy controller takes the same time to settle.

Functional performance evaluation on a robotic platform

We have tested our fuzzy visual servocontroller on a real time robotic platform. We aim to center an object in the image acquired by the vision system of a robot. The target was detected by using Hausdorff distance as the similarity metric on the edge image of a sequence. Our system process images up to a frame rate of about 10 frames/sec. Obviously, this rate depends largely on the complexity for the edge model of the target. We have optimized the frame rate for the visual servoing task by implementing a Monte Carlo version of the Hausdorff distance (Perez-Garcia et al., 2006).

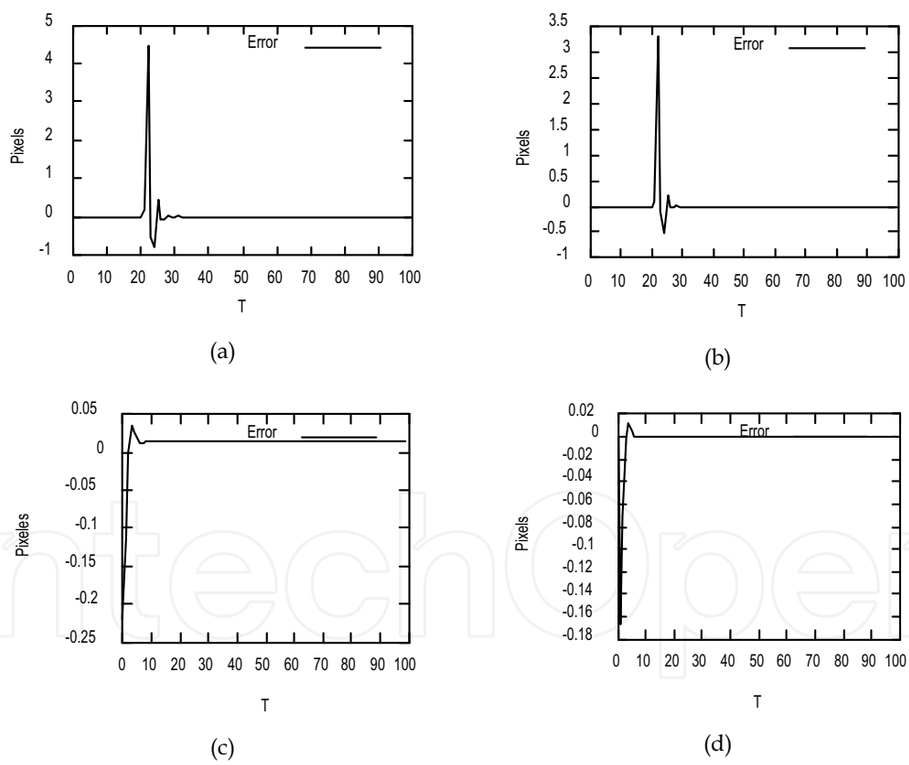


Fig. 17. (a),(b) Step error response for case III and case IV controllers respectively. (c),(d) Ramp error response for case III and case IV controllers respectively.

Controller type	Input Signal			
	Step input		Ramp input	
	Maximum Error (in pixels)	Settling time (in s)	Maximum Error (in pixels)	Steady state static error (in pixels)
Case I: Proportional	16.40	5.25	11.12	11.12
Case II: Fuzzy	12.00	1.50	0.18	0.17
Case III: Geno-fuzzy with rule base adaptation	4.48	2.00	0.22	0.01
Case IV: Geno-fuzzy with data base adaptation	3.31	1.50	0.16	0.00

Table 3. Controller performance comparison using different design methodologies for step and ramp inputs

5. Conclusion

Several conclusions arise from our experience applying soft computing (specifically fuzzy logic and genetic algorithms) to develop robotic vision applications. Fuzzy logic is well suited for problems where uncertainty representation is a critical issue. We already applied fuzzy logic to cope with illumination changes when tracking objects in a visual sequence. Another domain for application of fuzzy logic in robotic vision is to use it where only qualitative experience is available to perform a function. We have applied it on the implementation of a fuzzy visual servocontroller. Genetic algorithms are useful in optimization related tasks in robotic vision. For example, we have used this methodology to optimize the match between a parametric shape (a circle) and an observed set of edge points in an image. Another example of application was the tuning of the fuzzy servocontroller cited above by using a least squares criterion over a time frame.

From a systems perspective, we have used simulation as a tool to benchmark our algorithms before implementation on real platforms. We have also used composition of different modules to integrate more complex systems. This modular system approach has been essential to develop succesful real world applications.

Concerning our applications, we have proposed three methods to use soft computing technologies in robotic vision applications. We address a visual tracking system based on a fuzzy color description of the target, a parametric shape detection task using a genetic algorithm and a geno-fuzzy visual servoing task. The first two methods are developed using a single soft computing technique and the third one uses a hybrid approach by combining genetic algorithms and fuzzy logic as the basis of a robotic vision application. For these applications, we have developed the main aspects of the systems, their implementations and the tests we have carried out to evaluate their performance. Our systems are implemented on board of an experimetal robotics platform, namely a Pioneer P3AT robot. We show experimental results for all of them in real time applications.

Of course, soft computing applications on robotic vision tasks could include other approaches like artificial neural networks not included in the applications presented here. Such methods and a numer of hybrid techniques in soft computing must be taken into account before deciding a particular implementation.

6. Acknowledgements

This work has been partially supported by the CONCYTEG project "Herramientas Mecatrónicas para la Implementación de Entornos Virtuales" and by the UG project "Modelado difuso para la segmentación de imágenes y el reconocimiento de elementos de escenas naturales".

7. References

Abraham, A.; Ruiz-del-Solar; J. & Koppen, M. (Eds.)(2002). *Soft computing systems: Design, management and applications*, IOS Press, Amsterdam.

Ahuactzin, J.M.; Talbi, E.G.; Bessiere, P. & Mazer, E. (1993). Using genetic algorithms for robot motion planning, In: *Geometric Reasoning for Perception and Action*, Springer Verlag, Berlin.

Argyros, A. & Lourakis, M. (2004). Real time tracking of multiple skin-colored objects with a possibly moving camera, *Proc. of the European Conf. on Computer Vision (ECCV'2004)*, Vol. 3, pp. 368.

Barnes, N. & Liu, Z.Q. (2002). *Knowledge-based vision guided robots*, Physica-Verlag, New York.

Bloch, I & Saffioti, A. (2002). On the representation of fuzzy spatial relations in robot maps, *Proc. of the 9th Int. Conf. on Information Processing and the Management of Uncertainty (IPMU)*, pp. 1587-1594.

Braum, G.; Fairchild, M. & Ebner, F. (1998). Color gamut mapping in a Hue-linearized CIELAB color space, *Proc. of the 6th Color Imaging Conf.*, pp. 163-168.

Buckley, J.J. & Hayashi, Y. (1994). Fuzzy neural networks: A survey, *Fuzzy Sets Syst.*, Vol. 66, pp. 1-13, 1994.

Buschka, P.; Saffioti, A. & Wasik, Z. (2000). Fuzzy landmark-based localization for legged robots, *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2000)*, pp. 1205-1210.

Chiang, C. K.; Chung, H. Y. & Lin, J. J. (1997). A self-learning fuzzy logic controller using genetic algorithms with reinforcements, *IEEE Trans. on Syst. Man Cybern.*, Vol. 5, No. 3, pp. 460-467.

Comaniciu, D.; Ramesh, V. & Meer P. (2000). Mean Shift: A robust approach toward feature space analysis, *IEEE Trans. on Pattern Anal. Mach. Intell.*, Vol. 24, No. 5, pp. 603-619.

Cordón, O. & Herrera, F. (1995). A general study in genetic fuzzy systems, In: *Genetic algorithms in engineering and computer science*, John Wiley & Sons, pp 33-57.

Cordón, O.; Herrera, F. Magdalena, L. & Hoffman, F. (2001). *Genetic Fuzzy Systems: evolutionary tuning and learning of fuzzy knowledge bases*, World Scientific, Singapore.

Corke, P. I. (1996). Dynamic Issues in Robot Visual-Servo Systems, In: *Robotics Research*, pp. 488-498.

Corke, P. & Hutchinson, S. (2000). Real time vision, tracking and control, *Proc. of the Int. Conf. Robot. Automat. (ICRA 2000)*, Vol. 1, pp. 622-629.

de Sousa, G. & Kak, A. (2002). Vision for mobile robot navigation: a survey, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 24, No. 2, pp. 237-267.

Dote, Y. & Ovaska, S. J. (2001). Industrial applications of soft computing: A review, *Proc. IEEE*, Vol. 89, pp. 1243-1265.

Fischer, M., Bolles, R., 1981. Random sample consensus: A paradigm to model fitting with applications to image analysis and automated cartography, *Comm. ACM*, Vol. 24, No. 6, pp. 381–395.

Gasós, J. & Saffioti, A. (1999). Integrating fuzzy geometric maps and topological maps for robot navigation, *Proc. of the 3rd Int. Symposium on Soft Computing (SOCO)*, pp. 754-760.

Gerke, M. (1999). Genetic path planning for mobile robots. *Proc. of the 1999 American Control Conf.*, Vol. 4, pp. 2424-2429.

Hao, L. & Yang, S.X. (2003). A behavior-based mobile robot with a visual landmark-recognition system, *IEEE/ASME Trans. On Mechatronics*, Vol. 8, No. 3, pp. 390-400.

Hagras, H.; Callaghan, V. & Colley, M. (2001). Outdoor mobile robot learning and adaptation, *IEEE Robot. Automat. Mag.*, Vol. 8, No. 3, pp. 53-69.

Herrera, F. & Verdegay, J.L. (Eds.) (1996). *Genetic algorithms and soft computing*, Physica-Verlag, New York.

Herrera, F. & Cordón, O. (1997). A three-stage evolutionary process for learning descriptive and approximative fuzzy logic controller knowledge bases from examples, *Int. J. Approximate Reasoning*, Vol. 17, No. 4, pp. 369-407.

Iivarinen, J.; Peura, M.; Sarela,J. & Visa, A. (1997). Comparison of combined shape descriptors for irregular objects, *Proc. 8th British Machine Vision Conf.*, pp. 430–439.

Ishibuchi, H.; Nakashima, T. & Kuroda, T. (1999). A hybrid fuzzy-genetics based machine learning algorithm: Hybridization of Michigan approach and Pittsburg approach, *Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics (SMC)*, Vol. 1, pp. 296-301.

Ishibuchi, H.; Nakashima, T. & Kuroda, T. (2000). A hybrid fuzzy GBML algorithm for designing compact rule-based classification systems, *Proc. of the 9th Int. Conf. Fuzzy Syst. (FUZZ-IEEE 2000)*, Vol. 2, pp. 706-711.

Jin, Y. (1998). Decentralized adaptive fuzzy control of robot manipulators, *IEEE Trans. Syst. Man Cybern. B*, Vol. 28, No. 1, pp. 47-57.

Jones, G.; Princen, J.; Illingworth, J. & Kittler, J. (1990). Robust estimation of shape parameters. *Proc. British Machine Vision Conf.*, pp. 43–48.

Kang, D. ; Hashimoto, H. & Harshima, F. (1995). Position estimation for mobile robot using sensor fusion. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*, Vol. 1, pp. 271-276.

Karr, C. L. (1991). Genetic algorithms for fuzzy controllers, *AI Expert*, Vol. 6, No. 2. pp. 26-33.

Keller, M. & Matsakis, P. (1999). Aspects of high level computer vision using fuzzy sets, *Proc. of IEEE Int. Conf. on Fuzzy Systems*, pp. 847-852.

Klir, G. J. & Yuan B. (1995). *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall PTR, New Jersey.

Mata, M; Armingol, J.M.; de la Escalera, A. & Salichs, M.A. (2003). Using learned visual landmarks for intelligent topological navigation of mobile robots, *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'03)*, Vol. 1, pp. 1324-1329.

McNeill, F. M. & Thro, E. (1994). *Fuzzy logic: a practical approach*, Academic Press, New Jersey.

Mitra, S. & Hayashi, Y. (2000), Neuro-fuzzy rule generation: survey in soft computing framework, *IEEE Trans. Neural Networks*, Vol. 11, No. 3, pp. 748-768.

Montecillo-Puente, F.; Ayala-Ramirez, V.; Perez-Garcia, A. & Sanchez-Yanez R. E. (2003). Fuzzy color tracking for robotic tasks, *Proc. IEEE Int. Conf. on Systems, Man Cybernetics*, Vol. 3, pp. 2769-2773.

Nummiaro, K.; Koller-Meier, E. & Gool, L. V. (2003). Color features for tracking non-rigid objects, *Chinese Journal on Automation*, Vol. 29, No. 3, pp. 345-355.

Ovaska, S.J. (Ed.)(2004). *Computationally intelligent hybrid systems*, Wiley-IEEE Press.

Pal, S.K. & Mitra, S. (1999) *Neuro-fuzzy pattern recognition: Methods in soft computing*, Wiley, New York.

Pan, J.; Pack, D.J.; Kosaka, A. & Kak, A.C. (1995). FUZZY-NAV: A vision-based robot navigation architecture ussing fuzzy inference for uncertainty reasoning, *Proc. IEEE World Congress Neural Networks*, Vol. 2, pp. 602-607.

Pérez-García, A; Ayala-Ramírez, V. & Jaime-Rivas, J. (2003). Fuzzy visual servoing for an active camera, *Proc. 21st Int. Conf. Applied Informatics (AI'2003)*, pp. 292-296.

Pérez-García, A; Ayala-Ramírez, V.; Sanchez-Yanez, R.E. & Avina-Cervantes, J.G., Monte Carlo evaluation of the Hausdorff distance for shape matching, *Lecture Notes in Computer Science*, Vol. 4225, pp. 686-695.

Pomerleau, D.A. (1994). Reliability estimation for neural network based autonomous driving, *Robotics and Autonomous Systems*, Vol. 12, pp. 113-119.

Roth, G. & Levine, M.D. (1994). Geometric primitive extraction using a genetic algorithm., *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 16, No. 9, pp. 901–905.

Saffioti, A. (1997). The uses of fuzzy logic in autonomous robot navigation, *Soft Computing*, Vol. 1, No. 4, pp. 180-197.

Stanley, K.; Wu, J. & Gruver, G. (2001). A hybrid neural network based visual servoing robotic system, *Proc. Int. Conf. IFSA-NAFIPS*.

Suh, I. & Kim, T. (2000). A visual servoing algorithm using fuzzy logic and fuzzy neural networks, *Mechatronics Journal*, Vol. 10, pp. 1-18.

Tan, K. C.; Lee, T. H. & Khor, E. F. (2002). Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons, *Artif. Intell. Rev.*, Vol. 17, No. 4, pp. 251-290.

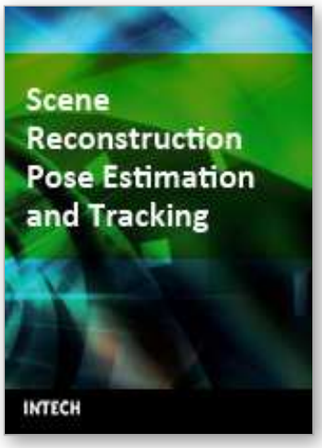
Vertan, C.; Boujemma, N. & Buzuloiu, V. (2000). A fuzzy color credibility approach to color image filtering, *Proc. IEEE Int. Conf. on Image Processing*, Vol. 2, pp. 808-811.

Yuen, H.; Princen, J.; Illingworth, J. & Kittler, J. (1990). Comparative study of Hough transform methods for circle finding, *Image Vision Comput.,* Vol. 8, No. 1, pp. 71–77.

Wang, L. X. (1994). *Adaptive fuzzy systems and control: design and stability analysis*, Prentice Hall, Eglewoog Cliffs, NJ.

Wang, P.H. & Tan, S. (1997). Soft computing and fuzzy logic, *Soft Computing*, Vol. 1, No. 1, pp. 35-41.

Zhimin, Y.; WangXu & Xianyi, Z. (2000). Multi-criteria Control Systems Parameters Optimization Based on Genetic Algorithm, *Proc. 3rd World Congress on Intelligent Control and Automation*, pp. 651-655.



Scene Reconstruction Pose Estimation and Tracking

Edited by Rustam Stolkin

ISBN 978-3-902613-06-6

Hard cover, 530 pages

Publisher I-Tech Education and Publishing

Published online 01, June, 2007

Published in print edition June, 2007

This book reports recent advances in the use of pattern recognition techniques for computer and robot vision. The sciences of pattern recognition and computational vision have been inextricably intertwined since their early days, some four decades ago with the emergence of fast digital computing. All computer vision techniques could be regarded as a form of pattern recognition, in the broadest sense of the term. Conversely, if one looks through the contents of a typical international pattern recognition conference proceedings, it appears that the large majority (perhaps 70-80%) of all pattern recognition papers are concerned with the analysis of images. In particular, these sciences overlap in areas of low level vision such as segmentation, edge detection and other kinds of feature extraction and region identification, which are the focus of this book.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Victor Ayala-Ramirez, Raul E. Sanchez-Yanez, Carlos H. Garcia-Capulin and Francisco J. Montecillo-Puente (2007). Soft Computing Applications in Robotic Vision Systems, Scene Reconstruction Pose Estimation and Tracking, Rustam Stolkin (Ed.), ISBN: 978-3-902613-06-6, InTech, Available from: http://www.intechopen.com/books/scene_reconstruction_pose_estimation_and_tracking/soft_computing_applications_in_robotic_vision_systems

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2007 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.