20

Object Recognition for Obstacles-free Trajectories Applied to Navigation Control

W. Medina-Meléndez, L. Fermín, J. Cappelletto, P. Estévez,
G. Fernández-López and J. C. Grieco
Universidad Simón Bolívar, Grupo de Mecatrónica, Valle de Sartenejas, Caracas Venezuela

1. Introduction

In this chapter applications of image and video processing to navigation of mobile robots are presented. During the last years some impressive real time applications have been showed to the world, such as the NASA missions to explore the surface of Mars with autonomous vehicles; in those missions, video and image processing played an important role to rule the vehicle.
Algorithms based on the processing of video or images provided by CCD sensors or video cameras have been used in the solution of the navigation problem of autonomous vehicles. In one of those approaches, a velocity field is designed in order to guide the orientation and motion of the autonomous vehicle. A particular approach to the solution of the navigation problem of an autonomous vehicle is presented here. In the first section of this introduction a state of the art review is presented, after it, the proposed algorithm is summarized; the following sections present the procedure. Finally, some experimental results are shown at the end of the chapter.

1.1 Review of Navigation of Autonomous Robots using Vision Techniques.

In the area of autonomous navigation using vision techniques, the previous works (Santos-Victor & Sandini,1997), and (Nasisi & Carelli, 2003) are corner stones. In the first mentioned study, robot control techniques are explored, using both cameras on board of the robots and external cameras. In that work it is shown that it is possible to accomplish effective control actions without doing a complete processing of the image captured or without the calibration of the camera. In the job of Nasisi & Carelli, a set of equations needed to establish relationships among a bidimensional image captured by a video camera and its corresponding tri-dimensional image is obtained, an equation set that is important when a single camera is being used. The jobs of S. Skaar et al., (who participated in the 2003 Mars Exploration Rover experiment of NASA), over the concept of Camera Space Manipulation (CSM) (Skaar et al., 1992) and the concept of Mobile Camera Space Manipulation (MCSM) (Seelinger et al., 2002), must be cited. The MCSM method consists of the estimation of the relationship among the characteristics position of the manipulator and its corresponding points in the image spaces of the two cameras mounted over the robot; the CSM concept is quite similar but with more restrictions. Both methods, the CSM and the MCSM require not

Source: Vision Systems: Applications, ISBN 978-3-902613-01-1
Edited by: Goro Obinata and Ashish Dutta, pp. 608, I-Tech, Vienna, Austria, June 2007

only the parameters of the cameras to be completely known, but also the kinematics model of the manipulator, even if they don't require the complete calibration of the camera and the manipulator. These methods require a set of cameras, while the methodology proposed in (Santos-Victor & Sandini,1997) and (Nasisi & Carelli, 2003) involves only one.

One vital characteristic of every navigation strategy is the way that the decisions are taken on it when the sensory system indicates the presence of obstacles on the robot trajectory. Different obstacle avoidance strategies have been presented; among those strategies the use of electrostatic fields - where the robot is modeled as a positive electric charge, as also the obstacles, and the objective of the trajectory is modeled as a negative charge - must be mentioned (Dudek & Jenkin, 2000) and (Khatib, 1985); the Coulomb law is applied to determine the trajectory of the mobile robot.

In 1995, Li and Horowitz presented the concept of velocity fields (Li & Horowitz, 1995); in such a case, a vector velocity is defined over the trajectory of the robot for each possible position coding a specific task. Using control schemes with the velocity field as the reference for the system, has allowed approaches to the problem employing different control schemes instead of the classic following trajectory problem. In this approach, the improvement of coordination and synchronization among the degrees of freedom of the robot are much more important than the time to execute a given task. In the Figure 1 an example of this strategy is presented.

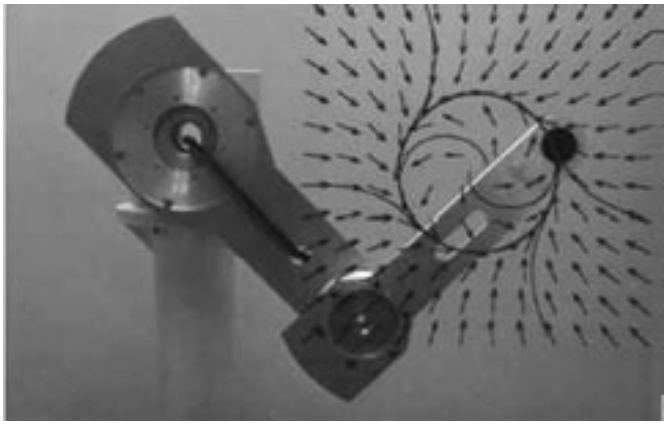

Figure 1. Velocity Field for a manipulator in the plane

With the introduction of the concept of velocity fields, also emerged the strategy of Passive Velocity Field Control (PVFC). In this control strategy the tasks are coded using the velocity and energy that the system could transfer to the environment and that it is limited by a constant (Li & Horowitz, 1999). Li and Horowitz presented the properties of the PVFC strategy doing special emphasis on the contour following problem in (Li & Horowitz, 2001a) and (Li & Horowitz, 2001b).

In many research works related with velocity fields, the stability of the control schemes is pursued; Cervantes et al. (Cervantes et al., 2002) proposed a Proportional – Integral Controller based on compensation errors techniques where it is not necessary to have a deep knowledge of the robot dynamics, obtaining semi global asymptotically stable conditions on the following errors of the velocity fields. Moreno and Kelly, in (Moreno & Kelly, 2003a),

(Moreno & Kelly, 2003b) and (Moreno & Kelly, 2003c), proposed different control schemes using velocity fields focusing mainly on the asymptotically stable conditions.
Other investigations related to velocity fields have focused in other areas, such as adaptive control (Li, 1999). In that work, Li proposed an adaptive control scheme using velocity fields as references for robots with unknown inertia parameters. Also in (Dixon et al., 2005) an adaptive derivative control is presented in order to follow a velocity field. In almost all the previous cases, the works had as an objective the design of control schemes taking a velocity field as a time-invariant reference, obtained after a theoretical or analytical procedure. Only few works in the literature have considered the case when the field is time-dependent.

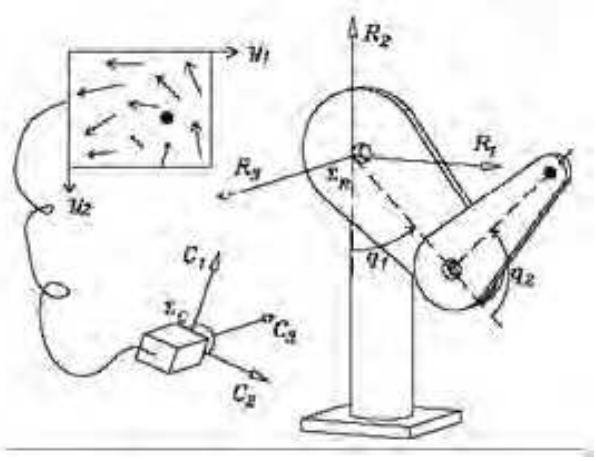

Figure 2. Robot system employed in (Kelly et al., 2004a)

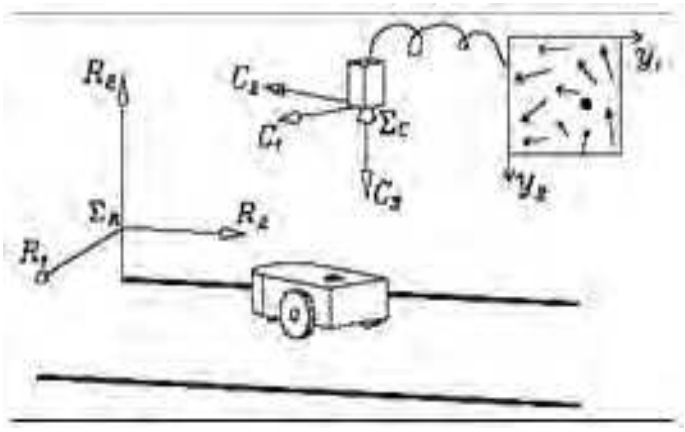

Figure 3. Robot system employed in (Kelly et al., 2004b)

Yamakita et al., in (Yamakita & Suh, 2000) and (Yamakita & Suh, 2001), applied PVFC to cooperative robots, obtaining field velocity generating algorithms for tasks where it is necessary to follow certain orders of a master robot. For this work the authors created an

augmented dynamic system using an inertia flywheel, in order to avoid the problems of control following by velocity fields. Recently, Kelly and collaborators in (Kelly et al., 2004a) and (Kelly et al., 2004b), used a camera as a main sensor in a control scheme by velocity fields in order to control a mobile robot and a robot manipulator. Particularly in (Kelly et al., 2004a) is presented a controller by velocity fields for a manipulator of two degrees of freedom that incorporates an external camera to capture the manipulator movements and the work environment. On the captured image a theoretical velocity field that the manipulator is capable to follow is pre-defined. In the Figure 2 an example for this case is presented.
In (Kelly et al., 2004b), the control is applied to a wheeled mobile robot where the video camera is located over the robot. Like in the previous case, the camera is located in such a way that covers the environment and the robot itself. In the Figure 3 can be visualized the system under study.

1.2. The proposed approach

During last years, as it was mentioned, the use of artificial vision in robot tasks has been increasing. In many applications, such as rescue jobs, the use of vision to generate velocity fields is under study. The dynamic modification and generation of the velocity field itself, changing it with the environment modifications, is a new research area. If timing is not an issue in the control objectives, then velocity field control, where the speed can be adjusted as desired, is a potentially important technique. For instance, if the objective is to control a vehicle's trajectory at 80 Km/h, the velocity field can be adjusted to that speed and modified in order to avoid obstacles detected by a camera (either on board or external), while keeping close to the original trajectory in order to reach the final objective. This approach could be of crucial importance in rescue tasks where a flying machine could be "understanding" the environment, and changing the velocity field references for another robot on the ground. Another potential application is in automatic warehouses where changing the velocity field references could be assigned different and changing tasks to the manipulator. In the future, the dynamic velocity field generator that is presented in this work will be modified to in order to allow the generation of a 3-dimensional velocity field. Also, the algorithm is going to be used to coordinate the tasks of cooperative robots for Futbot, submarine vehicles coordination, cooperative multi-robot observation, etc.
In order to perform the tests, an experimental setup has been integrated. This experimental setup consists of:

1. A Hemisson differential mobile robot, created by the Swiss company "K-Team Corporation".
2. 2.4 GHz wireless video camera, model XC10A and its wireless receptor VR31A. This is the only sensor employed in the experiments in order to detect the robot and the obstacles. The camera has a resolution of 640 x 480 pixels and offers up to 30 frames per second (fps).
3. Image acquisition card model NI-PXI-1407 using the Standard RS-170 for monochromatic video signals. All the characteristics can be seen in Table 1.
4. A PXI module from National Instrument is employed to process the data. The module is the NI-PXI-1000B.

Description	Standard monochromatic
Bus	PXI/CompactPCI
Video Inputs	1
Spatial Resolution	640 × 480 RS-170
	768 × 576 CCIR
Píxel depth	8 bits
Video input Standard	RS-170, CCIR
Digital I/O	1

Table 1. Characteristics of the NI-PXI-1407

On the NI-PXI-1000B run all the LabView DLL's and Matlab applications.
The working environment consists of an area of approximately 6 square meters where the robot navigates. The wireless video camera was located 2.5 meters over this area. In Figure 4 is shown a sketch of the experimental arrangement.

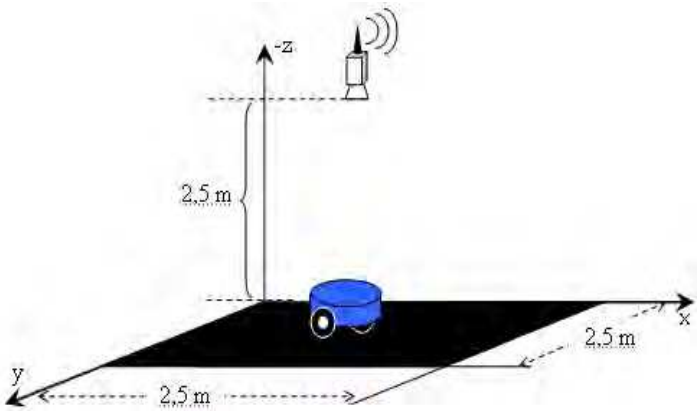

Figure 4. Environment employed for the experiments

The problem for the generation of a dynamic velocity field was divided in three different steps:

1. A generator of the initial velocity field.
2. The processing by the artificial vision system.
3. The generator of the evading velocity field.

1.3. Description of the generated system

Initially the user must define the task without taking into account the obstacles that might appear in the trajectory of the mobile robot. In order to indicate the trajectory a generator was developed for the initial velocity field. This generator is presented later in this chapter. In order to avoid the obstacles presented during the trajectory of the mobile robot, it is necessary to identify their existence, determining its location, size and orientation. In order to solve this problem the artificial vision system was. The information supplied by the artificial vision system is employed to modify the initial velocity field using the generator of the evading field. Once the desired task is well known and also the position and orientation of the obstacles the system creates the local velocity fields that surrounds each obstacle; the

final navigation field is obtained adding the initial field with the evading fields pondering each one properly.
In first place and before testing the system in the experimental setup, the results were validated using the Easy Java Simulations platform. The results are presented also in this chapter.

2. Vision System Implementation

This section describes two implementations of the vision system for the robot and obstacle identification process. In both cases the robot detection was made by using a pattern matching technique, and for the obstacle detection were employed two different approaches.
The first implementation is based on the hypothesis that any object present in the scene can be described as any of the following regular shapes: circle, square and rectangle. In this case it was utilized a classification strategy for its identification.
For the other system, the obstacle detection was made through a particle analysis, in order to cope with those problems arisen by using the previous approach in images containing obstacle with irregular shapes.

2.1. Robot identification by pattern matching

The pattern matching process consists of the fast detection into the image of those regions that have a high concordance with a previously known reference pattern. This reference pattern contains information related with edge and region pixels, thus allowing the deletion of redundant information contained into a regular shape.

2.1.1. Normalized Cross-Correlation

The correlation process based on the Euclidian distance, is described by the following equation:

$$d^2_{f,t}(u,v)=\sum_{x=0}^{L-1}\sum_{y=0}^{M=1}\left[f(x,y)-t(x-u,y-v)\right]^2 \tag{1}$$

Where f is the input image of size LxM, and the sum is done over (x, y) in the window containing the sub-image t localized at (u, v), of size JxK. By expanding d^2 we obtain:

$$d^2_{f,t}(u,v)=\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}\left[f^2(x,y)-2\cdot f(x,y)\cdot t(x-u,y-v)+t^2(x-u,y-v)\right] \tag{2}$$

Where $\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}t^2(x-u,y-v)$ is a constant. If $\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}f^2(x,y)$ is almost constant, then the resulting term for the cross correlation is the measure of similitude or concordance between the image and the pattern. For this case u = 0, 1,…, M-1, and v = 0, 1,…, L-1, and J=M and K=L.

$$C(u,v)=\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}f(x,y)\cdot t(x-u,y-v) \tag{3}$$

Figure 5 shows the correlation process assuming that the origin for *f* is placed in the upper left corner. Therefore, the correlation process consists on moving the template *t* through the area of the input image and to calculate the value for C. By this method, the maximum value for C points the position where is the highest similitude between *t* and *f*.

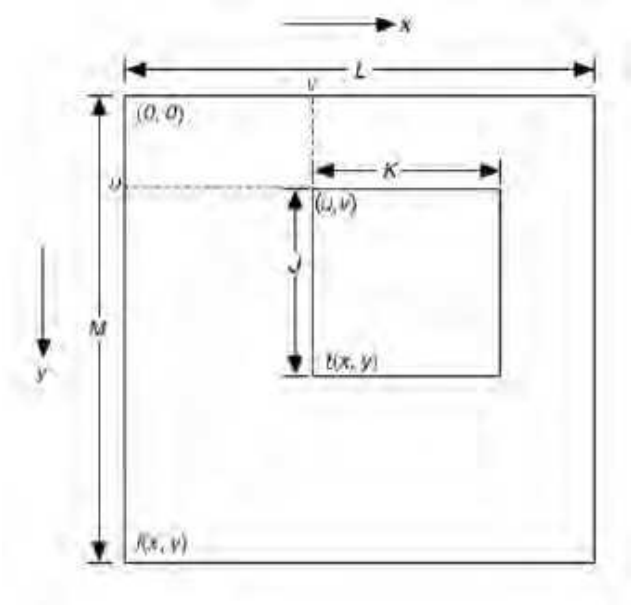

Figure 5. Window movement during the correlation process

Due to the dependence of the correlation values to intensity variation in the image, is required to compute the normalized version for the correlation given by (4):

$$R(u,v)=\frac{\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}\left(f(x,y)-\bar{f}_{u,v}\right)\cdot\left(t(x-u,y-v)-\bar{t}\right)}{\left[\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}\left(f(x,y)-\bar{f}_{u,v}\right)^2\cdot\sum_{x=0}^{L-1}\sum_{y=0}^{M-1}\left(t(x-u,y-v)-\bar{t}\right)^2\right]^{\frac{1}{2}}} \quad (4)$$

Where $\bar{t}$ is the average intensity value for the pixels in the image and $\bar{f}_{u,v}$ is the average of *f(x,y)* in the template. The obtained value for *R* is normalized between -1 and 1, and is independent to intensity of *f* and *t* images.

2.1.2. Pattern recognition process.

The detection pattern process is divided in to main subsystems: learning and recognition itself.

The learning phase analyzes the template image in order to extract features that can improve recognition process compared to standard approach.

The pattern learning algorithm can be described as follows (National, 2004):

- Pseudo-random Image sub-sampling: By this method, it can be obtained more uniformity on sampling distribution through the image, without using a predefined grid. With a uniform grid information like the presence of horizontal and vertical

borders could be lost. In the other hand, completely random sampling can produce large areas with poor sampling or over-sampled areas.

- Stability analysis: The pixels sampled by pseudo-random algorithm are analyzed in order to verify stability (uniformity) on their vicinities. Based on this information, each pixel is classified according to its uniform vicinity size (i.e. 3x3, 5x5). By doing so, it is reduced the number of comparisons required for further matching process.
- Features identification: it is an edge detection process, which output is used to detailed adjustment for the placement of the recognized pattern.
- Rotation invariance analysis: is based on the identification of a circular intensity profile on the pattern image, that will be used later to localize rotated versions of the same pattern because those versions will have the same profile with an displacement factor proportional to its original rotation.

The learning phase is computationally complex. It can take several seconds to be completed. However, this process is done only once and its result can be stored for further applications. The pattern recognition algorithm consists on two processes:

- The utilization of a circular intensity profile obtained in the learning phase, in order to localize rotated and displaced versions of that profile through the image.
- The utilization of those pixels obtained in the pseudo-random sub-sampling, that are employed in a correlation process between the candidates previously identified, giving it a score that is used to determine if that candidate can be classified as a pattern match.

2.1.3. Pattern recognition subsystem – Implementation

As it was stated in the previous section, the pattern recognition subsystem is separated in to stages: learning and matching; each of them was implemented as an VI in LabVIEW v7.1®. The corresponding VI for the matching stage also contains the implementation for the obstacle detection. However, in this section it will be described only the pattern recognition system.

Pattern Learning

The learning stage starts by loading an image that contains the pattern to be recognized. For this implementation, the desired recognition target is a mobile robot model Hemisson. Then it is performed the pattern learning and the resulting information is stored in a PNG image file.

Figure 6 shows the block diagram for the pattern learning process, described with more details in further sections.

- Initialization: A blank image with 8 bits per pixel is created.
- Image file loading: Over the previously created image it is loaded the file image containing the desired pattern.
- Image conversion: The image is converted into a grayscale one, to be processed by the learning module.
- Learning module configuration: it is configured so it generates information for the pattern recognition rotation invariant.
- Storing: The information related to the pattern learning in a new PNG image file.

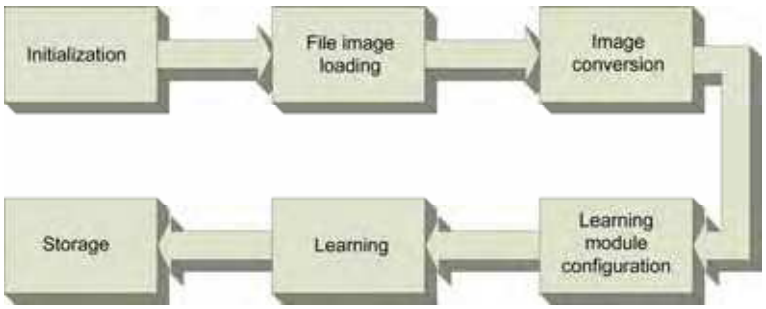

Figure 6. Block diagram of Pattern Learning Process

Figure 7 shows the VI created for this learning subsystem.

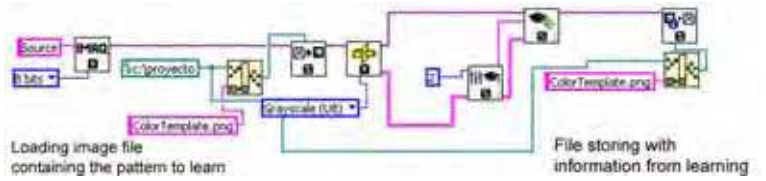

Figure 7. VI for the learning stage

This VI has neither explicit input nor output parameters, because it relays upon the loading of an image file for the learning process. After de VI construction, it is created a Dynamic Link Library (DLL) using the Math Interface Toolkit from LabVIEW®. This provides a DLL file with the subroutines so they can be called from Matlab®.

Pattern recognition

The recognition process consists in taking a snapshot from the workspace, loading the file containing the learning pattern previously obtained, and searching such pattern on the captured image. Later, it is determined the position and orientation of the pattern. The recognition process is described in Figure

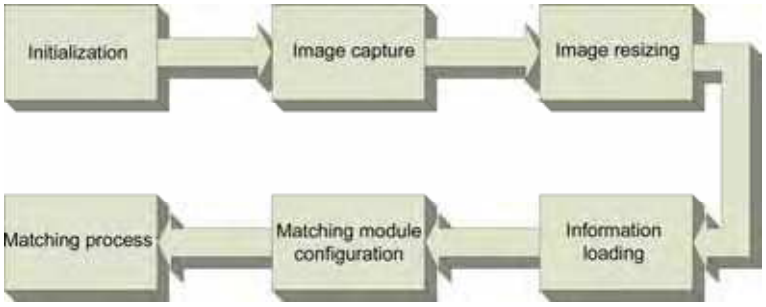

Figure 8. Block Diagram of the pattern matching process

- Initialization: two blank images with 8 bpp are generated. One image corresponds to the video capture and the other one is for the patter image loading. It is also started the video acquisition through the NI IMAQ 1407 module.

- Image Capture: It is acquired a real image from the workspace, the resulting image in grayscale.
- Image resizing: In order to improve position calculation and detection speed, the captured image is resized from 640 x 480 to 480 x 480 by cropping it. This image size corresponds to a physical working area of 4 m².
- Information loading: It is loaded the information related to the learning pattern contained in the PNG image file stored in the previous system.
- Matching module configuration so it performs the invariant rotation pattern search.
- Matching process: This process will be done with the acquired image and the image loaded with the learning information. In case of a successful detection of the desired pattern, it will be obtained the position of the coincidence in the working space and its rotation, as shown in Figure 4.7.

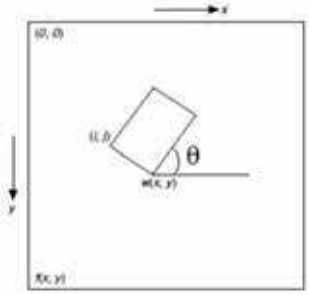

Figure 9. Representation of position and rotation convention

As it was done for the VI of the learning stage, it was created a DLL file from the pattern recognition VI, which we will identify as "detection engine". The matching process has no input parameters; it only loads the image containing the information in the learning process. The output parameters are the position and orientation for the detected pattern.

2.2. Obstacle Detection

2.2.1. Classification technique.

The obstacle detection is based on the classification of binary particles, in which an unknown particle is identified by comparison of their most relevant features against a set of characteristics that conceptually describes previously known classes samples.

The obstacle detection based on classification techniques has two main phases: the learning process, which was implemented utilizing the NI Vision Classification Training tool, and the classification process itself.

Learning Process

The training process consists in the recollection of a set of samples images that contains possible obstacles that can be found and detected by the camera. From these samples is obtained a set of features, known as characteristics vector that describe unequivocally each class of known sample. For example, it could be the object circularity or elongation. Once determined the characteristics vector and collected the samples, each object class is labeled.

Classification Process

The classification process involves the pre-processing of the input image, the features extraction and classification, as is shown in Figure 10.

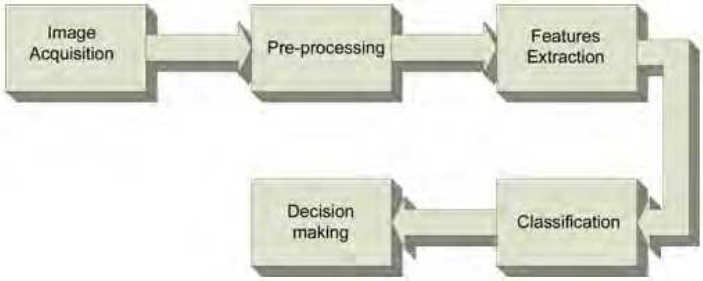

Figure 10. Steps for the Classification process

During the pre-processing stage, the image is prepared for the classification by adjusting the brightness to a level where the classifier can detect the object. This value depends of scene illumination at detection process. Other pre-processing operations are done, like thresholding and binary morphology: the input image in grayscale is converted into a binary image, and later an eroding process is applied to delete irrelevant particles.

Through feature extraction, the information contained in the image is reduced because only distinctive characteristics are retained to distinguish each class. Also, those features are scaling, rotation o symmetrical transformations invariant, while they still allows to do correct objects classification.

The final step is to classify the captured objects in the images by using the extracted features. The classificatory algorithm employed was Nearest Neighbor (*NN*), because it computational simplicity and effectiveness in low features situations. It was employed Manhattan metric.

Under this classification algorithm, the distance between a given feature input set X and an unknown class C_j is defined as the distance to the closest sample that represents such class:

$$D\left(X, C_j\right) = \min_i d\left(X, X_i^j\right) \tag{5}$$

Where $d(X, X_i^j)$ is the Manhattan distance between X and X_i^j.

Finally, applying the *NN* algorithm it is obtained the following rule:

$$X \in \text{class } C_j \text{ if } D\left(X, C_j\right) = \min_i D\left(X, C_i\right) \tag{6}$$

2.2.2. Implementation of the Obstacle Classification and Detection System.

As it was shown in section 2.2.1, the obstacle detection process by using the classification technique has two stages: learning and classification. The learning process is done through the Classification Training of NI Vision interface that generates a CLF file containing all the information related to the different obstacle classes. In the classification phase, it is loaded the information generated in the training interface and it is performed the specific classification of found objects.

Object Learning (Training interface):
As it was above mentioned, the learning process is done through NI Classification Training software. Figure 11 shows the sequence followed to create the classification file.

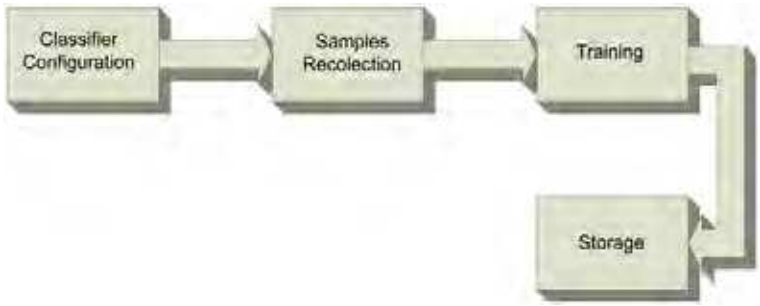

Figure 11. Block diagram of Object Learning phase

- Classifier configuration: features are selected according to specific requirements for the classifying process. It can be controlled the thresholding method called "clustering", that consists in sorting the image histogram into a discrete number of classes according the number of phases detected in the image.
- It is also configured so it can detect brilliant figures because many of the used objects have high white levels.
- In the engine options, it is selected the desired classification method (NN) and the distance metric (Manhattan).
- Sample recollection: are loaded files containing the pattern to be classified. If the image file contains more than one pattern, the desired one can be enclosed in a rectangle. Then, it is identified the class to where the object belongs, and a tag is added. Once the tag is specified one can proceed to add the sample.
- Training: once the sample has been added, it is performed the features vector of the sample that will identify in a unique way a class. This process must be repeated for each added class.
- Storing: at this stage, all the obtained data is saved into a file with information of each class.

Based on the file obtained on the previous section, is possible to create a classification session in a VI of LabVIEW that performs the object classification for the images acquired through the camera.

Objects detection and classification
As it was stated in section 2.1.3, this module is implemented with the pattern matching module in the same VI file, so the first three steps are almost the same; the only difference is that this specific module requires four white images.
This module has as inputs Brightness, Contrast and Gamma from the acquired image, which default values are 50, 45 and 1 respectively. According to the existing illumination in the scene, those values can be adjusted son the vision system can "see" the objects in the visual field. Usually is enough to increase the Brightness value.

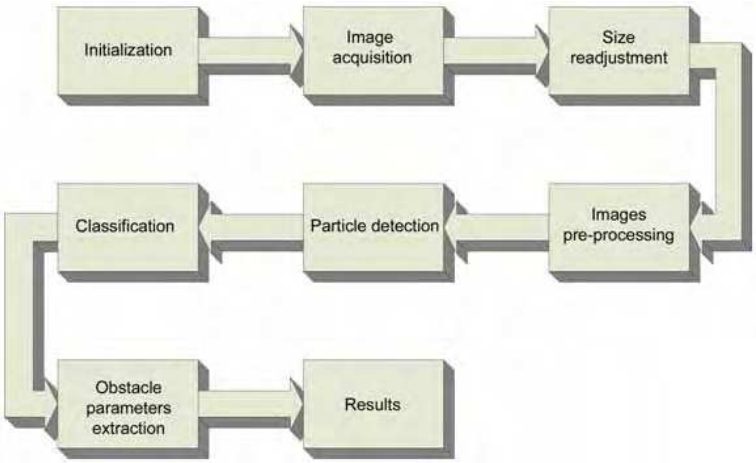

Figure 12. Block Diagram of the Object Detection and Classification Process

The detection and classification process is illustrated in Figure 12, and is implemented in the following way:

- Image pre-processing: a level inversion is applied to the grayscale image and a particle analysis is applied converting the grayscale into a binary image via thresholding. Then, the image is filtered through the morphological process of erosion in order to delete meaningless particles from the particle analysis.
- Particle detection: A classification particle analyzer is applied to the filtered image, which is similar to the simple particle analysis, but this one also provides the mass center of the particles, and the coordinates of the rectangle that enclose it.
- Classification: the particles in the binary images are classified with the particles positions and the classification sections created in the learning process.
- Obstacle parameters: once the object has been classified, it is calculated the diagonal of the smallest null-rotated rectangle that encloses the figure. For the cases of rectangle and square detection it is also extracted the object orientation, by using *Rotation Detect* from IMAQ Vision.

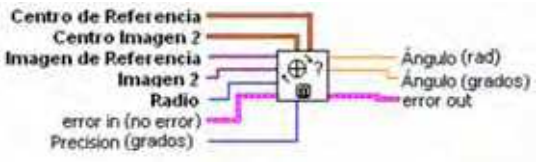

Figure 13. Function of NI-IMAQ package for detecting Object Rotation

- Results: the final outputs for the object detection and classification system are the positions of the detected figures on the original image, values obtained in the particle detection phase. Those values are sorted into an array for each corresponding figure, thus giving three different arrays for object positions. The rotation angles and diagonal lengths are also sorted according to the related figure, so it gives three more arrays for the angles y other three for the calculated diagonals.

4.2.3 Particle *Analysis Technique*

A particle is a group of pixel with non-zero values in an image, usually binary. The particles can be characterized by measurements related to its attributes as position, area, shape and others (National, 2004).

The particle detection consists in applying an erosion process to the original image so it can be removed small particles generated by noise present in the image acquisition. The resulting image is passed through a threshold filter in order to obtain a binary image.

The non-zero pixels and their neighbors with connectivity '8' create a particle with an arbitrary shape, but avoid the apparition of some holes.

The vision system here proposed takes the detected particles and for each one of them extracts the following parameters:

- DF: Maximum Feret's Diameter.
- F_{x1}: X coordinate for the starting point of the DF.
- F_{y1}: Y coordinate for the starting point of the DF.
- F_{x2}: X coordinate for the ending point of the DF.
- F_{y2}: Y coordinate for the ending point of the DF.
- Angle for DF: $\beta = \arctan\left(\frac{F_{y2} - F_{y1}}{F_{x2} - F_{x1}}\right)$
- Coordinates of the points that form the convex hull.

The Feret's Diameter (DF) is the straight line that connects the two most separated particles of a particle, and the convex hull is the convex polygon of minimum area that contains all the points of the particle.

With the coordinates of the convex hull, are found those points its points more separated at each side of the DF, which are used to define a perpendicular line to it. The combination of both straight lines defines the rectangle with lower area that contains the particle. By algebraic manipulation of this rectangle it is obtained the circumscribed ellipse. The Figure 14 shows this process:

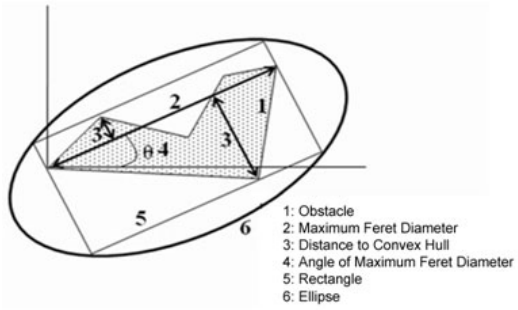

Figure 14. Parameter extraction process from particle analysis

3. Applications of the Vision System

3.1. Velocity Field Generation

As it was previously mentioned, expressing tasks or trajectories in terms of velocity is a research area very important today. It allows the use of velocity controllers, passive controllers, and help to improve the performance of the robot while it is doing its job.

Using the Vision System proposed consisting on a single camera watching the workspace of the robot is possible to detect the obstacles presented on the robot trajectory. The application of Velocity Field Generation based on Artificial Vision constitutes a valuable contribution to the state of the art.
In the following paragraphs a strategy to generate obstacles free velocity trajectories is presented. The problem was divided into two stages linked through the Vision System implementation. These stages are: the generation of an initial velocity field, and the generation of an evader velocity field for each object detected.

3.1.1. Initial velocity Field Generation

The system allows user defined trajectory to be followed by the robot. It can be a hand made one or a set of straight lines. The algorithm developed was tested for 41x41 velocity fields and can be described as follows.
The vision system takes a snapshot of the robot's workspace. This image is cropped to hold only the ROI which is subsampled to a 41x41 image (this resolution of the velocity field offers 1 vector each 5 cm, which is less that a half of the robot dimensions). Over it, the user traces the desired trajectory.
When the user finish defining the desired trajectory, coordinates of the pixels to be part of the trajectory are extracted by a thresholding process and stored in an N-size 1D array of (X,Y) coordinates pairs. N is the number of points or pixels of initial trajectory.
Trajectories can be open or closed. In both cases a sorting process is performed, establishing as sorting parameter the Euclidean distance from one point to another, organizing them from closers to more distant. When the trajectory is open, it is necessary knowing where it begins and where ends. Studying neighbors of each element of the sorted array, the start and end point are obtained.
Then an approximation vector field is defined. For that, it was considered a 2D array of 41x41 elements containing the coordinates (X,Y) of all pixels of a 41x41 image, i.e. element (*i, j*) have a value of (i,j). For each element of the 2D array, the closest element of the trajectory is searched based on the Euclidean distance from point (*i, j*) of the 2D array to each element of the 1D array containing the coordinates of trajectory. Each approximation vector is defined as the normalized one whose direction is obtained from the subtraction of the closest point of trajectory and the point of the 2D array being analyzed.
A tangent vector field is also defined. Each vector is obtained from the subtraction of the element $p + 1$ with element $p - 1$, where p is the index of the closest point of the trajectory to the point (*i, j*) being studied. When the closest point is the point where the trajectory starts, the subtraction is performed between elements $p + 2$ and p ($p = 0$), whereas if it is the ending one, points subtracted are the p and $p - 2$($p = N - 1$). With this assumption, tangent vectors will always have congruent senses. Tangent vectors are normalized too.
The "initial" velocity field is obtained performing a weighted sum, expressed in (7), between the approximation and tangent vector fields. The selection of weights depends directly of the distance between point (*i, j*) and the closest one of the trajectory. As a weight function a sigmoid was chosen. If point (*i, j*) is close to trajectory, the tangent vector will prevail over the approximation one and vice versa.

$$\vec{V}_{ij} = \vec{V}_{a_{ij}} \cdot f_1\left(d_{ij}\right) + \vec{V}_{t_{ij}} \cdot f_2\left(d_{ij}\right) \tag{7}$$

where $\vec{V}_{ij}$ is the vector of the final velocity field, $\vec{V}_{a_{ij}}$ and $\vec{V}_{t_{ij}}$ are the approximation and tangent vectors at *ij*, respectively. d_{ij} is the Euclidean distance from point *ij* to the desired trajectory, whereas $f_1(d_{ij})$ and $f_2(d_{ij})$ are defined by (8) y (9).

$$f_1(d_{ij}) = \frac{2}{1+e^{-0.4 \cdot d_{ij}}} - 1 \tag{8}$$

$$f_2(d_{ij}) = 1 - f_1(d_{ij}) \tag{9}$$

Figure 15 shows the effect of the weighting functions expressed in (8) y (9).
Parameter *a* was chosen to be 0.4 because this value allows an important attenuation of the tangent vectors when $d_{ij} > 6$ (3 times the dimensions of the robot). Figure 15 shows the effect of the weighting functions expressed in (8) y (9).

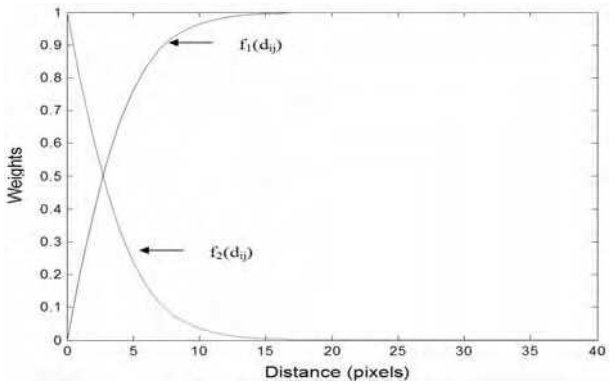

Figure 15. Weighting functions f_1 and f_2 effect. Note that for little distances the tangent vector is greater than the approximation one, and vice versa

3.1.2. Dynamic Velocity Field Modification

The "evader" velocity field generation module takes information provided by the vision system, parameterizes the correspond ellipse for each obstacles and create a velocity field that surrounds the object.
The proposed algorithm contemplates dividing the ellipse into four regions: one for entry, one for exit and two for transitions.
In the transition regions the velocity field is chosen to be parallel to the trajectory given by the ellipse contour, i.e. tangent to the ellipse. The general tangent line equation at any point (X_0, Y_0) is given by (10).

$$\frac{(X-P)\cdot(X_0-P)}{A^2} + \frac{(Y-Q)\cdot(Y_0-Q)}{B^2} = 1 \tag{10}$$

where (P, Q) are the coordinates of the location of the ellipse and *A* and *B* represent a half of the major and minor axes respectively. From (10), the unit tangent vector at any point (X_0, Y_0) can be deduced to be

$$\vec{V}_t(X_0, Y_0) = \left(V_{t_x}, V_{t_y}\right) \tag{11}$$

$$V_{t_x}(X_0, Y_0) = \frac{A^2 \cdot (Y_0 - Q)}{\sqrt{A^4 \cdot (Y_0 - Q)^2 + B^4 \cdot (X_0 - P)^2}} \tag{12}$$

$$V_{t_y}(X_0, Y_0) = \frac{-B^2 \cdot (X_0 - P)}{\sqrt{A^4 \cdot (Y_0 - Q)^2 + B^4 \cdot (X_0 - P)^2}} \tag{13}$$

In the entry region the field is defined in the direction and sense toward the ellipse contour and is turned aside smoothly until it converges to the tangent vector as the point is closer to the transition region. This is achieved through a weighted sum of the approximation and tangent vectors to the ellipse, where the weights depends on the proximity to the distance from a given point to the edge between entry and transition regions.
Entry and exit regions are always of the same size. Transitions regions too. The size (angle) for each region is chosen such as they have an area equal to a quarter of the ellipse's area. To accomplish this requirement, the area of the entry (or exit) region is given by (14)

$$\int_{\vartheta-\alpha}^{\vartheta+\alpha} \int_0^{r(\gamma)} \rho d\rho d\gamma = \frac{1}{2} \cdot \int_{\vartheta-\alpha}^{\vartheta+\alpha} \rho^2 d\rho = \frac{\pi \cdot A \cdot B}{4} \tag{14}$$

Consider Figure 16 where angles related to an ellipse are defined. Based on it, is possible to obtain the relations shown in (15).

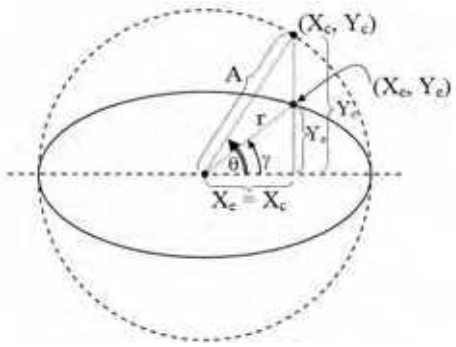

Figure 16. Definition of Angles and coordinates of the point belonging to the ellipse

$$\begin{aligned} &Xc = A \cdot \cos(\theta) \qquad X_e = r \cdot \cos(\gamma) \\ &Y_c = B \cdot \sin(\theta) \qquad Y_e = r \cdot \sin(\gamma) \qquad Y_e = B \cdot \sin(\theta) \end{aligned} \tag{15}$$

r is defined by (16) in terms of θ, or, considering (17), it can be defined by (18) in terms of γ.

$$r(\theta) = \sqrt{A^2 \cdot \cos^2(\theta) + B^2 \cdot \sin^2(\theta)} \tag{16}$$

$$\tan(\theta) = \frac{A}{B} \cdot \tan(\gamma) \tag{17}$$

$$r(\gamma) = A \cdot \sqrt{\frac{1+\tan^2(\gamma)}{1+\left(\frac{A}{B}\right)^2 \cdot \tan^2(\gamma)}} \tag{18}$$

Solving (14), the size (angle) of the entry and exit region is defined by (18).

$$2 \cdot \varphi = 2 \cdot \arctan\left(\sqrt{\frac{\left(\frac{B}{A}\right)^2 + \tan^2(\gamma)}{1+\left(\frac{B}{A}\right)^2 \cdot \tan^2(\gamma)}}\right) \tag{19}$$

The orientation of regions is given by the angle of the original field at the point where the object is located. Regions must be rotated for achieving an entry region aligned with the original velocity field.

For the exit region the same approach used for the entry region is employed. However, in this case, the field is defined leaving the ellipse.

Approximation vectors a any point (X_0, Y_0) is given by (20)

$$\vec{V}_a(X_0, Y_0) = \left(V_{a_x}, V_{a_y}\right) \tag{20}$$

$$V_{a_x}(X_0, Y_0) = \frac{B^2 \cdot (X_0 - P)}{\sqrt{A^4 \cdot (Y_0 - Q)^2 + B^4 \cdot (X_0 - P)^2}} \tag{21}$$

$$V_{a_y}(X_0, Y_0) = \frac{A^2 \cdot (Y_0 - Q)}{\sqrt{A^4 \cdot (Y_0 - Q)^2 + B^4 \cdot (X_0 - P)^2}} \tag{22}$$

The division proposed by using the defined regions wants to achieved an "evader" velocity field similar to the sketch shown in Figure 17.

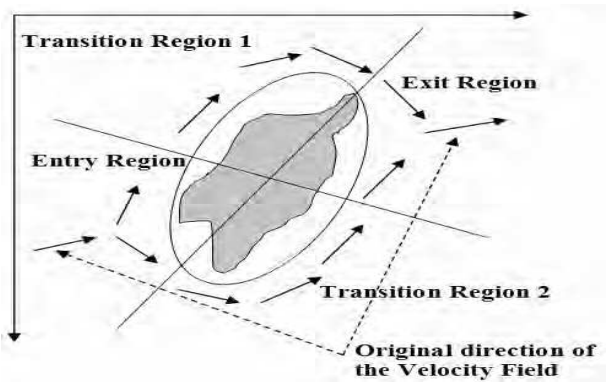

Figure 17. Sketch of "evader" field. Following the direction and sense of the initial field at the point where is located the obstacle, the different regions are defined. Note the deviations of field in the entry and exit regions

3.1.3. Results of the Vision-Based Velocity Field Generator

For testing the "initial" velocity field generator two hand made traces was introduced. Figure 18 shows the obtained velocity fields.

In case (a) the trajectory is open, has an 'Z' shape, and the field generated converged successfully to it; inclusive, the end point of the trajectory results to be a sink, as it was desired. Case (b) corresponds to the well known circular trajectory (Li & Horowitz, 1995) (Moreno & Kelly, 2003c), here hand-traced. It is observed that velocity field converged as expected. It is important to remark that while the distance to the desired trajectory is higher the approximation vector prevails over the tangent one, and when it is lower, the tangent one prevails.

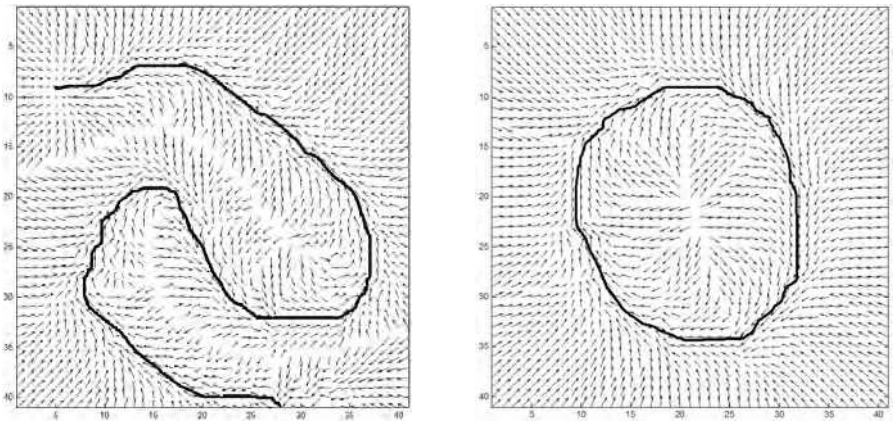

Figure 18.Velocity Field. Note the hand made desired trajectory remarked in black

The "evader" algorithm responds to an arbitrary object as shown in Figure 19.

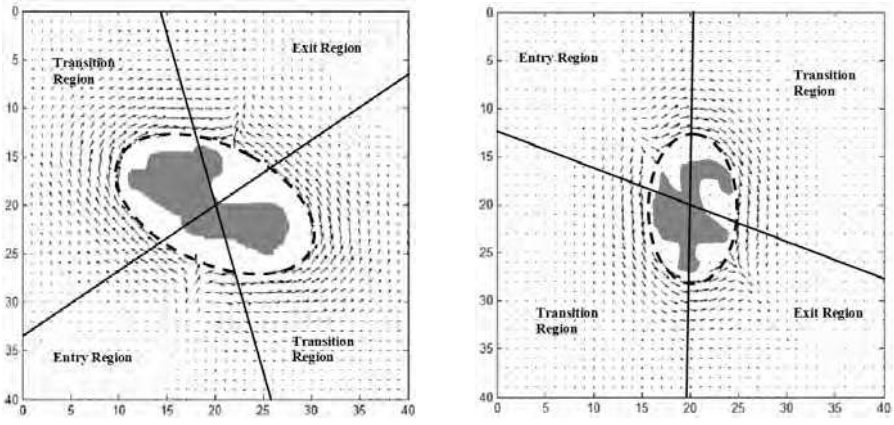

Figure 19. "Evader" velocity field for an arbitrary object. The four predefined regions are shown and the behavior of the algorithm can be observed

Figure 19.a shows an object located on a place where the original field has an orientation of −75° and the circumscribed ellipse has 25°. Figure 19.b presents the evader field for an object whose circumscribed ellipse has 90° and the original field at the point the object is placed has 55°. In both figures the exponential fading imposed is shown. This effect assures that the evader field only affects the neighborhood of the object.
Now it is presented a test of the system with the three modules combined. Figure 14 resumes the results obtained.

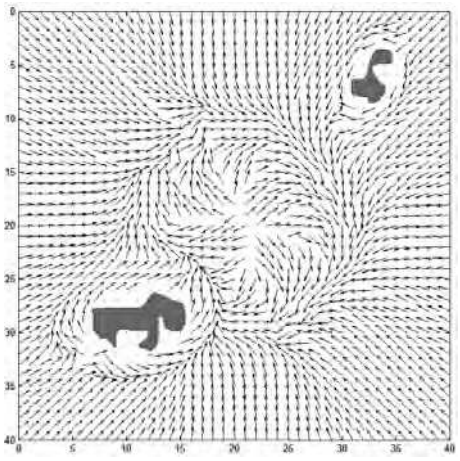

Figure 20. Modified velocity field for two obstacles detected

Inserting two arbitrary objects at two arbitrary positions for the circular velocity field shown in Figure 20, the final velocity field obtained offers a free-obstacles trajectory, however, at the edges between the influence "evader" field and the initial one, it is not enough smooth as desired.

3.2. Fuzzy Logic Controller and Velocity Field Control

The purpose of this system is to control the position of a small wheeled mobile platform on a two dimensional work space, using an overhead vision system. The main sensor used is a wireless camera placed at 2.7 m from the floor and able to observe all the workspace of the mobile platform. This camera can provide a maximum of 30 frames per second of 640x480 pixels, but the sample rate used is determined by the processing speed of the algorithm, since it works over a snapshot taken on each cycle and, usually, processing cycle is larger than the sampling rate. The image acquired is cropped in order to provide an image of 480x480 pixels, covering an area of 6.25 m^2.
For test purposes, a 2.4GHz Wireless Color Camera model XC10A was used, connecting it to the PC by a generic RCA to USB adapter. The robot employed was a differential drive Lynxmotion Carpet Rover The software was developed employing LabVIEW v7.1 with NI IMAQ Vision v7.1 and Matlab v7.1. All the experiments were done with RGB images, and the surface of the workspace was not altered to deal with the differences of luminosity and other typical issues associated with vision systems.

3.2.1. *Vision System Description*

The objective of this stage is to obtain the position (in pixels) and the angle (in degrees) of the mobile platform at any moment. This is achieved using a slight modification of the system proposed by (Bolaños et. al., 2006).

Mobile Robot Detection

The pattern matching algorithm (National, 2004) consists in the localization of regions that match with a known reference pattern on a RGB image. The reference pattern is also known as template, and contains information related to edge and region pixels, removing redundant information in a regular image. In this way, the matching process is done in a faster and more accurate manner. In the case where a pattern appearing rotated in the image is expected, it is possible to store pixels specially chosen, whose values reflect the pattern rotation.

The comparison between the template and different regions of the camera image in the pattern matching process is done using a correlation algorithm based in the calculus of the squared Euclidean distance:

$$d^2_{f,t}(u,v) = \sum_{x,y} \left[f(x,y) - t(x-u, y-v) \right]^2 \tag{23}$$

where f is the image. The sum is performed over (x, y), in the window containing the sub-image t located at (u, v). Expanding d^2, and making some considerations (Bolaños et. al, 2006), the remaining term of the cross correlation

$$C(u,v) = \sum_{x,y} f(x,y) \cdot t(x-u, y-v) \tag{24}$$

is the similarity or matching measure between image and template.

Process Description

The detailed description of the pattern matching algorithm implementation is as follows (Bolaños et al., 2006):

- Initialization: Two RGB blank images are generated.
- Image capture: A real RGB image from the workspace is captured.
- Cropping: The captured image is cropped to a 480x480 pixels (from 640x480 pixels). This image size allows the visualization of a workspace of 6.25 m^2.
- Information load: The information (related to pattern learning) contained in the PNG image stored in the learning process is loaded.
- Pattern matching module setup: The pattern matching module is set to rotation-invariant mode so it can detect the desired pattern regardless of its rotation.
- Matching: The matching process is done according to the configuration above described between the captured image and the loaded image (with the information from the learning process). If the desired pattern is located, the result will be its position within the image and its orientation.

3.2.2. Graphic Interface

The graphic interface was made in LabVIEW v7.1. First the user selects the template the software will try to find, in this case the mobile platform, with an angle of 0º (Figure 21.a).

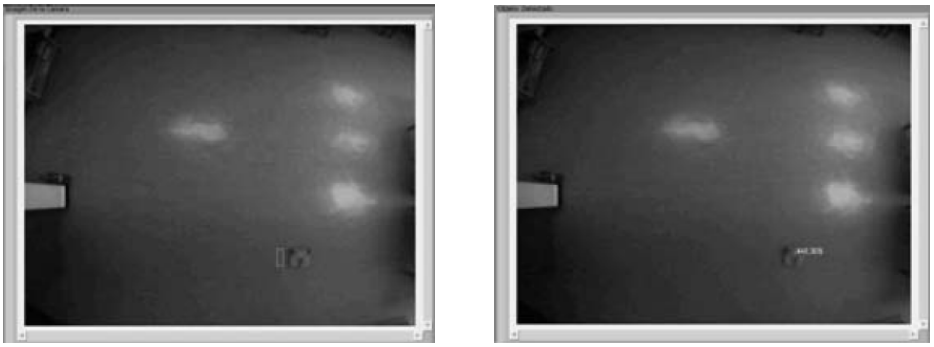

Figure 21. Template selection process and pattern matching illustration

Once the template is selected the software is able to find the best match and return the X and Y coordinates of its position in pixels (Figure 21.b).

In practice, there is a problem due the uncontrolled luminosity. Sometimes the algorithm gives false matches which affect the whole system. To deal with this problem, the developed program was allowed to give not only the match with the highest correlation but also others with a lower correlation. A discriminator then verifies which of the matches is inside the neighborhood of the last position. Besides, the algorithm gives a noisy estimation, which is reduced by filtering it.

With these considerations the response of the system is highly improved, obtaining an accurate position in most of the cases.

3.2.3. Controllers Implementation

In this case the task of the controller is to move the platform from one point to another regardless of the trajectory. The problem is that the system is highly non-linear and even if lineal controllers have shown good performance when non-linear systems are linearized, other controllers exhibit better performance. Neural and fuzzy controllers are among them.

The control technique was based in the idea of decoupling the locomotion system of the mobile platform. To achieve this, the error in X and Y coordinates (Cartesian mode) is transformed into magnitude and angle errors (Polar mode), then these errors are taken by the controller which gives a new references for linear and rotational velocity. Finally these references are transformed into references of right and left velocities (Dudek & Jenkin, 2000) by the equation (25).

$$\begin{aligned} v_{left} &= v_{linear} + w \cdot a \\ v_{right} &= v_{linear} - w \cdot a \end{aligned} \tag{25}$$

A. Linear Controller

This controller was a conventional PI where the error and the sum of previous errors are added to generate the control signal. The inputs of the control system are distance and angle

errors and the outputs are references for right and left velocities directly. The control scheme employed is shown in Figure 22.

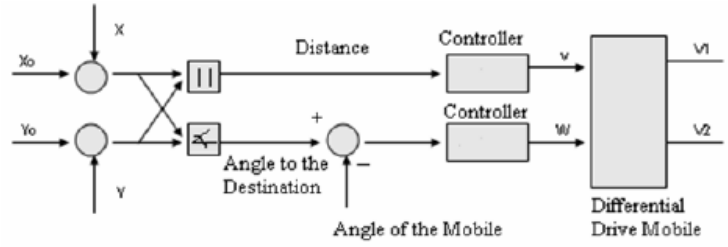

Figure 22. Linear control scheme

B. Fuzzy Logic Controller

As an option to the linear controller, a fuzzy logic one was designed to work on the same inputs and outputs as the linear controller. The FIS Editor of Matlab v7.1 was used for this purpose. Triangular membership functions were chosen because of their simplicity to implement in microcontrollers.

The range of each of these variables was divided in the following membership functions:

- Angle Error (Degrees): Small Positive (0. -90.), Big Positive (90. -180.), Small Negative (180. -270.) and Big Negative (270. -360.).
- Distance to the destination: Far (150 cm - 800 cm), close (20 cm -150 cm) and very close (0 cm - 40 cm).
- Velocities (Left and Right wheels): Fast (8.5cm/s), Medium (5 cm/s), slow(3.5cm/s) and very slow(0.8cm/s).

With these membership functions the system of fuzzy inference is based on the Mamdani's aggregation method (Ying, 2000), with 9 fuzzy rules, and defuzzyfication technique based on the gravity center.

The base of rules was formed like is shown in Table 2.

Distance	Angle Error	Left Velocity	Right Velocity
Very Close	X	Very Slow	Very Slow
Close	Small Positive	Slow	Very Slow
Close	Small Negative	Very Slow	Slow
Close	Big Positive	Medium	Very Slow
Close	Big Negative	Very Slow	Medium
Far	Small Positive	Fast	Medium
Far	Small Negative	Medium	Fast
Far	Big Positive	Fast	Slow
Far	Big Negative	Slow	Fast

Table 2. Fuzzy Rules Base

Proposed linear controller for this system is able to guide the robot towards the desired position, but presented some problems. Figure 23 shows the resulting trajectory when the

proposed linear controller was used, and can be observed how the robot oscillates near the final position. Oscillations at the end point were very strong around it and impossible to eliminate, since controllers parameters that worked well far from the destination, did not give good results in its proximity. The controlled variables saturated in some circumstances, affecting the response of the controller. Also the tuning of the controller was very difficult due to the interdependence between v_{linear} and ω.

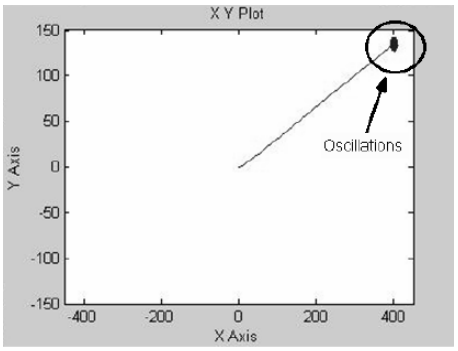

Figure 23. Resulting Trajectory with the linear controller

The Fuzzy controller response shown in Figure 24 was more reliable. The robot stops at destination showing a good behavior both far and in the proximity of the destination. Also, the characteristics of the membership functions ensure that the outputs won't saturate and its decoupling facilitates the design and tuning of the controller.

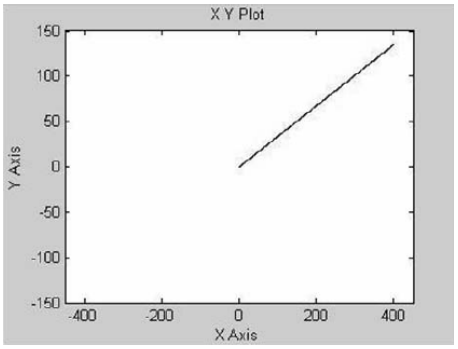

Figure 24. Resulting Trajectory with the fuzzy controller

The system was able to produce a reliable position measurement based only in the visual sensor and the related vision system, working in difficult situations of illumination and noise, thanks to the spatial continuity considerations taken into account.
The implemented controllers guided the robot towards the final destination, but the fuzzy system showed advantages over the linear controller both in the design and tests stages. The fuzzy logic controller has a better performance mainly because it is non-linear and is designed to deal with non-linear systems. Also it can absorb possible errors of the vision platform, by minimizing its effect in the controller. Additionally it is easier to tune than the linear one.

5. References

Bolaños, J. M.; Medina-Meléndez, W.; Fermín, L.; Cappelletto, J.; Fernández-López, G. & J. C. Grieco. (2006). Object recognition for obstacle avoidance in mobile robots. *Artificial Intelligence and Soft Computing, ICAISC 2006, Lecture Notes in Computer Science*, pp. 722–731, ISBN: 3-5403-5748-3, Zakopane, Poland, June 2006.

Cervantes I.; Kelly, R.; Alvarez, J. & Moreno J. (2002). A robust velocity field control. *IEEE Transactions on Control, Systems and Technologies*, Vol. 10, No. 6, (November 2002) 888–894, ISSN: 1063-6536.

Dixon, W. E.; Galluzo, T.; Hu, G. & Crane, C. (2005). Adaptive velocity field control of a wheeled mobile robot. *Proceedings of the 5th International Workshop on Robot Motion and Control, RoMoCo '05*, pp. 145–150, ISBN: 83-7143-266-6, Poznan, Poland, June 2005.

Dudek, G. & Jenkin M. (2000) *Computational Principles of Mobile Robotics*, Cambridge University Press, ISBN: 0-5215-6876-5, U.S.A.

Kelly, R.; Moreno, J. & Campa, R. (2004). Visual servoing of planar robots via velocity fields. *Proceedings of the IEEE 43rd Conference on Decision and Control*, pp. 4028–4033, ISBN: 0-7803-8682-5, Atlantis, Paradise Island, Bahamas, December 2004.

Kelly R.; Bugarín, E. & Campa, R. (2004). Application of velocity field control to visual navigation of mobile robots. *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles*, on CD, Lisbon, Portugal, June 2004.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots, *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 500–505, St. Louis, U.S.A., March 1985.

Li, P. Y. & Horowitz, R. (1995). Passive velocity field control of mechanical manipulators. *Proceedings of the IEEE International Conference on Robotics and Automation ICRA'01*, pp 2764–2770, ISBN: 0-7803-1965-6, Nagoya, Japan, April 1995.

Li, P. Y. (1999). Adaptive passive velocity field control. *Proceedings of the 1999 American Control Conference*, Vol. 2, pp. 774–779, ISBN: 0-7803-4990-3, San Diego, U.S.A., June 1999.

Li, P. Y. & Horowitz R. (1999). Passive velocity field control of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 4, (August 1999) 751–763, ISSN: 1042-296X.

Li, P. Y. & Horowitz R. (2001). Passive velocity field control (PVFC): Part I - Geometry and Robustness. *IEEE Transactions on Automatic Control*, Vol. 46, No. 9, (September 2001) 1346–1359, ISSN: 0018-9286.

Li, P. Y. & Horowitz R. (2001). Passive velocity field control (PVFC): Part II - application to contour following. *IEEE Transactions on Automatic Control*, Vol. 46, No. 9, (September 2001) 1360–1371, ISSN: 0018-9286.

Mitchell, T. (1997). *Machine Learning*, McGraw-Hill Science/Engineering/Math, ISBN: 0-0704-2807-7, U.S.A.

Moreno J. & Kelly R. (2003a). On manipulator control via velocity fields. *Proceedings of the 15th IFAC World Congress*, pp. 1420–1427, Barcelona, Spain, Julio 2003.

Moreno J. & Kelly R. (2003b). Velocity control of robot manipulator: Analysis and experiments. *International Journal on Control*, Vol. 76, No. 14, (September 2003) 1420–1427, ISSN: 0020-7179.

Moreno J. & Kelly R. (2003c). Hierarchical velocity field control for robot manipulators. *Proceedings of the IEEE International Conference on Robotics and Automation ICRA'03*, Vol. 3, pp. 4374–4379, ISBN: 0-7803-7736-2, Taipei, Taiwan, September 2003.

National Instruments (2004). *IMAQ Vision Concepts Manual*, National Instruments.

Santos-Victor, J. & Sandini G. (1997). Visual Behaviors for Docking. *Computer Vision and Image Understanding: CVIU*, Vol. 67, No. 3, (September 1997) 223–238, ISSN: 1077-3142.

Seelinger, M.; Yoder J-D.; Baumgartner, E. T. & Skaar, S. B. (2002). High-precision visual control of mobile manipulators. *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 6, (December 2002) 957–965, ISSN: 1042-296X.

Skaar, S. B.; Yalda-Mooshabad I. & Brockman W. H. (1992). Nonholonomic camera-space manipulation. *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 4, (August 1992) 464–478, ISSN: 1042-296X.

Yamakita, M. & Suh, J. H. (2000). Adaptive generation of desired velocity field for cooperative mobile robots with decentralized PVFC. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'00*, pp. 1841–1846, ISBN: 0-7803-6348-5, Takamatsu, Japan, Oct./Nov. 2000.

Yamakita, M. & Suh, J. H. (2001). Adaptive generation of desired velocity field for leader follower type mobile robots with decentralized PVFC. *Proceedings of the IEEE International Conference on Robotics and Automation ICRA'01*, pp. 3495–3501, ISBN: 0-7803-6576-3, Seoul, Korea, May 2001.

Ying, H. (2000). Fuzzy Control and Modeling: Analytical Foundations and Applications, IEEE Press Series on Biomedical Engineering, Wiley-IEEE Press, ISBN: 0-7803-3497-7, U.S.A.

Vision Systems: Applications

Edited by Goro Obinata and Ashish Dutta

ISBN 978-3-902613-01-1

Hard cover, 608 pages

Publisher I-Tech Education and Publishing

Published online 01, June, 2007

Published in print edition June, 2007

Computer Vision is the most important key in developing autonomous navigation systems for interaction with the environment. It also leads us to marvel at the functioning of our own vision system. In this book we have collected the latest applications of vision research from around the world. It contains both the conventional research areas like mobile robot navigation and map building, and more recent applications such as, micro vision, etc.The fist seven chapters contain the newer applications of vision like micro vision, grasping using vision, behavior based perception, inspection of railways and humanitarian demining. The later chapters deal with applications of vision in mobile robot navigation, camera calibration, object detection in vision search, map building, etc.

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