A Visual Servoing Robot Control Architecture

RAZVAN TUDOR TANASIE*, DORIAN COJOCARU**
*Software Engineering Dept., **Mechatronics Dept.
University of Craiova
Blvd. Decebal, Nr. 107, 200440, Craiova
ROMANIA

Abstract: - This paper presents a project dealing with visual servoing applications. A general architecture for the software control system was developed. Each of the modules was already implemented in beta versions: image processing techniques adapted for visual servoing applications, path planning algorithms for visual servoing – elaboration and simulation, system state monitor which surveys and coordinates the activities of the other modules. Fuzzy algorithms and methods were used in order to support the implementation of the image processing procedures. The architecture is completed by an image acquisition module and a servo-control module. The project is financed by Romanian National University Research Council CNCSIS.

Key-Words: - visual servoing, image processing, path finding, collision detection, computer graphics, fuzzy techniques.

1 Introduction
The goal of a visual servoing system [20] is to control the movement of a robot throughout the environment to the target, using the information obtained from digital images which are acquired from a camera or a system of digital cameras.

Usually, every visual application is implemented in a particular way based on its own hardware and software structures.

This architecture was created so that to achieve the following principles:

- Portability – the software has to be developed in such a way that it can be used with different hardware architectures. The software has to be platform independent.
- Additivity – the architecture must allow easy expansion by adding new modules which would bring new functions to the system.
- Modularity – the activity of the already implemented modules must not be affected by the addition of a new module to the architecture.
- Parallelism – all the modules work in parallel.

Fig. 1 Visual servoing architecture
Multiple goals – the control algorithm has to be able to accomplish more than one goal. The avoidance of an approaching obstacle is an immediate goal, while the reach of the target goal is a long term one. The planning algorithm has to take into consideration all the goals when creating the movement control plan.

For these principles to be fulfilled, a modular architecture was chosen (Fig. 1):
- a data acquisition module, which includes image acquisition,
- an image processing module for visual servoing, which has the structure presented in Fig. 2,
- a path planning module for robotic application (Fig. 3),
- a visual servoing module, including robot control Kinematics and dynamic laws and robot servo-controller,
- a monitor module which surveys and coordinates the activities of the other modules through communication channels.

All the modules work in parallel. The image acquisition module obtains images from the robot work space. The output of this module is an image. This image is processed by the image processing module. This module uses fuzzy techniques to enhance the image [3][4], to segment the enhanced image [14] in order to identify the obstacles and the target and afterwards creates a map of the environment. The output of this module is this map.

The path planning module [2] creates an algorithm for path finding based on this map, adapts it to the changes that appear in the environment, tests for collisions and finally simulates the elaborated algorithm. It outputs a control path plan.

The visual servoing module takes the path plan and controls the robot [5][9] according to it. The monitor module inspects the state of the robot (did it collide with an obstacle or not, did it reach the target, is the target unreachable). This module is linked to all the other modules through communication channels. It is informed by the other modules of problems and malfunctions occurrences and it can temporarily suspend or even stop the activity of the servo-controller in the event of such an occurrence.

Regarding robot control [10][12], the research team designed and implemented a wide variety of control laws. Robotic control laws [11][23] were adapted in order to be used in virtual servoing applications.

Following, each module will be presented. The paper will focus on the presentation of the image processing module and path finding sub-module which have already been implemented.

### 2 Image Processing Module

Typically applications ask the enhancement process to be capable of removing noise, smoothing regions where gray levels do not change significantly, and sharpening abrupt gray-level changes. It is, however, hard to incorporate all these requirements into one framework since smoothing a region might destroy a line or an edge, a sharpening might lead to unnecessary noise. A good enhancement process is, therefore, required to be adaptive so that it can process each region differently based on the region properties. By using crisp logic only one of the three filters can be applied at a time. This is of course a drawback. Since fuzzy logic can easily incorporate heuristic knowledge concerning a specific application in the form of rules, it is ideally suited for building an image enhancement system.

As a general concept, the job of the cluster analyze is the data partition into a number of groups, or clusters. Applying this partitioning operation on images, the image segmentation - a very important task for image processing - is obtained.

#### 2.1 Fuzzy Image Enhancement Algorithm

This algorithm is based on Choi’s and Krishnapuram’s algorithm [6][18] who view image
enhancement as replacing the gray level value of every pixel in the image with a new value depending upon the local information. If the local value region is relatively smooth, then the new value of the pixel may be a type of average of the local values. On the other hand, if the local region contains an edge or noise points, a different type of filtering should be used. So it is needed to create a bank of filters from which to choose the needed one. The choosing, unlike in crisp algorithms, need not be of only one of the filters but according to fuzzy logic each filter is selected in a certain proportion. The filter selection criteria constitute the antecedent clauses of the fuzzy rules, and the corresponding filters values constitute the consequent clauses of the fuzzy rules. The algorithm is based on the if–then–else rule paradigm referred to as FIRE (Fuzzy Inference Ruled by Else-action).

In order to enhance the image, the algorithm uses a spatial window which has a radius that can be defined by the user. This window (matrix) "slides" over the entire image, the central pixel of the matrix being enhanced using the influence of the other pixels of the window. This application uses three filters:

- Filter A, edge sharpening filter,
- Filter B, impulse noise removal filter,
- Filter C, smoothing filter.

The FIRE system of rules used is:

If $M$ is $mic$, then use filter B
If $M$ is $mare$, then use filter C
Else use filter A

$M$ is a coefficient that is computed for each pixel and its value depends directly on the values of the pixels in the sliding matrix and indirectly, and consequently in a smaller proportion, on every pixel in the region. The membership functions for the linguistic labels “mare” and “mic” are defined on the domain of $M$ ($0 \leq M \leq 1$). They are fuzzy membership functions that take an exponential form. These functions depend on two parameters: $M$ and the image enhancement coefficient. Its value must be selected carefully considering the type of the image that is enhanced. In order to solve the if-then-else rules system, multiplication is selected as composition operator, minimum for aggregation, weighted average for the defuzzyfication function [22] and the filter output for the consequent, so the following output is obtained:

$$I(X_i) = \frac{c_1 \cdot f_b + c_2 \cdot f_c + c_3 \cdot f_a}{c_1 + c_2 + c_3}$$

where: $I(X_i)$ = the gray-level of pixel $X_i$; $c_1 = mic$ ($M$); $c_2 = mare$ ($M$); $c_3 = 1 - \max(c_1, c_2)$; $f_b$ = the output from filter A; $f_c$ = the output from filter B; $f_a$ = the output from filter C;

After applying the defuzzyfication process to the entire image, the enhanced image is obtained.

### 2.2 Fuzzy Image Segmentation Algorithm

There are many techniques of fuzzy image segmentation: histogram thresholding, edge based segmentation, region growing, fuzzy clustering algorithms, fuzzy rule-based approach, fuzzy integrals, measures of fuzziness and image information, fuzzy geometry, but among them the most dominant are fuzzy clustering and fuzzy rule based segmentation techniques.

Fuzzy C-means [7] is an algorithm based on one of the oldest segmentation methods which allows data to have membership of multiple clusters, each to varying degrees. The algorithm is based on minimization of the following function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2,$$  \quad \text{for } l \leq m < \infty \quad (2)

where: $m$ is any real number greater than 1; $u_{ij}$ is the degree of membership of $x_i$ in the cluster $j$; $x_i$ is the $i^{th}$ of d-dimensional measured data; $c_j$ is the d-dimension center of the cluster; $\|\cdot\|$ is any norm expressing the similarity between any measured data and the center. This algorithm realizes an iterative optimization of the $J_m$ function, updating membership $u_{ij}$ and the cluster centers $c_j$ using the following formulas:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left( \frac{\|x_i - c_j\|^m}{\|x_i - c_k\|^m} \right)^{m-1}} \quad (3)$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{j=1}^{N} u_{ij}^m} \quad (4)$$

The minimization of $J_m$ is achieved only when the $u_{ij}$ function saturates, that is, the stop criterion is given by the equation:

$$\max_{ij} \left\{ \left| u_{ij}^{(k+1)} - u_{ij}^{(k)} \right| \right\} < \varepsilon \quad (5)$$
where \( \epsilon \) is a number between 0 and 1, and \( k \) is the iteration step.

Fuzzy c-means algorithm has the following steps:
- Consider a set of \( n \) data points to be clustered, \( x_i \).
- Assume that the number of clusters, \( c \), is known, \( 2 \leq c < n \).
- Choose an appropriate level of cluster fuzziness, \( m \in \mathbb{R} > 1 \).
- Initialize the \( (n \times c) \) sized membership matrix \( U \) to random values such as \( u_{ij} \in [0,1] \) and \( \sum_{j=1}^{c} u_{ij} = 1 \).
- Calculate cluster’s centers \( c_j \) using (9) for \( j = 1 \ldots c \).
- Calculate the distance measures \( d_{ij} = \left\| x_i - c_j \right\| \), for all clusters \( j = 1 \ldots c \) and data points \( i = 1 \ldots n \).
- Update the fuzzy membership matrix \( U \) according to \( d_{ij} \).
  - If \( d_{ij} > 0 \) then \( u_{ij} = \left[ \sum_{k=1}^{c} \left( \frac{d_{ik}}{d_{ij}} \right)^{m-1} \right]^{-\frac{1}{m-1}} \).
  - If \( d_{ij} = 0 \) then the data point \( x_j \) coincides with the cluster center \( c_j \), and so full membership can be set \( u_{ij} = 1 \).
- Repeat from 5th step until the change in \( U \) is less than a given tolerance, \( \epsilon \).

This algorithm’s fuzzy behaviour is given by the membership function, which links the data to each cluster [16][21]. Matrix \( U \) factors represent the degree of membership between the centers of the clusters and the data. The smaller \( m \) is, the crisper the algorithm \((m=1)\) represents a crisp algorithm), while the greater \( m \), the fuzzier the clusters are (smaller values for the membership function).

2.3 Map Construction

The segmentation process determines the positions of the obstacles and the target. Based on the dimensions of the environment, an \( n \) by \( m \) locations map is constructed. Each location can be:
- Robot,
- Obstacle,
- Target,
- Vacant.

The dimension of a location is created equal to the robot dimension. The image processing module generates such a map which can also be implemented as a mesh graph. An example of such a map is presented in Fig. 4, where a black location denotes an obstacle, a red location represents the target, a blue location – the robot and a blank location a vacant space.

2.4 Module Implementation

The module was implemented in Microsoft Visual C++ from the package Microsoft Visual Studio .NET. In order to test the application, a user interface was developed. It has a menu from which the user can select the desired function and a toolbar from which images can be opened, saved or the about dialog can be also opened. The functions implemented by this program are mainly those presented in the algorithms in the earlier chapters.

The user has the possibility to open and work with .bmp grayscale images. No function is allowed before an image is loaded. Once an image loaded, the user can enhance the image, using his own parameters (in a specified range), draw the histogram of either the opened image or both it and the enhanced image and compare them. The user may choose to find the edges, apply image segmentation, threshold the original image or the image containing the edges. All of these functions are parameterized, parameters which can be chosen according to wishes of the client. Once the image was segmented, the user may choose to generate the map and to create a graph based on the map, graph that is saved in a file which will be used by the path planning module.

3 Path Planning Module

The path planning module creates an algorithm for path finding based on the map generated by the
image processing module (the graph read from file), adapts it to the changes that appear in the environment, tests for collisions and finally simulates the elaborated algorithm. It outputs a control path plan.

This module has four components:
- Kinematics computing,
- Path finding,
- Collision detection,
- Simulation.

3.1 Path finding

The goal of the robot is to start from its current position and reach a target location. The movement can take place in either a static or dynamic environment. If the obstacles are fixed, than, with a good path planning algorithm, collisions can be avoided and the target can be got to, if it is not unreachable (all the paths leading to it are blocked by obstacles). In this case the only errors that must be corrected are physical ones, which appear due to the mechanics, kinematics and dynamics of the robot [15].

A path finding algorithm requires considering an environment map and elaborating a step by step plan which will present the robot motion, location by location, until the target is reached.

The algorithm proposed is based on the principle of artificial potential field, more exactly, on attraction and repulsion forces.

The scope of the robot is to reach the target. This target is like an attraction pole for the robot. In order to illustrate this in the path finding algorithm, the target is considered to generate an artificial field which induces an attraction force $F_a$. This force is strongest near the target, and grows weaker while moving away from it.

This force would be enough to guide the robot to target on the shortest path in an obstacle free environment. This is generally not the case. In order to make the robot avoid the obstacles, these are made to generate an artificial field which repulses the robot. This field induces a repulsion force $F_r$. This force is strongest near the obstacle that generates it, and grows weaker while moving away from it.

There are two important aspects about these repulsion forces that must be underlined:

- $F_r < F_a$  \hspace{1cm} (6)

The main goal of the robot is still reaching the target and not moving as further as possible from the obstacles. The influence of an obstacle should be limited in order not to interfere with the robot motion when this is far away from the obstacle. For this, the repulsion force must be chosen carefully.

- $F_r = \sum_{i=0}^{p} F_{r}^{i}$  \hspace{1cm} (7)

where $p$ is the number of obstacles.

The repulsion forces are additive. That is, the total repulsion force in a location is equal to the sum of the repulsion forces generated by each obstacle for that location.

The path finding algorithm uses the graph generated from the map and gives weights to each node. The node weights are computed according to the formula:

$$W_l = F_a^l + F_r^l = F_a^l + \sum_{i=0}^{p} F_{r}^{i,l}$$  \hspace{1cm} (8)

where $W_l$ represents the weight of node $l$, $F_a^l$ is the attraction force generated in node $l$ by the target and $F_r^l$ is the summed repulsion force generated in node $l$ by all the obstacles.

After the weights are computed the path is generated. At each step the robot is instructed to move to the neighbor location that has the biggest weight. The location from where it arrived to its current position is not considered in this comparison.

If the functions that implement the forces are not carefully selected, loops may appear. In order to avoid that, a list of visited nodes is always kept. If a node is revisited, the loop is deleted from the path and the node with the next biggest weight is selected as the new step.

As an example consider the following attraction force:

$$F_a = \begin{cases} n - d_r, & n \geq m \\ m - d_r, & n < m \end{cases}$$  \hspace{1cm} (9)

where $n$ is the length of the map and $m$ is its width and $d_r$ is the distance to the target.

The repulsion force of obstacle $i$ is defined as:

$$F_r^i = \begin{cases} \frac{n - d_o^i}{2}, & n \geq m \\ \frac{m - d_o^i}{2}, & n < m \end{cases}$$  \hspace{1cm} (10)

where $n$ is the length of the map and $m$ is its width and $d_o^i$ is the distance to obstacle $i$. 

Proceedings of the 10th WSEAS International Conference on COMPUTERS, Vouliagmeni, Athens, Greece, July 13-15, 2006 (pp1111-1118)
The weighted graph for the map in Fig. 4 and the attraction function (9) and repulsion function (10) is shown in Fig. 5.

### 3.2 Collision detection

The collision detection [19] sub-module is used by both the simulation sub-module – to test the appearance of possible to happen collisions – and the monitor module - asks for the occurrence of already happened collisions.

No matter how good a path planning algorithm is, it can not guarantee collision avoidance in a dynamic environment. A fast enough obstacle will always be able to catch up and hit the robot.

If after the map analysis it becomes clear that the target is unreachable, this is signaled to the monitor which stops the robot with an error message.

### 3.3 Simulation

The simulation sub-module tests the elaborated path finding algorithm taking into consideration the information received from the environment. If the environment is static, this simulation is relatively simple, while if it is dynamic it is necessary to use the information about its dynamics. This information is provided by the kinematics computing sub-module. The visual simulation is realized using the Microsoft DirectX libraries [1][13].

### 3.4 Kinematics computing

The kinematics computing sub-module computes and keeps the speed and the direction of movement for each obstacle. This module estimates by comparing each received map with the previous one a direction and computes a speed for each obstacle. This information is essential (determinant) in selecting the path to follow by the path finding algorithm. A set of wrong information provided by the kinematics computing sub-module can generate an imminent collision that would not be detected by the simulation sub-module.

If the generated algorithm passes the simulation test, the path planning module generates a path plan based on which the robot moves. At each new generated map, the kinematics computing sub-module re-computes the speeds and directions for each obstacle. If no change occurred, the other sub-modules of the path planning module remain in stand by and the robot is controlled based on the previously elaborated path plan.

### 4 Monitor Module

The monitor module inspects the state of the robot (did it collide with an obstacle or not, did it reach the target, is the target unreachable). This module is linked to all the other modules through communication channels. It is informed by the other modules of problems and malfunctions occurrences and it can temporarily suspend or even stop the activity of the servo-controller in the event of such an occurrence [8].

If changes appear between the new map and the previous one, this is signaled to the monitor module, which temporarily halts the path plan execution. The simulation sub-module tests the current algorithm with the new kinematics data. If the algorithm is still viable, it is signaled to the monitor to permit the restart of the path plan execution. The monitor communicates to the visual servoing module [17] to continue motion according to the path plan. Else, the path finding sub-module will generate a new algorithm.

If the created algorithm does not pass the simulation test, the best possible path is chosen, that is, the one having the least number of collisions and the most far away ones, hoping that some of the obstacles might change their direction or speed, such that to allow the target reach. Also the monitor is signaled about the current status and an alarm message is sent.
The monitor module tests at each step for collision detection and target reach. If a collision takes place, it stops the robot with an error message. If the target is reached, the robot is stopped with an OK message.

If any of the modules encounters an error in execution, it communicates this to the monitor which, in turn, signals the visual servoing module to temporarily cease robot movement.

5 Conclusions

A general structure for a software system dedicated to visual servoing applications was developed. The project is financed by Romanian National University Research Council CNCSIS.

Some of the software modules were already implemented: image processing module – image processing techniques for visual servoing applications, map generation. Fuzzy algorithms and methods were used in order to support the implementation of the image processing procedures for both image enhancement and image segmentation.

Practical applications concerning image processing may encounter problems that are vague or ambiguous or need simultaneous approaches in solving. This paper presents some fuzzy algorithms as an alternative solution to this kind of problems.

The paper is focused on the presentation of the image processing module and path finding submodule.

As it is structured on “specialized” functions (small parts that compute specific tasks), the implementation of these algorithms makes the adding of further functions (algorithms) easier, the new functions wouldn’t need to be fully implemented because they make use of some parts of those algorithms that were already implemented, only the specific part of the new tool having to be created at that time. This way, one of the five principles that stood at the elaboration of this architecture is achieved – additivity.

Each specialized set of functions and procedures is grouped together and no other function does not interfere in their internal activity. So the architecture is modular.

The modules work at the same time, each using the output of another at a previous stage. This means the architecture supports parallelism.

The functions are implemented in C which is one of the most platform independent programming languages. It follows that the architecture is portable.

The path planning module includes both a path finding sub-module that will contribute to the plan to reach a target – long term goal, and a collision detection sub-module, to help avoid hitting obstacles – short term goal. Having this, the architecture supports multiple goals.

Further development of the project would be to add new tools and improving the already implemented ones by adding new rules to the FIRE systems or modifying the weights computing functions or to optimize the path finding algorithm.

The used path finding algorithm resembles in a small way to the A* algorithm. It does not necessarily offer the shortest path solution, but it is much faster, fact that is very important in a real-time control system.

As a general concept, the job of the cluster analyze is the data partition into a number of groups, or clusters. Applying this partitioning operation on images, the image segmentation - a very important task for image processing - is obtained. This paper presents a fuzzy segmentation algorithm, fuzzy c-means, applied to grayscale images.

Image processing field, and even more, fuzzy image processing, is a very large field, and for best results, user’s experience is essential. One algorithm has not the same performance on all types of image and the chosen parameters could have a good or bad influence on the results. So the algorithm should be personalized on the image characteristics. This is achieved by the use of a set of parameterized functions. The user’s experience will guide him to select the appropriate values of the parameters for each image processing algorithm that is applied, different values for different types of images, depending on the image that is being processed.

The design of these algorithms was intended as another proof, although not necessary, that the fuzzy approach is an important alternative way to consider when developing an image processing application.

This paper presents a solution for a module of path planning adapted to robotic applications. The path planning module creates an algorithm for path finding based on the map generated by the image processing module, adapts it to the changes that appear in the environment, tests for collisions and finally simulates the elaborated algorithm. It outputs a control path plan.

No matter how good a path planning algorithm is, it can not guarantee collision avoidance in a dynamic environment. A fast enough obstacle will always be able to catch up and hit the robot. The path planning algorithm will always offer the best solution possible, but that solution might not be able to help the robot
reach the target because of collisions or because the
target is unreachable.

A monitor module is also implemented in order to
survey and coordinate the activity of the other
modules. It is in contact with all the modules through
communication channels. It can stop at any time the
servo controller [24].

From the general application point of view, the
other modules of the system must be implemented.
Regarding robot control, the research team designed
and implemented a wide variety of control low, both
for usual robotic structure and for hyper redundant
ones.

References:
[1] Adams, A., Advanced animation with
DirectX, The Premier Press Game Development
Publ., 2003.
planning: a distributed representation approach, The
[4] Bezdek, J. C., Pattern Recognition with
Fuzzy Objective Function Algorithms, Plenum Press,
1981, New York
System for a Mobile Robot, IEEE Journal on
Enhancement Based On Fuzzy Logic, Proceedings of
the IEEE International Conference on Image
Processing (ICIP ’95), 1995
Processing: Introduction in Therory and Practice,
[8] Ivanescu, M., Bazdoaca, N., Cojocaru, D., A
Fuzzy Controller for Tentacle Cooperative Robots,
ASME International Symposium on Advances in
Robot Dynamics and Control, New York, 2001,
IMECE2001-DSC-24631.
[9] Ivănescu, M., Cojocaru, D., Popescu, N.,
Popescu, D., Tănăsie, R.T., Visual Based Control
System for a Tentacle Manipulator, WSEAS
Transactions on Information Science and
[10] Ivănescu, M., Position Dynamic Control for
a Tentacle Manipulator, Proceedings IEEE
International Conf. on Robotics and Automation,
[12] Ivănescu, M., Sisteme de conducere fuzzy,
Mircea Ivănescu Şcoala de Robotici din Craiova,
process and its use detecting compact well-separated
preprocessing of configuration space for fast path
planning, IEEE Int. Conference on Robotics and
Techniques in Image Processing, Springer Verlag,
2000.
[17] Khatib, O., Real-time Obstacle Avoidance
for Manipulators and Mobile Robots, International
[18] Krishnapuram, R., Handbook of Fuzzy
[19] Lin, M.C., Efficient Collision Detection for
Animation and Robotics, PhD thesis, Department of
Electrical Engineering and Computer Science,
University of California, Berkeley, 1993.
[20] Marchand, E., Spindler, F., Chaumette, F., A
generic software platform for visual servoing,
[21] Umbaugh, S.E., Computer Vision, and Image
[22] Ville, D. Van der, Nachtegaele, M., Weken,
D. Van der., Kerre, E. E., Philips and W., Lemahieu,
I., Noise Reduction by Fuzzy Image Filtering, IEEE
Transactions On Fuzzy Systems, Vol. 11, No. 4,
2003.
[23] Wang, L. X., A Course in Fuzzy Systems and
[24] Yen J. and Langari R., Fuzzy logic,
intelligence, control and information, Prentice Hall,