VISION-BASED TRACKING SYSTEM FOR THE NOMAD XR4000 MOBILE ROBOT

Sérgio Roberto Gonsalves Tourino
Universidade de Brasilia, Departmento de Engenharia Mecânica
70910-900 - Brasília, DF, Brazil.
tourino@graco.unb.br

José Maurício S. T. Motta
Universidade de Brasilia, Departmento de Engenharia Mecânica
70910-900 - Brasília, DF, Brazil.
jmmotta@unb.br

Abstract. *Recent advances in computing processing power brought about the use of machine vision to new applications, like industrial inspection and robotic sensors. This project has as objective the development of a vision-based tracking system to the Nomad XR4000 mobile robot. The system is composed by four main modules: the acquisition module, using a Meteor frame grabber under Linux getting RGB format images; the pre-processing module, processing the acquired image by sampling reduction, low pass filtering, RGB-HSV color conversion and image's gradient calculation; the motion-extraction module, extracting from a gradient image a number of feature points from the scene, comparing them with previous images using SSD correlation and optical motion detection and also calculates the centroid and scattering data for those points; the control module, using the data computed by the previous module and by means of a fuzzy controller calculates the robot's response making it tracks the perceived image motion. The system was optimized in processing time by computational code changing. Also, an image correlation parameter optimization was carried out based on factorial design methodology. It was verified that two variables, the correlation window size and the search window size are the main variables related to the processing time and robustness of correlation algorithm. Tests were performed with the system using a robot simulator, validated qualitatively by comparing the speed curve observed from the simulator and from the Nomad robot. It was observed that the image processing system and the fuzzy logic controller were adequate to cancel the apparent motion in the image plane from the acquired images. The developed system could detect speeds by means of optical flow and control the robot using a fuzzy controller, and brought about a contribution with the study of important system variables by factorial design analysis.*

Keywords. *Machine vision, tracking system, optical flow, mobile robotics.*

1. INTRODUCTION

Visual Servoing of mobile robots is a thriving approach to the control of robot navigation since it emulates human sense of vision. The best control of a mobile robot would be achieved by constructing a complete three-dimensional world model, planning a path and then executing the required steps to move the robot along the path. However, many challenges are still posed ahead as object recognition, obstacle avoidance (Trucco, 1998, Klette, 1998) and sensor fusion.

Robot navigation based on visual tracking has had a significant amount of research work (Corke, 2000, Kara, 2000) in the last years. A robust robot tracking system can provide information about the relative 3-D motion of a target relative to the observer and may simplify the retrieval of a object shape and/or localization.

Several researchers have published different approaches to the problem of visual servoing of robots and many methods have been presented for tracking a moving object over a sequence of images, based on optical flow, image correlation or deformable contours (Santos-Victor, 1998, Kass, 1998).

Arsenio and Santos Victor (1997) addressed the problem of tracking a moving target by a monocular observer. Their strategy was based on the integration of correlation-based techniques together with active contours, using a Kalman filtering approach to update the target image position over time. A number of experiments have been presented to compare the robustness of each method.

Spindler (1998) presented an estimation of the apparent bi-dimensional motion induced in the image sequence by a subsea vehicle in order to compensate for it, using an affine model and a gradient-based image optic flow. Their model was computationally optimized by using methods to reduce the computational cost.

This article presents the basics of a vision-based tracking system developed for the Nomad XR4000 mobile robot (Fig. 1) architecture and an optimization scheme based on experimental statistics, namely factorial design analysis (Box, 1978, Rowlands, 1995, Hafeez, 2002), to investigate sensitive parameters of the image-based tracking system. Factorial Design is an experimental design which allows system parameters to be investigated in all possible combinations of their levels or values and the results are obtained by analysis of variance. Experimental tests were carried out using a robot simulator, which was validated by comparing the speed curve obtained from the simulator and the physical mobile robot using a single camera. A fuzzy logic controller was used to cancel the apparent motion from the acquired images.

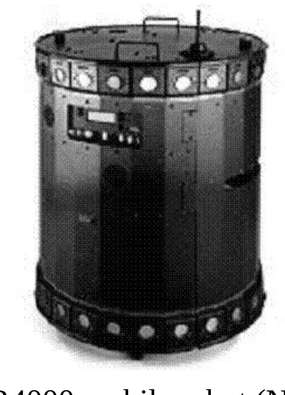


Figure 1: Nomad XR4000 mobile robot (Nomadic Technologies)

2. OPTICAL FLOW

The analysis of motion in images is performed by using several types of algorithms. The use of the optical flow approach has a better acceptation in research (Beauchemin, 1995), mainly due to its capacity to cope with motion with low computer processing time and/or high precision motion calculation.

Optical flow is an approximation of the real 3D motion in image, measured in the image plane. It's based on the assumption that the intensity structures of local regions in time-varying images are approximately constant under short duration motion, commonly named as the optical flow constraint equation, mathematically stated as [Anandan, 1992]:

$$\nabla I \, . v + I_t = 0 \qquad (1)$$

where ∇I is the spatial intensity gradient, v is the image velocity and I_t is the temporal intensity gradient.

There are many methods to calculate the optical flow. The three mainly used methods are the gradient based, the frequency based and the correlation based algorithms. Gradient and frequency methods are in general better for getting precise velocity information but at the expense of high processing time. The latter, correlation method, is better suited to real time problems, like vision based tracking systems. In spite of correlation is based on statistical methods, the SSD (sum of squared differences) approach is more used in optical flow calculation (Anandan, 1992):

$$S(x,d)=\sum_{j=-n}^{n}\sum_{i=-n}^{n} W(i,j)\left[I\left(x+(i,j),t\right)-I\left(x+(i,j),t+1\right)\right]^2 \tag{2}$$

where $W(i,j)$ is a weighting function, $I(x,t)$ is the image intensity in space coordinate x at time t , and d is a square neighboring area near x with size $(2n+1)^2$. Regions with smallest S values have better correspondence, indicating image displacement related with image motion.

3. VISION-BASED CONTROL

Machine vision based control is not a new technology. The work of Hutchinson (1995) reveals that there are mainly two architectures to cope with this subject: the first considers the errors in the image domain and the second in the position domain. These errors are the input to the control loop. It was concluded that data based on errors in the image domain are in some cases better than with second approach, since the first method does not need camera calibration. In another work, Plakas (1998) showed that for underwater applications uncalibrated systems are more robust than calibrated ones. His work uses a SSD correlation algorithm to make 3D reconstruction from the acquired images.

Many control algorithms can be used in a vision based control system. If possible, a PID controller is a good choice, since it can cope easily with linear problems. Besides, there are many tuning rules to this type of control system. Some applications in vision based control are non-linear, and PID controllers are not useful in these cases. To solve this problem other types of algorithms (artificial neural networks or fuzzy logic controllers) are used. Neural networks are used in the control of car wheels based on road images, for example.

4. SYSTEM ARCHITECTURE

The architecture of the vision control system was divided in four modules for acquisition, pre-processing, motion calculation and controlling. Figure (2) shows this architecture.

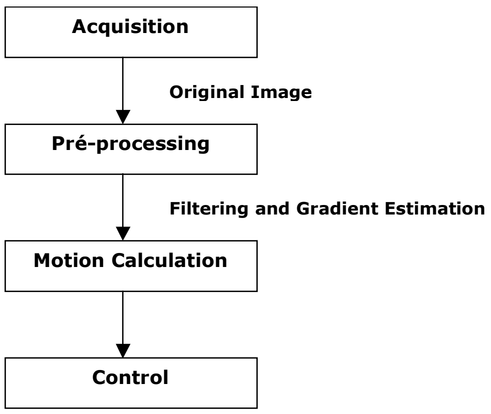


Figure 2: System architecture for object tracking.

4.1. Acquisition Module

The acquisition module is responsible for sending commands to the video acquisition device (framegrabber) in the robot. This module initializes the robot's communication system and configures the framegrabber.

Robot's initialization starts the framegrabber configuring its parameters: a) image's format (NTSC), b) image's resolution (256x256), c) depth color (24 bits), d) acquisition mode (continuous sequence), e) frame buffer memory address, and f) interruption signals in Unix style. The frame buffer memory addresses points to a global variable in the program, responsible to store the three components of image (Red, Green and Blue).

4.2. Pre-processing Module

The pre-processing module prepares the image to be used by the next module, motion extraction. There are five steps in pre-processing:

1. image decimation and filtering;
2. color conversion;
3. image's gradient;
4. histogram thresholding;
5. search of feature points in image.

The first step is carried out by using a one-step algorithm reducing simultaneously the image dimension and applying a low pass filter over the image. The new pixel is the mean value from four pixels in the original image as seen in Fig.(3). The algorithm uses only 25% of processing time compared a common two-step method.

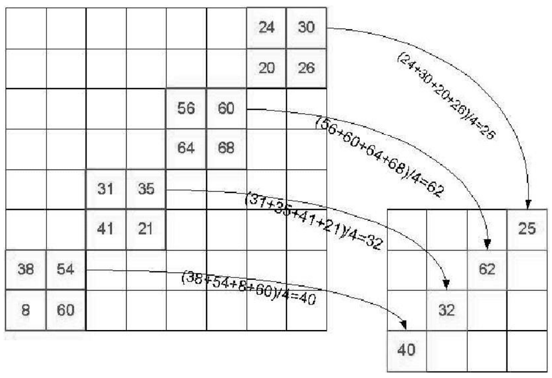


Figure 3. One pass decimation and filtering.

The color conversion step is necessary since most of the algorithms for image processing are devised for using monochromatic images for simplicity. The conversion from the RGB (Red-Green-Blue) to HSV (Hue, Saturation, Value) produces a monochromatic output and reduces storage space in memory as well as the processing time for each image. The mathematical formulation for this conversion is as simple as [Tourino, 2001):

$$V = \frac{R + G + B}{3} \tag{3}$$

where R, G and B are the components of original image (Red, Green and Blue, respectively), and V is the Value component in the HSV format.

The gradient step searches for discontinuities in the image, like borders and corners in objects, or indicates good points to be tracked in the motion analysis system. The algorithm used is the Sobel mask, mainly due to its simplicity and fast processing time. The calculated gradient has two components, *x* and *y*, that are normalized and merged to give only one gradient image as result.

The next step, histogram thresholding, is a procedure applied over the gradient image. The thresholding is carried out by using points with high gradient values. The cutoff value is obtained from the histogram, based on a percentage of the total points in image. Figure (4) presents an example of this process, in which 189 is the cutoff value.

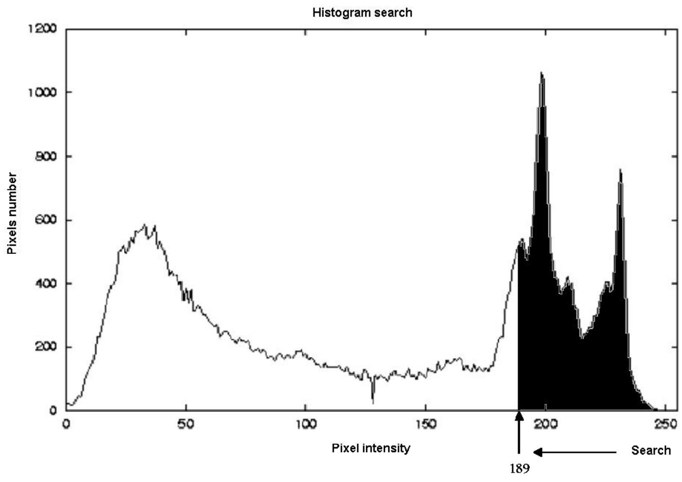


Figure 4. Histogram thresholding of a gradient image.

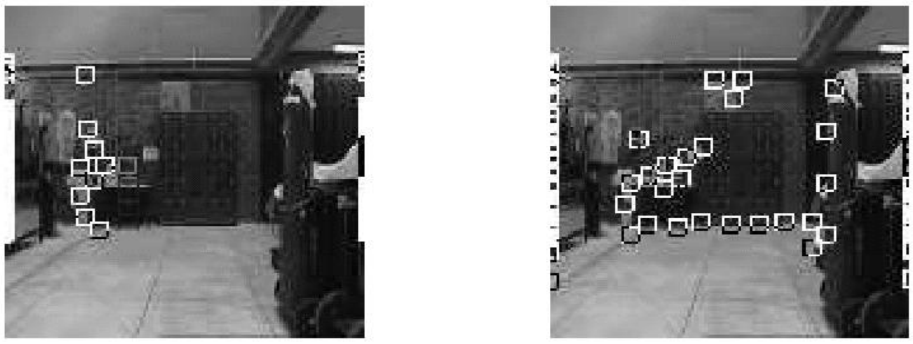


Figure 5. (a) Non-uniform and (b) uniform searching for feature points.

Searching for feature points in the image is performed by selecting a limited number of points which are present in the thresholded gradient image. A uniform search algorithm was conceived to get a better point distribution in the image space, increasing the probability of finding feature points representing the actual object's motion. Figure (5) shows the results using this approach.

4.3. Motion Calculation Module

This module calculates motion by processing data from the control module. Optical flow is calculated by using the correlation method, in which the previous image is compared to the current one to get the displacement of the feature points or the apparent object's motion.

The extraction module has the following steps:

1. extraction of the last and previous sub-image;
2. displacement calculation from correlation;
3. updating feature points data;
4. centroid and scattering data calculation from feature points.

Sub-image extraction is performed storing a square window of the original image centered in each feature point. This new image is compared to the previous stored sub-image and the correlation algorithm is applied over it. When the correlation is maximum the new location of a feature point is marked as well as velocity.

The correlation algorithm is based on the SSD. In this work the following modified expression was used to calculate correlation:

$$S(x_c, y_c) = \sum_{x=-l}^{l} \sum_{y=-l}^{l} [I(x+i, y+j) - J(x+x_c, y+y_c)]^2 \tag{4}$$

where *I* and *J* are respectively the last and the previous images, (x_c, y_c) is the central point in correlation window, (x, y) are the index of the correlation window, (i, j) is the localization of the point in the correlation window, and *l* is the correlation window size.

When the maximum correlation point is found, the system can calculate feature point displacements. The data stored in the system is then updated with new positions and velocity for the next iterations.

Subsequently the system starts to calculate the centroid and scattering information from the feature points. Centroid data is associated with the mean position of points, and scattering data is the mean distance of points relative to its centroid. Scattering values were defined as dispersion around the average, as:

$$x_e = \sum_{i=0}^{N} |\bar{x} - x_i| \qquad y_e = \sum_{i=0}^{N} |\bar{y} - y_i| \tag{5}$$

The vectors in Eq. (5) are used in the control module to perceive object's motion. With this information the robot motion can be controlled. A simple algorithm is used in this approach: centroid data is associated with lateral movement of an object, so if the centroid value increases it turns that the robot must walk to right. If it decreases, the robot must move to the left. The same process is used with the scattering data: if points in a group are getting closer (scattering value decreases) this means that the object is getting farther from the camera, so the robot must walk forward. On the other hand, if the points are getting farther the robot must walk backward.

Table (1) presents centroid and scattering data in *x* and *z* direction. It is shown that during the motion along the *x* axis the scattering values are almost constant, while the centroid value, x_c, increases. Similar results are found along the *z* motion, when centroid values remain practically stable and scattering data decreases.

Table 1. Centroid and scattering data from *x* direction motion (a) and *z* direction motion (b).

t	***xc***	***yc***	***xe***	***ye***
1	67	64	188	405
2	71	63	205	398
3	75	64	218	403
4	80	64	235	397
5	84	63	251	395

(a)

t	***xc***	***yc***	***xe***	***Ye***
1	42	74	29	21
2	42	76	17	20
3	43	75	17	24
4	44	75	15	21
5	45	74	15	21

(b)

4.4. Control Module

This module is responsible for getting information from the previous modules and taking action for the robot motion. A fuzzy controller was developed to compose the control module. Its main characteristics are:

- three variables: two inputs (group of points displacement and its velocity) and one output (robot's walking distance);
- each variable has seven fuzzy sets, with triangular or trapezoidal membership functions;
- fuzzy rules were defined by simple expressions like “if displacement is medium and speed is slow then walk is medium”;
- defuzzyfication method was centroid based;
- system was implemented using the Xfuzzy software (Inse, 2000).

A snapshot from the Xfuzzy software and results from the fuzzy controller are shown in Fig. (6) in a surface response graph.

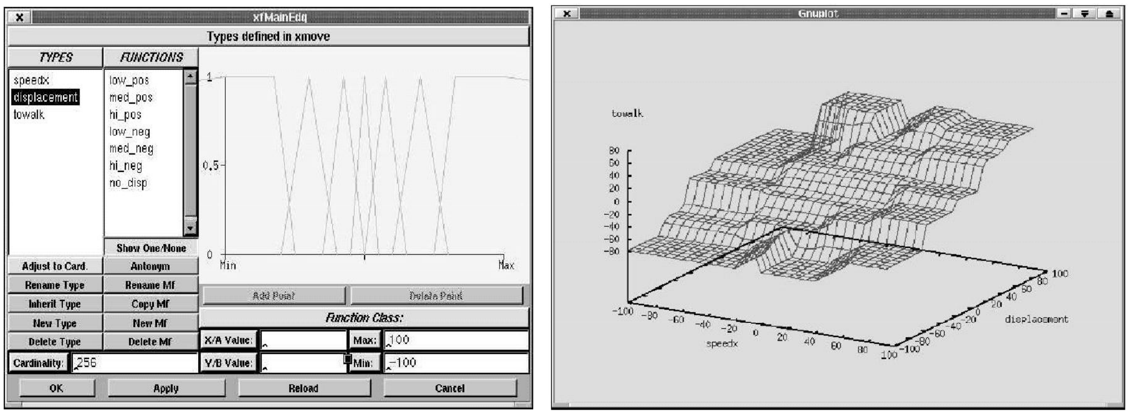


Figure 6. Snapshots from Xfuzzy software and the fuzzy controller´s surface response.

5. SYSTEM OPTIMIZATION

The developed system had its code optimized and the sensitivity of its variables was studied by means of factorial design analysis. The final optimized system was computer simulated for getting more information about its efficiency and accuracy.

6.1. Code Optimization

Code optimization was carried out along the computer program sections that are extensively used, like the correlation algorithm code. In a normal image the correlation code is repeated about 32,000 times, so a small gain in processing time in this part reduces considerably the total processing time. The optimization implemented in this work was as in the following lines:

g = img[i][j]+img[i+1][j]*img[i][j]; changed to t=img[i][j]; and g=t+img[i+1][j]*t;

Despite of this new code uses more memory the processing time is shorter, since it refers to the memory address in the variable *img* only once. This simple code substitution could reduce the processing time around 7.5% for each image.

6.2. Optimization by Factorial Design and Results

Factorial Design is a technique to design an experiment or to estimate how much input parameter changes may affect the system output. Before conceiving the experiment design and applying a performance test, it is necessary to consider the amount of time spent to perform experimental runs. Time consumption is closely related to the number of experimental parameters, the number of levels of each parameter investigated, and the amount of data required. In factorial design, the total number of runs (N) is determined using the expression $N = (L)^V$, where L is the number of levels of each parameter and V is the number of experimental parameters investigated. As an example, when studying four parameters with three levels each, the total number of runs is $3^4 = 81$. (Box, 1978)

To achieve a generic factorial design, one selects a fixed number of levels (or versions) for each of the parameters (or factors) and carries out the experiments with all possible combinations. A level of a parameter refers to the discrete values of that parameter domain. For example, if an experiment is such that the studied temperature has the values of 20°C, 50°C and 100°C then the experiment has 3 levels associated to the variable temperature (parameter).

Factorial design possesses an important faculty of showing up the interaction between parameters, but this does not mean that these interactions are numerically appreciable. The main effects of a parameter alone tend to be larger than the effects of interactions with two parameters, and those larger than three, and so on.

The number of runs or experiments performed in a factorial design 2^n increases geometrically as n increases. However, in many cases the information desired from the experiments can be obtained with only a part or fraction of the total runs, leading to the concept of fractional factorial design.

Fractional factorial design disregards smaller effects from interactions of higher orders to reduce the number of experiments. Various fractions can be used to reduce of the number of experiments, as 1/2, 1/4, 1/8 or 1/16. In this case one can define a fractional factorial design on the basis of the original exponent minus the fraction factor. However the greater the fractioning the less confidence there will be in the results. (Hafeez, 2002)

Five parameters were tested in the system. Table (2) shows the main parameters in study.

The output data analyzed included four variables: t_1, processing time for the first image; t_f, processing time for the following images; e_x, error in the x component of the optical flow; and e_y, error in the y component of the optical flow.

Table 2. Meaning, maximum and minimum values for parameters in factorial design analysis.

#	Parameter	Meaning in system	Minimum	Maximum
			(-)	(+)
1	ROWS_SIZE	Correlation window size	5	15
2	SEARCH_SIZE	Search window size	10	30
3	THRES_PERCENT	Threshold for feature points	0,1	5
4	MAX_FEAT_NUM	Maximum number of feature points	50	200
5	UNGROUP_SIZE	Distance to erase nearby points	5	20

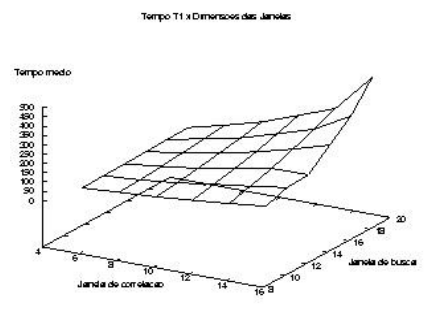
It was used a 3 x 2^{5-1} fractional factorial design with repetitions, leading to 48 runs. Repetitions were necessary to calculate the average processing time for each test. Fractional design was used to reduce the number of runs.

Factorial design results are summarized in Tab. (3), where numbers are related to the parameters shown in Tab. (2).

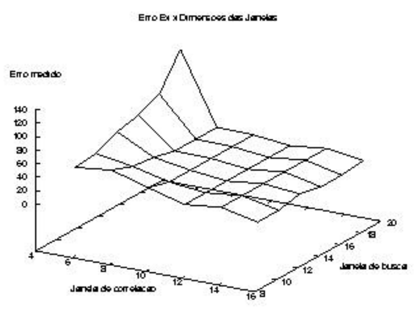
Table 3. Results from factorial design runs.

Output variable	Inputs related to this output
T_1	1, 12, 3 and 4
t_f	1, 2 and 12
e_x	1, 2 and 12
e_y	1, 2, 12 and 4

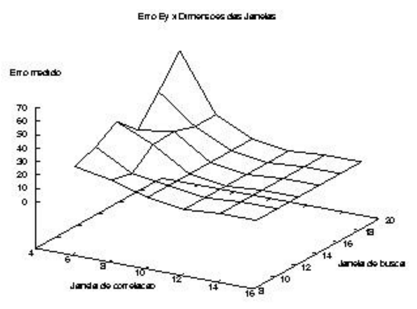
In Table (3), the number 12 means that the output variable is dependent on the interaction between parameters 1 and 2. The results show that parameters 1 and 2 (ROWS_SIZE and SEARCH_SIZE, respectively) are the main factors in the output results.



(a)



(b)



(c)

Figure 7. Sensitivity analysis graphs in optimization phase: (a) time versus windows' dimensions; (b) e_x error versus windows' dimensions; (c) e_y error versus windows' dimensions.

A sensitivity analysis on the variables 1 and 2 was performed for obtaining optimal values for processing time and the minimum correlation error. Figure (7) shows graphs as studied in this optimization phase. From this study the 'optimal' values were arisen for the following variables: ROWS_SIZE (=9) and SEARCH_SIZE (=17). Those values were chosen from the graphs due to their respective short processing time and relatively low correlation error.

6.3. System Simulation

A simulator for the Nomad robot was conceived to validate the system's performance. The simulator emulates the robot's motion in the x and z directions relative to the camera coordinate system, providing artificial images to the tracking algorithm. The simulator's performance was verified by comparing simulated speed curves to actual robot data under the same conditions. This procedure proved to be acceptable to validate the emulation of the robot's motion.

The simulator emulates all the tracking modules. The last three modules are used identically as tracking algorithms or simulating ones, but the image acquisition module (first) is changed to an artificial image generator when used in simulation. Synthetic images are built using white squares over random grayish background. The square positions in the image are calculated from the object and robot positions in the 3D world, using a perspective projection.

The simulator was tested in the x and z directions of the object's motion. The first run was performed with the tracking controller turned on, and the second run with controller turned off. Results were compared based on the tracking algorithm performance.

7. DISCUSSION

Simulations and tests have shown that the tracking algorithm is capable to follow the object's motion. The use of centroid and scattering data showed to be simple and efficient to characterize the motion of groups of points (seen as points in same object). Control using fuzzy logic was not able for tracking along the z direction of the object's motion. This can be explained due to the fact that the controller surface response is not smooth in some regions, leading to instabilities in the control loop. The presence of a speed control algorithm in the robot server, Robserver, may also interfere in the fuzzy controller performance.

8. CONCLUSIONS

The development of a tracking system for the Nomad robot have led to the following results: optical flow calculations based on a correlation method showed efficiency and short processing time; the fuzzy controller in the *x* direction was successful, and tracking in the *z* direction can be improved by using a separate controller. Factorial design and code optimization were useful for reducing the processing time and the errors of velocity measurements. Future improvements to this work can be pointed towards the use of a better control loop, an integrated optimization step (image processing and control optimization) and the use of a non-linear model for searching feature points in the correlation algorithm. A procedure based on variance analysis would automated the choice of optimal parameter values.

9. REFERENCES

Anandan, P. Bergen, J. R., 1992, "Hierarchical Model-Based Motion Estimation", Proceedings Of The European Conference On Computer Vision. 1992.

Arsénio, A. and Santos-Victor, J., 1997, "Robust Visual Tracking by na Active Observer", Proceedings of the IROS97, pp. 1342-1347.

Beauchemin, S. S. Barron, J. L., 1995, The Computation Of Optical Flow. University Of Western Ontario. Acm Computing Surveys, Vol. 27, N^o 3.

Box, E. P.; Hunter, W.C., 1978, Statistics for Experimenters: An introduction to design, data analysis and model building, New York, USA.

Corke, P, Hutchinson, S. A., 2000, "Real-Time Vision, Tracking and Control", Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco-CLA, USA.

Hafeez, K.; Rowlands, H.; Kanji, G.; and Iqbal, S., 2002, "Design optimization using ANOVA", Journal of Applied Statistics, Vol. 29, N. 6, pp. 895-906.

Hutchinson, S. Hager, G. Corke, P. ^{a}, 1996, "Tutorial On Visual Servo Control", University Of Illinois At Urbana-Champaign. IEEE Trans. Robotics And Automation, Vol. 12, N^o 5, Pp. 651-670.

Inse, Instituto De Microeletrônica De Sevilha. Xfuzzy., 2000, [Online] Available on Internet through WWW. Url: http://www.inse.cnm.es/xfuzzy. Sevilha, Espanha, 2000.

Johnsson, M. Wiberg, P. Wickström, N., 1997, "Vision-Based Low-Level Navigation Using A Feed-Forward Neural Network", Centre For Computer Systems Architecture. Halmstad University. Halmstad. Sweden. Proc. International Workshop On Mechatronical Computer Systems For Perception And Action. Mcpa'97. Pisa. Italy. Feb. 10-12.

Kara, R, Wira, P., Kihl, H., 2000, "Robot Vision Tracking with with a Hierarchical CMAC Controller", Fourth International Conference on knowledge-Based Intelligent Engineering Systems & Allied Technologies, Brighton, UK.

Kass, M, Witkins, A. and Terzopoulos, D, 1988, "SNAKES: Active Contour Models", IJCV, Vol. 1, N. 4.

Klette, R., Schlüns, K., Koschan, A., 1998, Computer Vision: Three-Dimensional Data from Images, Springer-Verlag, Singapore.

Plakas, K. Trucco, E. Fusiello, A., 1998, "Uncalibrated Vision For 3d Underwater Applications", Proc. Ieee Oceans'98. pp. 272-276. Nice, France.

Rowlands H. and Pham D. T., 1995, "Application of the Taguchi method to the design of a robot sensor", Robotica, Vol. 13, pp. 607-617, Cambridge University Press.

Santos-Victor, J. Sentieiro, J., 1994, "The Role Of Vision For Underwater Vehicles", IEEE International Symposium On Autonomous Underwater Vehicle Technology – Auvt94. Boston, Usa.

Spindler, F., Bouthemy, P.,1998, " Real-time estimation of dominant motion in underwater video images for dynamic positioning", Proceedings of the 1998 IEEE International Conference on Robotics & Automation, Louvain, Belgium, pp. 1063-1068.

Tourino, S. R. G., Álvares, A. J., 2001, "Tele-Fuzzy Control of A Mobile Robot", In: International Conference On Cad/Cam, Robotics & Factories Of The Future, Cars&Fof 2001. Durban, South Africa.

Trucco, E. & Verri, A., 1998, Introductory Techniques for 3-D Computer Vision, Prentice-Hall, New Jersey, USA.