PROGRAMMING A ROBOT MANIPULATOR TO FOLLOW TRAJECTORY CAPTURED BY A VISION SYSTEM

Giovanni J. Rosa
Universidade de Taubaté, Rua Daniel Danelli, s/n – Campus da Juta – Taubaté – S.P. – 12080-000
giovanni@lgphilips-displays.com

Álvaro M. S. Soares
Universidade de Taubaté, Rua Daniel Danelli, s/n - Campus da Juta
12060-440 - Taubaté- S.P.
alvaro@mec.unitau.br

Abstract. — *This work presents the implementation of a system that drives a robotic manipulator using a vision system. The trajectory is marked randomly by the user in a sheet of paper. The trajectory image is captured by a vision system composed by a CCD camera and a frame grabber board and converted into a robot trajectory. A computer program was developed to transform the image information into the robot manipulator joint space and drive the robot according to the trajectory. The results of the experimental implementation have shown an error of 5% in the trajectory marked by the manipulator and also a robot repeatability of 3%. Some trajectory limitations should be respected because of the robot configuration and computer program. These limitations are such as no crossing points, maximum proximity of 3 mm between trajectory segments, and maximum interruption of 5 mm in a range of 100 mm. The system works in a two-dimensional environment due to vision system limitation.*

Keywords*. Robotics; Vision; Trajectory; Camera; Image.*

1. Introduction

The movement of a robot is usually controlled by a software, which sets trajectory, stopping points, speed, acceleration etc. The aim of this work is to acquire a trajectory marked in a sheet of paper by a user, using a vision system and have a robotic manipulator follow this trajectory. Benefits are that the trajectory does not have to be empirically obtained, diminishing the adjusting time of a manipulator for a new trajectory.

Fig. 1 shows the complete system to capture the trajectory and convert into manipulator joint spaces, driving the robot according to the trajectory.

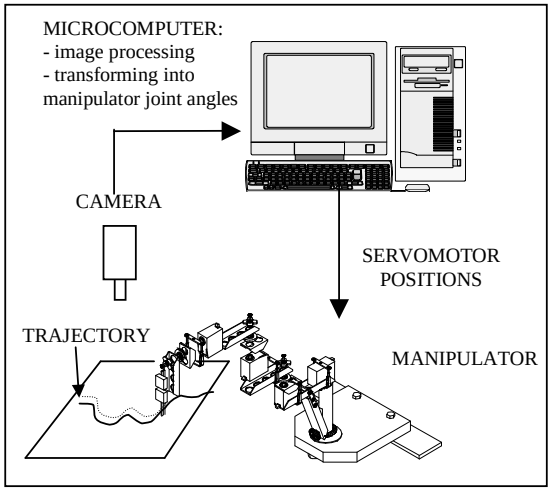

Figure 1. Complete system to capture the trajectory and convert into manipulator movement

2. Implementation of the system

2.1. Image Processing

A CCD (charge-coupled device) camera is utilized to capture the image. A frame grabber transforms the image information from the camera into computer memory. The image is stored in computer memory as a matrix of different levels of gray with 210x256 pixels, it has to be processed, so the trajectory can be sorted out from the background. Three processes are done: edge detection, thresholding and point sequencing. These processes are explained as follows.

2.1.1. Edge Detection

An edge filter is applied to the image. The principle of the filter is explained in the Fig. 2.

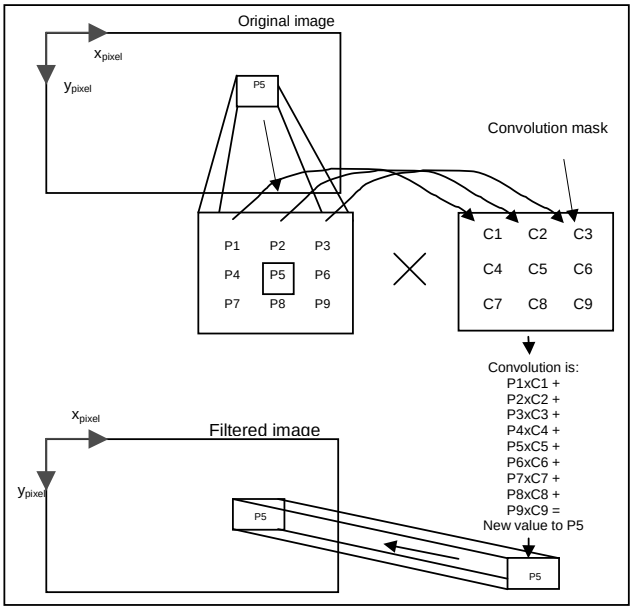

Figure 2. Convolution of an image with a filter

As it is illustrated in Fig. 2, the image is processed with a filter by applying a convolution. There are different edge filters, the principle is to derivate the image. Lindley (1991) and Myler (1993) bring the explanation for different edge filters. The one used in this work is Prewitt. The convolution mask is expressed as follows:

$$g_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ (mask for x direction)}$$

$$g_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ (mask for y direction)}$$

The point P5 in Fig. 2 will have the following value after the filter is applied:

$$P5 = \sqrt{g_x^2 + g_y^2} \qquad (1)$$

Besides that, an edge determination will actually cause 2 edges, first from the transition background-trajectory, second from the transition trajectory-background.

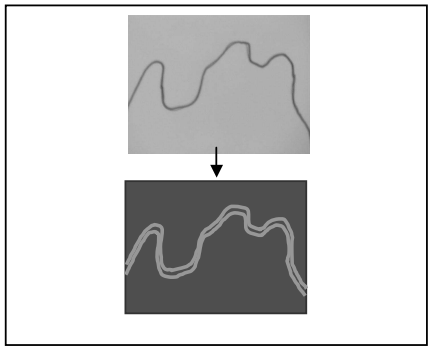

Figure 3. Prewitt edge detector of an image

As Fig. 3 shows, the filter emphasizes the edges, but creates 2 trajectories. As only one trajectory is needed, a condition is implemented in the algorithm. The condition considers the detection valid only if both g_x and g_y are positive.

If $g_x>0$ and $g_y>0$ then $P5 = \sqrt{g_x^2 + g_y^2}$ (2)

The result of equation (1) and (2) are shown in Fig. 4.

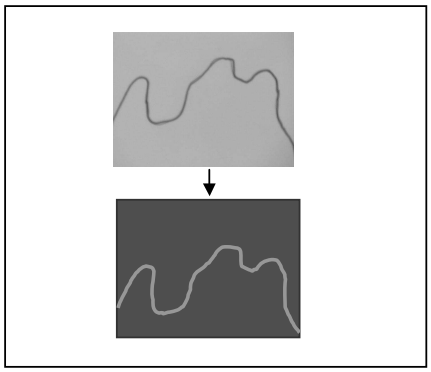

Figure 4. Prewitt edge detector with condition for positive values.

2.1.2 Thresholding

After detection the edge, it is necessary to binarize the image in order to separate the object from the background. The histogram of gray level of a picture has peak for the object and for the background as shown in Fig. 4.

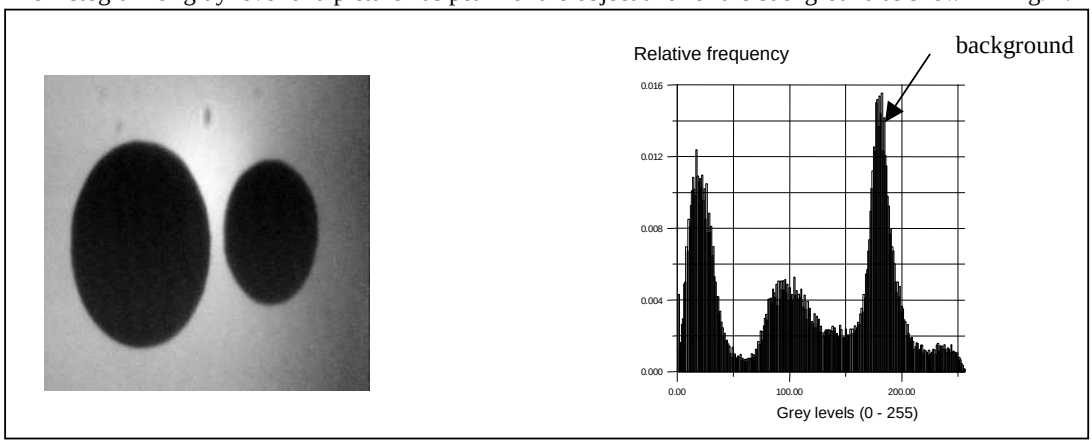

Figure 5. Example of an image and its histogram

Binarization is a technique that leads to only 2 levels of gray – black or white and separate the object and background. It is done by calculating the threshold, value of gray level that determines the edge between the object and background. Gray levels of the image above will be considered white (background) and below threshold black (object).

Parker (1997) has a proposal to calculate threshold. The formula takes into account the histogram peak and calculates:

Max $[(k - j)^2 h(k)]$ $(0 \le k \le 255)$ (3)

h – histogram with 256 gray levels,

j – position of histogram peak

k – each position of the histogram, range 0 to 255.

Equation (3) has the term $(k - j)^2$ that amplifies values that are far from the peak in a way that is possible to find the second peak, corresponding to the object. Fig. 6 shows the result of the thresholding process of an image.

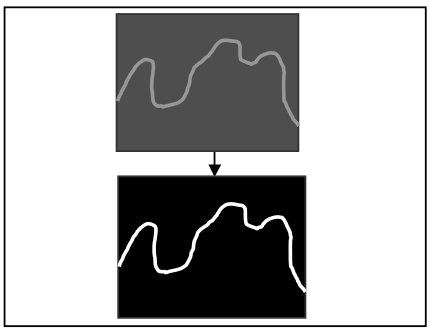

Figure 6. Thresholding of an image

2.1.2. Point sequencing

After the thresholding, the image becomes a matrix of black and white points. These points do not have a sequence. First step is to determine the starting point of the trajectory. According to the user determination, starting point can be set as, for example, the first white point at the left. The algorithm for that is determining the white point of the coordinate with lowest x_{pixel} value. Second step is to find the closest white point to the starting point. The algorithm determines the distance of each white point to the starting point and considers the one with the lowest distance. This step is repeated to the following point and a sequence is established as in Fig. 7.

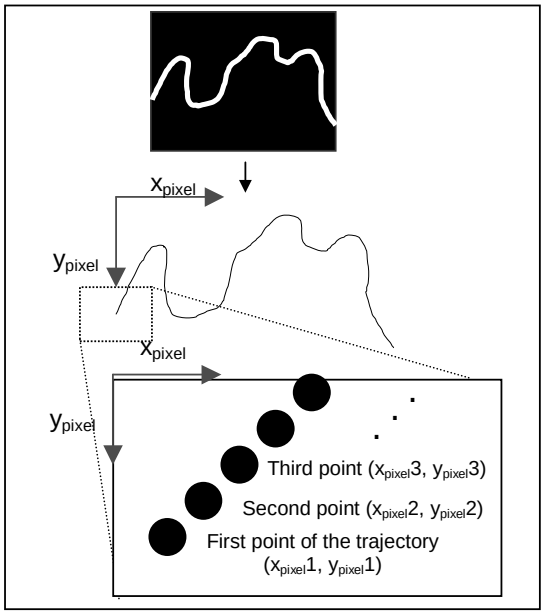

Figure 7. Point sequencing. First point extreme left, following are the points with smallest distance.

2.2. Manipulator Movement

Section 2.1 has converted the trajectory into a sequence of points with x_{pixel},y_{pixel} coordinates. Each point has to be transformed into the manipulator joint space.

The working space of the manipulator can be calculated by the analysis of direct kinematics.

The manipulator used in the implementation was a Robix RCS-6, an educational manipulator with small dimensions (links with 9 and 14 mm), 2 joints, cylindrical configuration as shown in Fig. 1. By using the approach described in Fu(1988), the coordinate system from the end-effector of the manipulator can be translated into the base coordinate system. This is done by homogeneous matrix in equation (4).

$$ {}^{0}A_{4}=\begin{bmatrix} -\mathbf{sen}(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3) & -\cos(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3) & 0 & -\mathbf{a}_4\mathbf{sen}(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3)-\mathbf{a}_2\mathbf{sen}\boldsymbol{\theta}_1 \\ \cos(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3) & -\mathbf{sen}(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3) & 0 & \mathbf{a}_4\cos(\boldsymbol{\theta}_1+\boldsymbol{\theta}_3)+\mathbf{a}_2\cos\boldsymbol{\theta}_1 \\ 0 & 0 & 1 & \mathbf{d}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4) $$

${}^{0}A_4$ - homogeneous matrix to transform end-effector coordinate (x_4,y_4,z_4) into (x_o,y_o,z_o).

θ_1 - angle of joint #1

θ_2 - angle of joint #2
a_2 - link between joint #1 - joint #2 (9mm)
a_4 - link between joint #2 - end-effector (14mm)
d_1 - height of joint #1

A simulation varying angles was done in MatLab and Fig. 8 has the result of the working area.

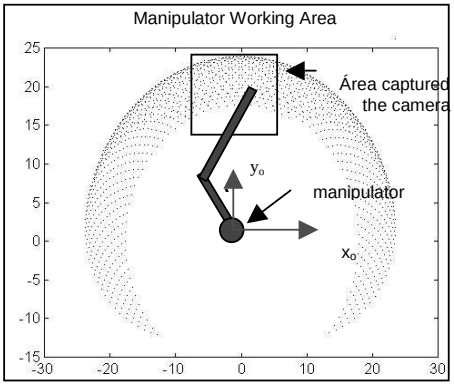

Figure 8. Working area of the manipulator and captured by the camera.

To transform coordinate to joint space, an analysis of "working area - manipulator - camera" is done in Fig. 9.

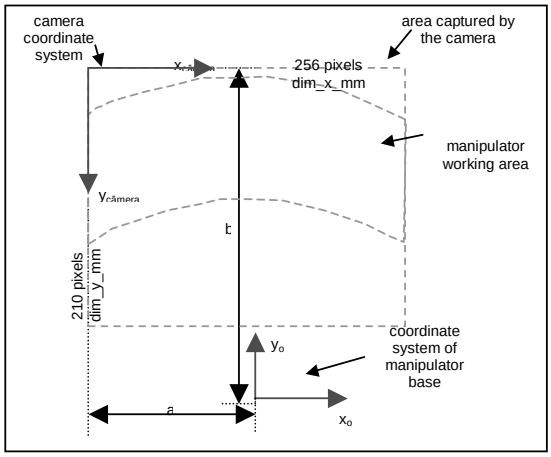

Figure 9. Working space - manipulator – camera analysis

$x_o = x_{pixel} \cdot dim\_x\_mm / 256 - a$
$y_o = - y_{pixel} \cdot dim\_y\_mm / 210 + b$
a= 79 mm
b = 243 mm
dim_x_mm = 146.5 mm
dim_y_mm = 100 mm

$$x_o = x_{pixel} \cdot \frac{146.5}{256} - 73 \qquad (5)$$

$$y_o = -y_{pixel} \cdot \frac{100}{210} + 243 \qquad (6)$$

The analysis of coordinate system and angles of joints yields to the following reverse kinematics:

$$\theta_1 = 2 \cdot \operatorname{arctg}\left(\frac{-r_2 \operatorname{sen}\theta_3 \pm \sqrt{\Delta}}{y_0 + r_1 + r_2 \cos\theta_3}\right) \qquad (7)$$

where $\Delta = r_2^2 \operatorname{sen}^2\theta_3 - y_o^2 + r_1^2 + r_2^2 \cos^2\theta_3 + 2 r_1 r_2 \cos\theta_3$

$$\theta_3 = \arccos\left(\frac{x_o^2 + y_o^2 - r_1^2 - r_2^2}{2\, r_1\, r_2}\right) \qquad (8)$$

Where: r_1 - link between joint 1 and joint 2

r_2 - link between joint 2 and end-effector

As Section 2.1 has the image described as a sequence of (x_{pixel},y_{pixel}), the current section shows the transformation into (x_o,y_o), and then into joint angles (θ_1, θ_3).

The manipulator was designed with an interface that controls the servomotor in each joint.

The manufacturer has defined the joints to be moved according to units of servomotor. These units do not have any physical meaning, they are only a translation of the angle of the servomotor. Empirically the corresponding unit of servomotor and the correlation with angles was determined. This was done with the command "move 1 to 400" (this command moves the servomotor 1 to 400 units of servomotor) and the corresponding angle was measured. The same was repeated to servomotor 2. The result is:

$$\text{pos\_servo } 1 = \theta_1 \cdot 1046 \qquad (9)$$

$$\text{pos\_servo } 2 = \theta_3 \cdot 1060 \qquad (10)$$

In summary, after having a point of the image with coordinate x_{pixel},y_{pixel}, it is possible to calculate x_o,y_o by equations (5) and (6), determine θ_1 and θ_3 , by equations (7) and (8) and finally units of servo1 and servo2 with (9) and (10).

The command for the manipulator will be:

"move 1 to pos_servo1"

"move 2 to pos_servo2"

2.3. Control Program

The manipulator has a library that allows programming in C language. Simple commands like "move 1 to 400" are executed in this language.

Image processing and manipulator movement are all written in C language, therefore, one program can execute the sequences described in sections 2.1 and 2.2.

The algorithm has the following parts:

- manipulator initialization;
- frame grabber and image capture initialization;
- image edge detection (Prewitt);
- threshold calculation and binarization;
- point sequencing;
- reverse kinematics calculation and movement of the manipulator.

2.4 Results

The system implementation described in Fig. 1 was carried out and the Fig. 10 shows the results.

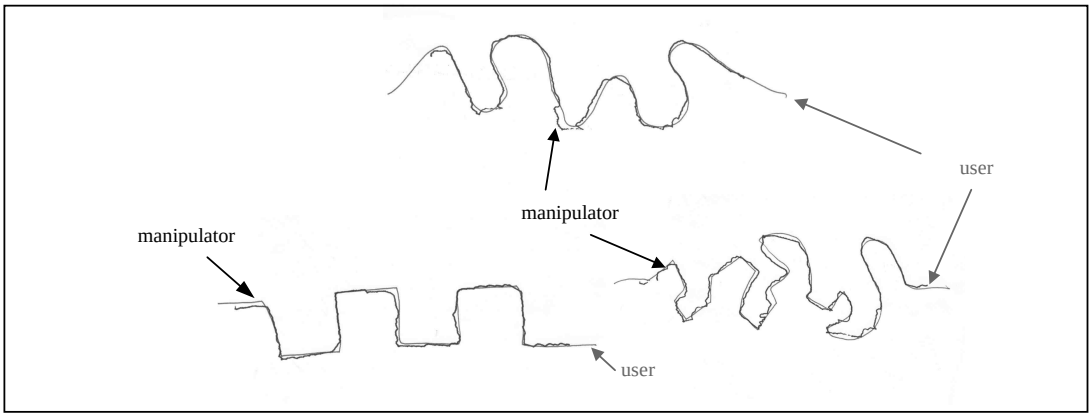

Figure 10. Results of trajectory marked by the manipulator

2.4.1 Accuracy

Analyzing Fig. 10, there are points within 3 mm of error in x direction and 1 mm in y direction. Considering that x has a length of 140 mm for the camera range, and y 100mm, errors are inside 5%.

Regarding to a qualitative analysis, the curve made by the manipulator is close to the one marked by the user.

2.4.2 Repeatability

This test was carried out with 4 identical trajectories and results are in Fig. 11.

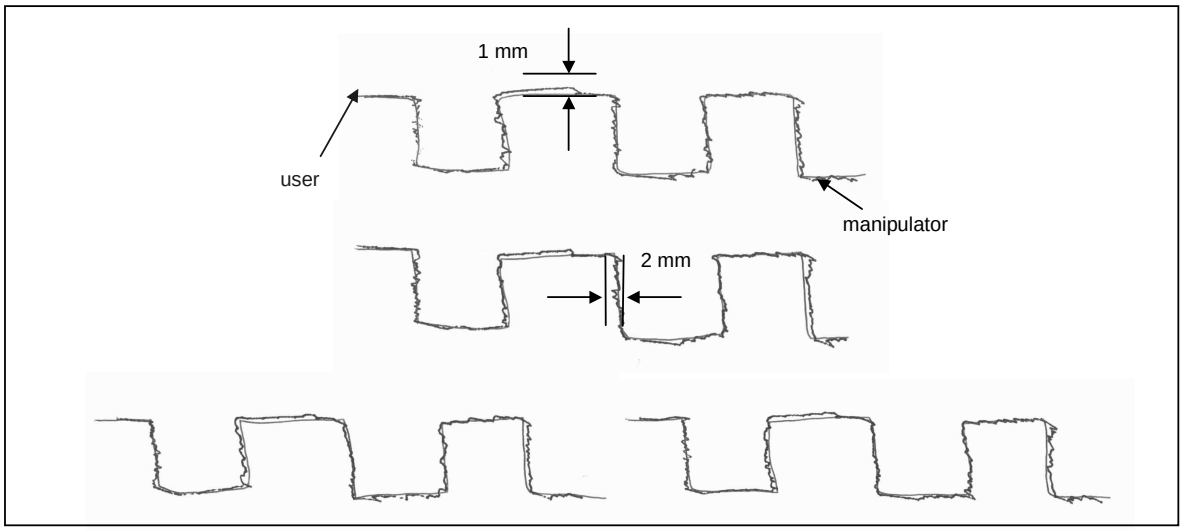

Figure 11. Repeatability assessment with 4 identical trajectories

Fig. 11 shows that similar errors are repeated for all trajectories. In x direction, the error is within 2 mm, which means 2 mm in 146.5mm (<2%). In y direction, repeatability is better, within 1 mm in 100mm (1%).

The difference between x and y can be explained:

- in y direction the angular movement of servomotor 1 is bigger than servomotor 2, i.e., error is influenced by only one servomotor;
- in x direction both servomotors 1 and 2 influence the movement, this effects directly the repeatability.
-

2.4.3. Trajectory maximum variation (maximum proximity of segments)

The smallest distance allowed between segments of the trajectory so that the system still recognizes and follows the trajectory depends on:

- accuracy of image capture system, i.e., 256 x 210 pixels for an area of 146.5 x 100 mm;
- accuracy due to edge detector, because it is necessary information around one pixel to detect the edge as explained in section 2.1.1;
- accuracy of binarization. Thresholding may have faults, mainly when low contrast is present after edge detection;
- point sequencing is calculated based on pixel distances, so if the errors above happen, it might generate improper sequencing;

All those factors are interrelated, therefore a trajectory was marked as Fig. 12 to check the maximum allowed proximity between segments.

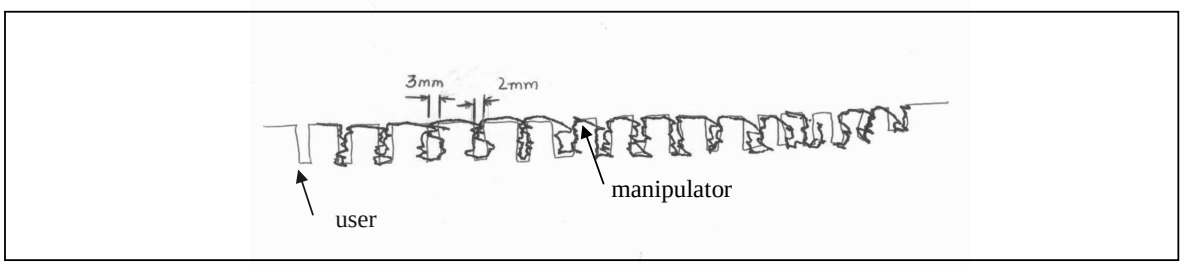

Figure 12. Trajectory with different distances between segments

Segments closer than 3 mm in a range of 146.5mm create a fault in the system and the curve is not properly detected.

2.4.4 Constraints of the trajectory trace

Due to the system configuration, the trajectory must:

- be inside the working are of the manipulator;
- have no crossing points;
- have segments with 3 mm of distance minimum as shown in Fig. 10;
- have interruptions smaller than 5 mm;

2.5. Trajectory optimization

The system camera - microcomputer generates several errors and it may occur:

a) error in the trajectory detection by the camera;
b) edge detection fault;
c) thresholding error;
d) fault in point sequencing of the trajectory;
e) error in calculation of joint angle and units of servomotor;
f) backlash of servomotor gears;
g) manipulator arm may have deformations.

Some of the errors listed may be corrected by applying a compensation.

Optimization is to have an initial trace marked by the manipulator and, by the assessment of the trace, the error with reference to the trajectory will be minimized.

The optimization may be a continuous process, with a closed-loop that corrects the movement during the tracing or with a calibration process, starting from an error and correcting with iterations.

The methods can be:

a) utilize the feedback loop in real time and capture the image of the manipulator while tracing;
b) capture the trajectory image after the manipulator finalizes the complete trace. In this option, the correction will be in the next cycle of trajectory trace.

To implement option a, several problems have to be solved:

- manipulator stays between the image and the camera, so with the current configuration, the image capture is unfeasible;
- for each point the camera has to capture the image and the microcomputer must make a new calculation taking into account the error measured, but the errors like manipulator deformation or gear backlash do not have the same behavior in all directions;
- variation in the trace width creates a quantity of points different from the original trajectory and may influence the error calculation;
- after capturing the trajectory trace, an interpretation must be carried out. The interpretation is done by edge detection, thresholding and point sequencing, i.e., it requires many operations by the computer, which takes time. This has an impact in a real time performance.

Option b for optimization requires an approach to solve the following problems:

- determination of new trajectory marked by the manipulator, subtracting the initial trajectory. Also it is done a edge detection, thresholding and point sequencing;
- after determination of new trajectory it is necessary to calculate the difference between trajectory marked and initial trajectory. The quantity of points determined in the initial trajectory will not be the same as the one by the manipulator because it depends on pen pressure, image contrast, lighting, and threshold value calculated;
- the variables make the determination of a correlation between points of initial trajectory and the one marked is a complex task;
- a proposal could be to use an average trajectory for initial and marked trajectory, but this approach is also complex.

This work proposes an alternative method to option b, to simplify the trajectory optimization:

- capture the image and process it;
- each point to be marked, the manipulator do not mark the complete trajectory, but a small line, and after that the manipulator moves away from the working area;
- the camera captures the image around the point to be marked as a window and subtracts the original image;
- a calculation of the threshold for the region is carried out to determine x_{pixel} and y_{pixel} of the small line;
- next cycle the manipulator will make a complete new trace considering all differences calculated.

Fig. 13 shows the cycle to optimize the trajectory trace.

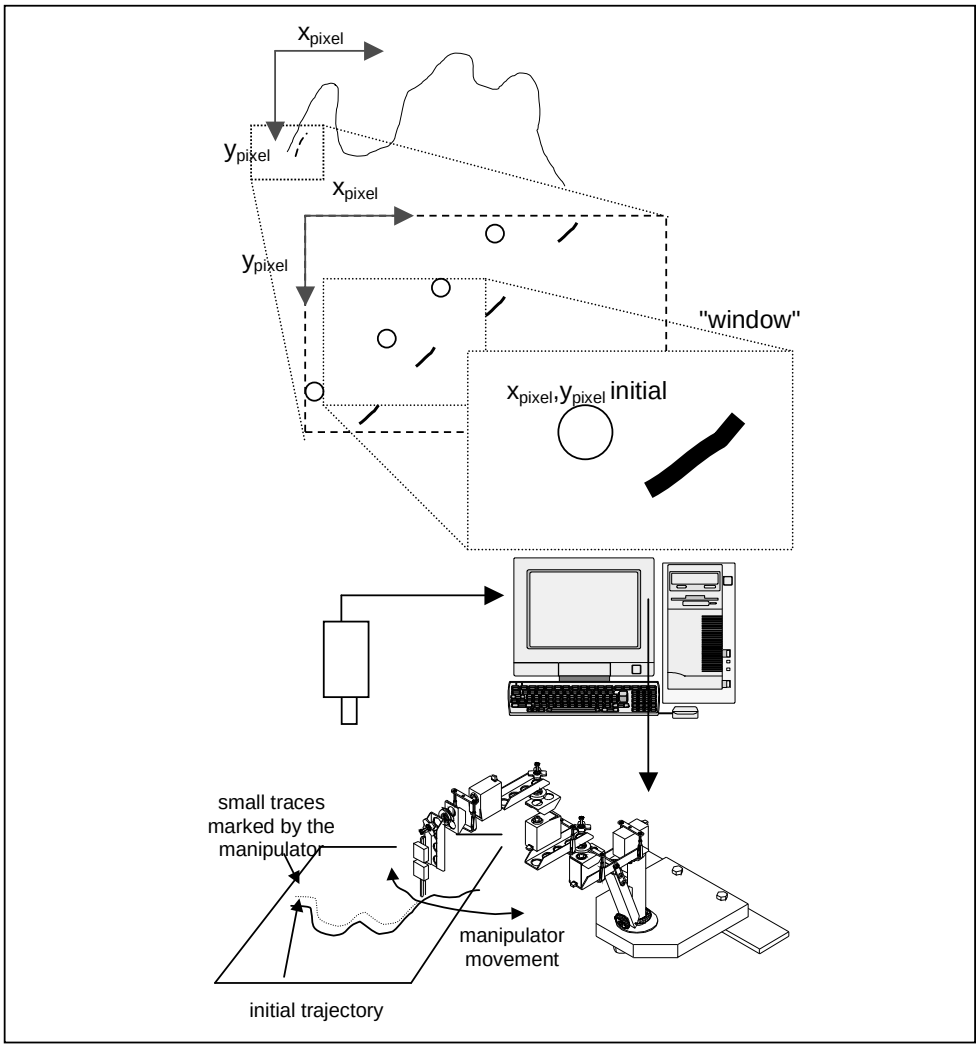

Figure 13. Trajectory optimization

2.6. Conclusion

The proposal of this work was to implement a system with a robotic manipulator that follows a trajectory randomly marked by a user in a sheet of paper.

For that purpose, it was implemented a system shown in Fig. 1, comprising a robotic manipulator, Robix RCS-6, a CCD camera model Sony HAD B&W, a frame grabber FG 201 Humosoft, a microcomputer with C language and Robix RCS-6 library installed.

The camera captures the trajectory, the image is received by the frame grabber and sent to a microcomputer and a program will process it. The trajectory will be transformed into a sequence of points and to the joint space (angles) of the manipulator. Then the program moves the manipulator according to those angles, marking the trajectory.

Trajectory contrast in a sheet of paper is not enough to separate object (trajectory) from the background (sheet of paper). Therefore, image processing techniques are used.

The first technique is to use a edge detection filter, so called Prewitt filter. Following is the calculation of threshold that allows binarizing the picture (separating object, white points, from the background, black points). Finally a point sequencing is attained from a initial point determined by the user.

Fig. 4, 6 and 7 show the sequence of image processing.

The determination of the working area is done by using Denavit-Hartenber homogeneous matrix, correlating coordinates from the sheet of paper to joint space (angles) of the manipulator.

The system working area - manipulator - camera was analyzed and a sequence of coordinate transformation was determined.

Results are presented from the implementation in Fig. 10 showing that the trace has points that coincide with the original trajectory and errors less than 3 mm in a range of 100mm, i.e., less than 5%. Visually, the shape of the trajectory marked is similar to the one marked by the user.

Practical results are in Tab. 1.

Table 1. Practical results from implementation

	x	y
Accuracy	< 5%	< 5%
Repeatability	< 3%	≅ 1%
Trace (distance between trajectory segments)	≅ 3 mm	≅ 3 mm

To achieve a trajectory trace, according to the practical results, the following limitations should be considered:

- trajectory must be inside working area of the manipulator as shown in Fig. 9;
- trajectory must have no crossing point;
- trajectory segments must have a distance of 3 mm minimum;
- interruptions of the trajectory are allowed until 5 mm, but the manipulator will disregard the interruption.

To attain the required accuracy of the trace, it is suggested an optimization process in Fig. 13. The manipulator marks each point with a small line and moves away from the working area so the camera can capture the new image and the system sorts out the line from the original trajectory, calculating the error that will be used in the next cycle of trajectory marking by the manipulator.

Another way to minimize the errors is to improve the manipulator robustness.

C language allowed the development of routines and also an interface to communicate with the manipulator, by utilizing the library supplied by the manufacturer.

Following is a list of future developments to improve the system and integrate it to robotic applications:

- use the same system in a bigger robot;
- use the system conjugated with a welding system;
- develop a two-camera system, allowing three-dimensional movements;
- implementation of the system together with an object detection to allow new trajectory for new configuration of intermediary objects.

3. Acknowledge

To Professors from the Mechanical and Electrical Engineering Departments that helped the development of the system.
To Aurelio for the support during the implementation.
To Professor Grandinetti for releasing the equipment from Robotics Laboratory.
To my friend Carlos and to Mr. Kojima for the incentive, support and motivation.

4. Reference list

CORKE, P.I. (1996) *Visual control of robots*. Research Studies Press Ltd, Somerset, England, 353 p

FG201 High Resolution gray scale Frame Graber. (1992) User's manual. Humusoft, Praga, Tchek Republic, , 16 p.

FU, K. S. et al. (1988). *Robotics: control, sensing, vision, and intelligence*, McGraw-Hill, Singapore, 580 p.

KABAYAMA, A. M. (1999). *Implementação e análise de técnicas de fusão sensorial*, São José dos Campos. Tese Instituto Tecnológico de Aeronáutica. 151 p.

LINDLEY, C.A. (1991) *Practical image processing in C: acquisition, manipulation, storage*. John Wiley & Sons, Inc., Canada. 554p.

OGATA, K. (1996) *Modern Control Engineering*. Prentice-Hall, New Jersey, EUA, 3rd. Ed., 997p.

PARKER, J.R. (1997) *Algorithms for image processing and computer vision*. John Willey & Sons, Inc., New York, USA, 417 p.

QUERIDO, S. C. F. (1999) *Implementação de um sistema de aquisição de imagem*. Taubaté.. Trabalho de graduação interdisciplinar. 67p

RIBEIRO, N.F. & YAMAGUCHI, O. (1999) *Cinemática direta e inversa de manipuladores robóticos com elos rígidos*. Taubaté. 1999. Trabalho de graduação interdisciplinar. Computação Científica da Universidade de Taubaté. 99 p.

Robix RCS-6 Robot Construction Set. (1995) User guide and project book. Advanced Design. EUA.. 100 p.

User guide and project book, Robix RCS-6 robot construction set, manipulator user manual download from www.robix.com, Advanced Design, Inc., 1995

www.robix.com (Robix® site). Advanced Design, Inc, Tucson, Arizona, EUA. Consulted in 20/Jan/2001.