DEVELOPMENT OF A FAST VISUAL SENSOR OF 3D POSITION FOR A MANIPULATOR ROBOT

Marcelo B. Nogueira[1*], Adelardo A. D. Medeiros[1] , Pablo J. Alsina[1]

[1]Universidade Federal do Rio Grande do Norte
Departamento de Engenharia de Computação e Automação
UFRN-CT-DCA – Campus Universitário – 59072-970 Natal-RN Brazil

nogueira,adelardo,pablo@dca.ufrn.br

Abstract. *This paper presents a visual sensor of 3D position for a manipulator robot. The proposed system analyzes digital images to compute the configuration of the robot tool directly in Cartesian space. The mathematical background is revised and we propose two simple and accurate techniques to process the images during the phases of calibrating the camera and detecting the tool. The proposed algorithms were tested and considered robust and fast enough to be used in real time control and with noise images.*

1. Introduction

The conventional control techniques for manipulator robots measure the joint angles and use the inverse kinematic model of the robot to generate references to the joint controllers, by calculating a set of joint angles which attains the given position and orientation. This kind of partially open-loop control is limited by the precision of the inverse kinematic model. To accomplish a really closed-loop control, we must measure the tool configuration directly in Cartesian space.

This paper proposes to obtain this information using a extern camera. We suppose the robot is working in a controlled environment, in such a way we can introduce a distinguishable mark in the end effector of the manipulator and there is no possibility of occlusion and/or confusion with similar patterns.

The methods to solve this problem are well known, provided we can detect some notable points in the image. In real-time applications, most of the usual image processing techniques are too time-consuming to be useful. In this work we resume the existing theory about determing a 3D position from a 2D image and propose a fast image processing algorith to determine the required notable points. The calculation of the 3D position is accomplished in two steps: an off-line calibration, presented in sections 2.1 and 3, and an on-line determination of the tool configuration, detailed in sections 2.2 and 4. Some experimental results and the conclusions are presented in sections 5 and 6, respectively.

*Beneficiary of financial aid by CAPES–Brazil

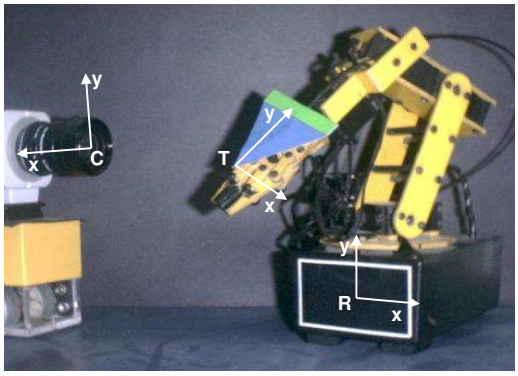

Figure 1: Position of the frames

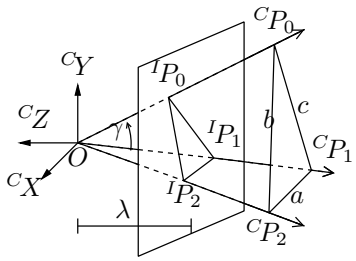

Figure 2: 2D projection problem

2. Visual determination of 3D position

Let $\{R\}$, $\{C\}$ and $\{T\}$ be respectively the frame on the robot basis, at the camera center of perspective and on the tool, as in figure 1. Suppose a point P relative to $\{R\}$, that is $^{R}P = [\ ^{R}x \quad ^{R}y \quad ^{R}z\ ]^{T}$. This same point can be expressed relative to $\{C\}$ as shown in equation 1, where $^{C}P = [\ ^{C}x \quad ^{C}y \quad ^{C}z\ ]^{T}$ is a point relative to the camera frame and $^{C}T_R$ is the transformation matrix between $\{C\}$ and $\{R\}$.

$$\begin{bmatrix} ^{C}P & 1 \end{bmatrix}^{T} = {}^{C}T_R \cdot \begin{bmatrix} ^{R}P & 1 \end{bmatrix}^{T} \tag{1}$$

Every target point has a representation in the image projection plane, as in figure 2. Using the pinhole camera model, the relations between the image coordinates $^{I}P = [\ ^{I}x \quad ^{I}y\ ]^{T}$ and the camera coordinates ^{C}P are given in equation 2, where λ is the camera focal distance.

$$^{I}x = \left(\lambda / -\,^{C}z\right) {}^{C}x \qquad\qquad ^{I}y = \left(\lambda / -\,^{C}z\right) {}^{C}y \tag{2}$$

2.1. Camera Calibration

The problem of camera calibration is to compute the camera intrinsic and extrinsic parameters. The intrinsic parameters characterize the inherent properties of the camera optics, including the focal lenght, the image center, the image scaling factor and the lens distortion coefficients. The extrinsic parameters indicate the position and orientation of the camera with respect to a world coordinate system. Several techniques have already proposed to determine the intrinsic parameters [Nomura et al., 1992, Gurdjos and Sturrn, 2003] and will not be revised in this article: we suppose known the focal length and the conversion from pixels to centimeters.

The adopted calibration technique to determine the extrinsic parameters is based on the P4P (the *Perspective Four-Points* problem) formulation proposed by Kamata *et al* [Kamata et al., 1992], because of the uniqueness of the solution. We suppose we know the position of four coplanar points relative to the robot, $^{R}P_0$ to $^{R}P_3$, and the corresponding 2D points in the image, $^{I}P_0$ to $^{I}P_3$.

To calculate $^{C}T_R$, we introduce two intermediate coordinate systems, the $\{A\}$ system and the $\{B\}$ system, and write $^{C}T_R$ as in equation 3:

$$^{C}T_R = {}^{C}T_B\, {}^{B}T_A\, {}^{A}T_R \tag{3}$$

2.1.1. Calculation of the matrix ${}^{A}T_{R}$

We choose the A-system so that the target points in this system are in what we call a *standard position*: ${}^{R}P_0$ is at the origin, ${}^{R}P_1$ is on the positive x-axis and ${}^{R}P_2$ is in the first or second quadrant of $x-y$ plane. The a_{ij} elements of the ${}^{A}T_{R}$ matrix are determined by the following equations, where for short $s\odot$, $c\odot$ and $\Delta\ominus_{ij}$, $\odot \in (\alpha, \beta, \theta)$, $\ominus \in (x, y, z)$, stand for $\sin(\odot)$, $\cos(\odot)$ and ${}^{R}\ominus_i - {}^{R}\ominus_j$, respectively:

$$\theta = \begin{cases} 0 & \text{if } \Delta y_{10} = \Delta x_{10} = 0 \\ \tan^{-1}\left(\dfrac{\Delta y_{10}}{\Delta x_{10}}\right) & \text{otherwise} \end{cases} \qquad \beta = \tan^{-1}\left(\frac{-\Delta z_{10}}{\Delta x_{10} c\theta + \Delta y_{10} s\theta}\right)$$

$$\alpha = \tan^{-1}\left[\frac{(\Delta x_{20} c\theta + \Delta y_{20} s\theta)s\beta + \Delta z_{20} c\beta}{-\Delta x_{20} s\theta + \Delta y_{20} c\theta}\right]$$

$$a_{11} = c\beta c\theta \qquad a_{12} = c\beta s\theta \qquad a_{13} = -s\beta \qquad a_{14} = -{}^{R}x_0 c\beta c\theta - {}^{R}y_0 c\beta s\theta + {}^{R}z_0 s\beta$$

$$a_{21} = -c\alpha s\theta + s\alpha s\beta c\theta \qquad a_{22} = c\alpha c\theta + s\alpha s\beta s\theta$$

$$a_{23} = s\alpha c\beta \qquad a_{24} = -{}^{R}x_0(-c\alpha s\theta + s\alpha s\beta c\theta) - {}^{R}y_0(c\alpha c\theta + s\alpha s\beta s\theta) - {}^{R}z_0(s\alpha c\beta)$$

$$a_{31} = s\alpha s\theta + c\alpha s\beta c\theta \qquad a_{32} = -s\alpha c\theta + c\alpha s\beta s\theta$$

$$a_{33} = c\alpha c\beta \qquad a_{34} = -{}^{R}x_0(s\alpha s\theta + c\alpha s\beta c\theta) - {}^{R}y_0(-s\alpha c\theta + c\alpha s\beta s\theta) - {}^{R}z_0(c\alpha c\beta)$$

$$a_{41} = a_{42} = a_{43} = 0 \qquad a_{44} = 1$$

2.1.2. Calculation of the matrix ${}^{C}T_{B}$

The B-system is chosen to be aligned with a fictitious camera which has its lens center at the same location as the real camera, but is oriented such that ${}^{J}P_0$, the new image of ${}^{R}P_0$, is at the image origin and ${}^{J}P_1$ is on the positive x-axis. We say this fictitious camera is in *ideal position*. Using the notation $s\odot = \sin(\odot)$ and $c\odot = \cos(\odot)$, $\odot \in (\phi, \omega, \rho)$, the ${}^{C}T_{B}$ matrix is given by:

$$\phi = \tan^{-1}\left(\frac{-{}^{I}x_0}{\lambda}\right) \qquad \omega = \tan^{-1}\left(\frac{{}^{I}y_0 c\phi}{\lambda}\right) \qquad \rho = \tan^{-1}\left(\frac{{}^{I}y_1 c\omega + {}^{I}x_1 s\phi s\omega - \lambda c\phi s\omega}{{}^{I}x_1 c\phi + \lambda s\phi}\right)$$

$${}^{C}T_{B} = \begin{bmatrix} c\phi c\rho + s\phi s\omega s\rho & -c\phi s\rho + s\phi s\omega c\rho & s\phi c\omega & 0 \\ c\omega s\rho & c\omega c\rho & -s\omega & 0 \\ -s\phi c\rho + c\phi s\omega s\rho & s\phi s\rho + c\phi s\omega c\rho & c\phi c\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.1.3. Calculation of the matrix ${}^{B}T_{A}$

The matrix ${}^{B}T_{A}$ represents the solution to a problem in which target points in standard position are viewed by a camera in ideal position. The coordinates of the target point in standard position, ${}^{A}P_i$, are found by applying the transform ${}^{A}T_{R}$ to the original target points, ${}^{R}P_i$. The image coordinates of these points as seen by a camera in ideal position, ${}^{J}P_i$, are found by applying the transform $({}^{C}T_{B})^{-1}$ $[= ({}^{C}T_{B})^{T}]$ to the 3D points $({}^{I}x_i, {}^{I}y_i, \lambda)$, producing the points $({}^{B}x_i, {}^{B}y_i, {}^{B}z_i)$, and then setting ${}^{J}x_i = -\lambda {}^{B}x_i / {}^{B}z_i$ and ${}^{J}y_i = -\lambda {}^{B}y_i / {}^{B}z_i$.

Because of the way in which we defined the $\{A\}$ and $\{B\}$ systems, the c_{ij} parameters of the ${}^{B}T_{A}$ can be unequivocally determined. First we calculate some intermediate values:

$$b_1 = \lambda({}^{A}x_2\,{}^{A}y_3 - {}^{A}x_3\,{}^{A}y_2) \qquad b_2 = {}^{B}x_2\,{}^{A}x_2\,{}^{A}y_3 - {}^{B}x_3\,{}^{A}x_3\,{}^{A}y_2 \qquad b_3 = ({}^{B}x_2 - {}^{B}x_3)\,{}^{A}y_2\,{}^{A}y_3$$
$$b_4 = {}^{B}x_2\,{}^{A}y_3 - {}^{B}x_3\,{}^{A}y_2 \qquad b_5 = {}^{B}y_2\,{}^{A}x_2\,{}^{A}y_3 - {}^{B}y_3\,{}^{A}x_3\,{}^{A}y_2 \qquad b_6 = ({}^{B}y_2 - {}^{B}y_3)\,{}^{A}y_2\,{}^{A}y_3$$
$$b_7 = {}^{B}y_2\,{}^{A}y_3 - {}^{B}y_3\,{}^{A}y_2 \qquad b_8 = \lambda\,{}^{A}x_1(b_4b_6 - b_7b_3) - {}^{B}x_1b_1b_6$$
$$b_9 = {}^{B}x_1[{}^{A}x_1(b_4b_6 - b_7b_3) - (b_2b_6 - b_5b_3)]$$

Then, the coefficients are given by:

$$c_{11} = \begin{cases} +\sqrt{1/[1 + (b_8/b_9)^2]} & \text{if } \lambda/{}^{B}x_1 \leq b_8/b_9 \\ -\sqrt{1/[1 + (b_8/b_9)^2]} & \text{otherwise} \end{cases}$$
$$c_{31} = -b_8c_{11}/b_9 \qquad c_{34} = -{}^{A}x_1(\lambda c_{11}/{}^{B}x_1 + c_{31})$$
$$c_{32} = \begin{cases} -(b_5c_{31} + b_7c_{34})/b_6 & \text{if } b_6 \neq 0 \\ -(b_1c_{11} + b_2c_{31} + b_4c_{34})/b_3 & \text{otherwise} \end{cases}$$
$$c_{12} = \frac{\lambda\,{}^{A}x_2c_{11} + {}^{B}x_2\,{}^{A}x_2c_{31} + {}^{B}x_2\,{}^{A}y_2c_{32} + {}^{B}x_2c_{34}}{-\lambda\,{}^{A}y_2}$$
$$c_{22} = -{}^{B}y_2({}^{A}x_2c_{31} + {}^{A}y_2c_{32} + c_{34})/(\lambda\,{}^{A}y_2) \qquad c_{13} = -c_{22}c_{31} \qquad c_{23} = c_{12}c_{31} - c_{11}c_{32}$$
$$c_{14} = c_{21} = c_{24} = c_{41} = c_{42} = c_{43} = 0 \qquad c_{33} = c_{11}c_{22} \qquad c_{44} = 1$$

2.2. Tool Configuration Determination

In this phase, our objective is to calculate the position and orientation of the tool frame relative to the robot frame, by determining the matrix ${}^{R}T_{T}$. This matrix can be derived from the matrices ${}^{C}T_{R}$ and ${}^{C}T_{T}$, as shown in equation 4. The matrix ${}^{C}T_{R}$ was calculated in the camera calibration phase. Then, this step is actually about calculating the transformation matrix ${}^{C}T_{T}$.

$${}^{R}T_{T} = {}^{R}T_{C} \cdot {}^{C}T_{T} = ({}^{C}T_{R})^{-1} \cdot {}^{C}T_{T} \tag{4}$$

Given the perspective projection of three known points in 3D space (the *P3P – Perspective Three-Point –* problem), it is possible to determine as many as four possible sets of positions of the points*. The three observed points form a triangle, illustrated in figure 2. Let the known side lenghts of the triangle be:

$$a = \left\|{}^{R}P_1 - {}^{R}P_2\right\| = \left\|{}^{C}P_1 - {}^{C}P_2\right\| \qquad b = \left\|{}^{R}P_2 - {}^{R}P_0\right\| \qquad c = \left\|{}^{R}P_0 - {}^{R}P_1\right\|$$

The unit vectors e_i from the center of perspectivity to the points ${}^{C}P_i$ are given by:

$$e_i = \frac{1}{\sqrt{{}^{I}x_i^2 + {}^{I}y_i^2 + \lambda^2}} \begin{bmatrix} {}^{I}x_i \\ {}^{I}y_i \\ -\lambda \end{bmatrix}$$

*The calcul of ${}^{C}T_{T}$ is similar to calculating ${}^{C}T_{R}$ and could be done using the same technique, the P4P approach. However, the determination of the tool configuration must be done in real-time, recommending the adoption of the faster P3P formulation.

Let the angles at the center of perspectivity opposing the sides a, b and c be α, β and γ, respectively (the γ angle is represented in figure 2). These angles are given by $\cos\alpha = c\alpha = e_1 \cdot e_2$, $\cos\beta = c\beta = e_2 \cdot e_0$ and $\cos\gamma = c\gamma = e_0 \cdot e_1$. Let the unknown distances of the points ${}^C P_0$, ${}^C P_1$ and ${}^C P_2$ from the center of perspectivity be d_0, d_1 and d_2, respectively. To determine the position of the points with respect to the camera reference frame, it is sufficient to determine the d_i's, since ${}^C P_i = d_i e_i, i = 1, 2, 3$.

By the law of cosines:

$$\begin{cases} d_1^2 + d_2^2 - 2d_1 d_2 c\alpha = A & (A = a^2) \\ d_0^2 + d_2^2 - 2d_0 d_2 c\beta = B & (B = b^2) \\ d_0^2 + d_1^2 - 2d_0 d_1 c\gamma = C & (C = c^2) \end{cases}$$

Doing $d_1 = ud_0$ and $d_2 = vd_0$, these equations can be manipulated in different ways to obtain a solution. Comparisons of the numerical properties of different resolution methods [Haralick et al., 1991, Juang, 1997] indicate the Fischler and Bolles' solution [Fischler and Bolles, 1981] as the best one between those without singularities and with a direct solution. The final solution can be obtained by solving the the fourth order polynomial equation:

$$D_4 u^4 + D_3 u^3 + D_2 u^2 + D_1 u + D_0 = 0 \tag{5}$$

where

$$\begin{aligned} D_4 &= 4BCc^2\alpha - (A-B-C)^2 \\ D_3 &= -4C(A+B-C)c\alpha c\beta - 8BCc^2\alpha c\gamma + 4(A-B-C)(A-B)c\gamma \\ D_2 &= 4C(A-C)c^2\beta + 8C(A+B)c\alpha c\beta c\gamma + 4C(B-C)c^2\alpha - 2(A-B-C)(A-B+C) \\ &\quad - 4(A-B)^2c^2\gamma \\ D_1 &= -8ACc^2\beta c\gamma - 4C(B-C)c\alpha c\beta - 4ACc\alpha c\beta + 4(A-B)(A-B+C)c\gamma \\ D_0 &= 4ACc^2\beta - (A-B+C)^2 \end{aligned}$$

Corresponding to each of the four roots of equation 5 for u there is an associated value for v:

$$v = \frac{(A-B+C) - 2(A-B)c\gamma u + (A-B-C)u^2}{2C(c\beta - c\alpha u)}$$

The distances d_0, d_1 and d_2 ($d_i > 0$) are:

$$d_0^2 = \frac{C}{1 + u^2 - 2uc\gamma} \qquad d_1 = ud_0 \qquad d_2 = vd_0$$

Finally, the coordinates of the observed point in the camera frame are:

$${}^C P_0 = d_0 e_0 \qquad {}^C P_1 = d_1 e_1 \qquad {}^C P_2 = d_2 e_2$$

We designed the triangle mark and placed the $\{T\}$ frame in such a way that ${}^T P_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, ${}^T P_1 = \begin{bmatrix} X & Y & 0 \end{bmatrix}^T$ and ${}^T P_2 = \begin{bmatrix} -X & Y & 0 \end{bmatrix}$. The transformation matrix ${}^C T_T$ is:

$${}^C T_T = \left[\begin{array}{ccc:c} & \Omega & & D \\ \hdashline & & & \\ 0 & 0 & 0 & 1 \end{array}\right]$$

Under these assumptions, the D vector can be calculated by:

$$^{C}P_0 = {}^{C}T_T \cdot {}^{T}P_0 \quad \Rightarrow \quad D = {}^{C}P_0$$

Solving the equations $^{C}P_1 = {}^{C}T_T \cdot {}^{T}P_1$ and $^{C}P_2 = {}^{C}T_T \cdot {}^{T}P_2$, and adopting the roll (ϕ), pitch (θ) and yaw (ψ) representation, the Ω matrix is given by:

$$\Omega = \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & c\phi s\theta c\psi + s\phi s\psi \\ s\phi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix}$$

$$c^2\theta = \frac{(^{C}z_2^2 - {}^{C}z_1^2)\left[(^{C}x_1 - {}^{C}x_2)^2 - (^{C}y_1 - {}^{C}y_2)^2\right]}{4X^2\left[(^{C}x_1^2 - {}^{C}x_2^2) + (^{C}y_1^2 - {}^{C}y_2^2))\right]}$$

$$c\psi = \frac{4X^2c^2\theta(^{C}y_1 + {}^{C}y_2) + (^{C}y_1 - {}^{C}y_2)(^{C}z_1^2 - {}^{C}z_2^2)}{4XYc\theta(^{C}x_1 - {}^{C}x_2)Y}$$

$$s\theta = \frac{-^{C}z_1 + {}^{C}z_2}{2X} \qquad s\psi = \frac{^{C}z_1 - {}^{C}z_2}{2Yc\theta} \qquad s\phi = \frac{^{C}y_1 - {}^{C}y_2}{2Xc\theta} \qquad c\phi = \frac{^{C}x_1 - {}^{C}x_2}{2Xc\theta}$$

3. Determining Rectangle Vertices

To calibrate the camera, we need to find four known points in an image. We placed a rectangular mark on the robot basis, as shown in figure 1. Instead of directly detecting the rectangle vertices, we will look for the straight lines of its edges: the intersection of the edges would give the four points, as shown in figure 3.

We also need to label each point, so we can make the relationship between the points in the image and the coordinates in the robot frame. In this case, we suppose the camera is placed in such a way that the rectangle can always be entirely seen and the rotations are moderate, in the sense that the upper point in the image corresponds to P_2 or P_3, the rightest, to P_1 or P_3 and so on.

To calculate the edges of the rectangle we used the classical Hough algorithm [Hough, 1959]. This algorithm is a method to detect, in an image, a class of geometric forms as straight lines, circles and ellipses. It has a high computational cost, but this step is done only once and before the real-time control process starts.

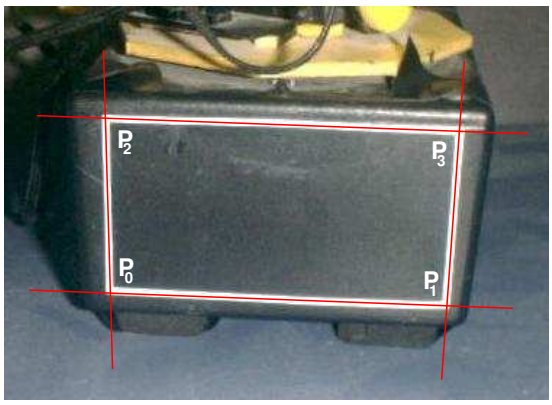

Figure 3: Vertices of the rectangle

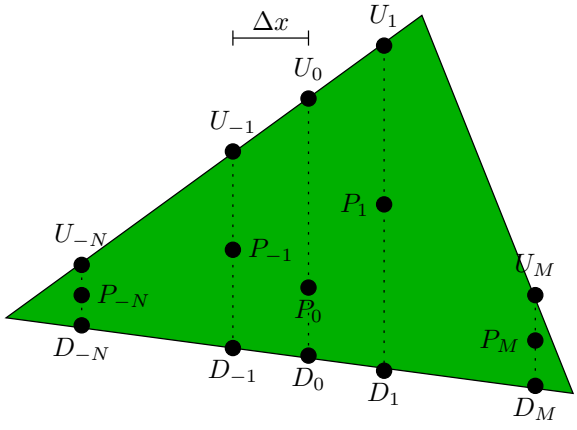

Figure 4: Points of the triangle edges

Table 1: Detecting points of triangle edges

1. Search for a initial pixel P_0 in the image that belongs to the triangle

// Right search

2. $k \leftarrow 0$; $M \leftarrow -1$
3. While the pixel P_k belongs to the triangle:
   (a) From P_k, search upward and downward for the last pixels belonging to the triangle (points U_k and D_k).
   (b) If $\|U_k - D_k\| <$ `MIN_LEN` break While
   (c) Calculate $P_{k+1} \leftarrow \frac{U_k - D_k}{2} + \begin{bmatrix}\Delta x & 0\end{bmatrix}^T$
   (d) $M \leftarrow k$; $k \leftarrow k + 1$

// Left search

4. $k \leftarrow 0$; $N \leftarrow +1$
5. While the pixel P_k belongs to the triangle:
   (a) From P_k, search for U_k and D_k.
   (b) If $\|U_k - D_k\| <$ `MIN_LEN` break While
   (c) Calculate $P_{k-1} \leftarrow \frac{U_k - D_k}{2} - \begin{bmatrix}\Delta x & 0\end{bmatrix}^T$
   (d) $N \leftarrow k$; $k \leftarrow k - 1$

4. Determining Triangle Vertices

To locate the three points in the image we used the same strategy used in the previous step: the intersection of straight lines would give the needed points. Since we need three points, instead of a rectangle we used a triangle (figure 1). We could have used the Hough algorithm once again to calculate the triangle edges, but this phase is executed at each sampling time and requires a faster response. Therefore, we developed a method to extract the edges of a triangle.

4.1. Triangle edges detection

The triangle mark has a very distinguishable color when compared to the other elements in the image (see figure 1). This characteristic makes possible to quickly distinguish a pixel belonging to the triangle. The procedure to detect the edges points is summarized in table 1 and represented in figure 4. At the end of this algorithm we will have determined $M + N + 1$ points that belong to the triangle edges, except for the approximations introduced by noise and quantization. One special case occurs when the triangle has a vertical edge. But this case can be easily detected by observing that the distance $U_k - D_k$ does not become less than `MIN_LEN` before P_k goes out of the triangle.

To reduce the influence of errors, we describe the triangle edges by the best straight lines (in the least squares sense) that fit the sets of points. The best coefficients, given n points (x, y), are:

$$y = ax + b \Rightarrow \begin{cases} a = \dfrac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2} \\ b = \dfrac{\sum(x^2)\sum y - \sum x \sum(xy)}{n\sum(x^2) - (\sum x)^2} \end{cases}$$

Table 2: Detecting lines of triangle edges

// Up

1. Initialize L_L^U with the coefficientes determined by the points U_{-N} and $U_{-(N-1)}$ and L_R^U, with U_M and U_{M-1}.
2. $i \leftarrow -(N-2)$
3. While $i \leq M-2$:
   (a) Calculate the distances e_L^U and e_R^U, the distances between the point U_i and L_L^U and L_R^U, respectively.
   (b) If $e_L^U < e_R^U$, recalculate the coefficients of L_L^U, considering U_i belongs to it; else, recalculate the coefficients of L_R^U.
   (c) $i \leftarrow i+1$

// Down

4. Do a similar procedure (steps 1 to 3) to determine L_L^D and L_R^D from the D_i points.

// Fusion

5. Calculate the difference of inclinations $e^U = \left\|a_L^U - a_R^U\right\|$ and $e^D = \left\|a_L^D - a_R^D\right\|$, where a_L^U is the a coefficient of the L_L^U straight line and so on.
6. If $e^U < e^D$, fuse L_L^U and L_R^U, by calculating the L^U straight line from all the U_i points; else, fuse L_L^D and L_R^D and calculate the L^D straight line.

We determine two straight lines corresponding to the U_i points (one from the left to the center, L_L^U, and the other from the right to the center, L_R^U) and another pair (L_L^D, L_R^D) for the D_i points. Finally, we fuse the straight lines of the pair with the more similar ones. That will give us three lines, whose intersections will determine the vertices. To label each vertices we used the method proposed by Soares [Soares et al., 2001]. The algorithm to calculate and fuse the straight lines is presented in table 2.

5. Results

We will present some results applying the proposed method to the environment shown in figure 1. We used a rectangular marking with $4.3cm$ of height and $7.8cm$ of width, and fixed the robot Frame R at center of the rectangle, so the points in the robot frame will be:

$$^R P_0 = \begin{bmatrix} -2.15 \\ -3.9 \\ 0 \end{bmatrix} \quad ^R P_1 = \begin{bmatrix} 2.15 \\ -3.9 \\ 0 \end{bmatrix} \quad ^R P_2 = \begin{bmatrix} -2.15 \\ 3.9 \\ 0 \end{bmatrix} \quad ^R P_3 = \begin{bmatrix} 2.15 \\ 3.9 \\ 0 \end{bmatrix}$$

The result of the rectangle vertices detection with Hough algorithm is shown in figure 5. The image points found were

$$^I P_0 = \begin{bmatrix} -16 \\ -146 \end{bmatrix} \quad ^I P_2 = \begin{bmatrix} 102 \\ -161 \end{bmatrix} \quad ^I P_3 = \begin{bmatrix} -16 \\ -75 \end{bmatrix} \quad ^I P_4 = \begin{bmatrix} 102 \\ -83 \end{bmatrix}$$

The Triangle marking was $5.1cm$ high and $6cm$ width. We fixed the tool

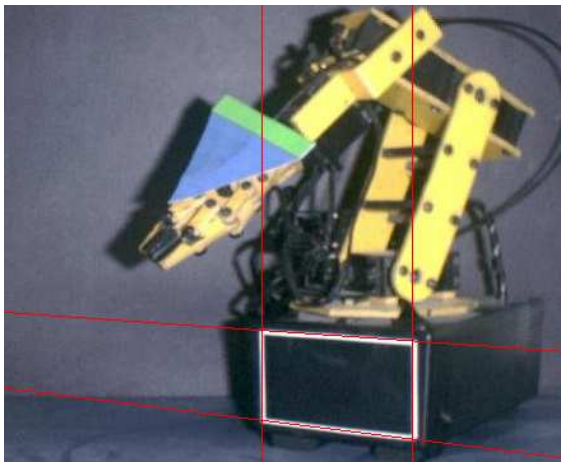

Figure 5: Rectangle vertices

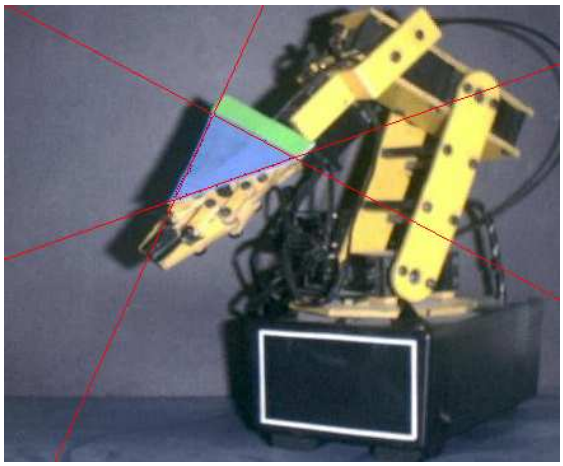

Figure 6: Triangle vertices

frame $\{T\}$ in such a way that:

$$ {}^{T}P_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad {}^{T}P_1 = \begin{bmatrix} -2.5 \\ 4 \\ 0 \end{bmatrix} \qquad {}^{T}P_2 = \begin{bmatrix} 2.5 \\ 4 \\ 0 \end{bmatrix} $$

The triangle vertices detection algorithm result can be seen in figure 6. The image points found were

$$ {}^{I}P_0 = \begin{bmatrix} -84 \\ 28 \end{bmatrix} \qquad {}^{I}P_1 = \begin{bmatrix} -54 \\ 94 \end{bmatrix} \qquad {}^{I}P_2 = \begin{bmatrix} 10 \\ 60 \end{bmatrix} $$

The experiments were reproduced under different conditions (natural and artificial ilumination with incandescent and/or fluorescent lamps, dark and light environments, etc.) and changing the camera position (but always guaranteeing the marks were visible). The results were satisfactory: in a series of 30 different images, the cartesian position of the robot end-effector was always calculated with an error less than 1.0cm.

6. Conclusions

Calibrating a camera using points with known positions detected in an image is a classic problem solved by different methods [Altug et al., 2003, Ansar et al., 2001]. This article presents the equations of the approaches we considere the more appropriated to be used in a real-time context, reduced to their simplest formulation.

For the calibration of the extrinsic parameters of the camera, we propose the adoption of a rectangular mark and the detection of its edges using the Hough algorithm, instead of the direct detection of its vertices, to increase the noise rejection. The method showed to be very satisfactory and robust to noise and illumination changes. The main drawback is the considerable computational cost, but the execution time (about 1 second in an Pentium III, 800 MHz PC computer running Linux) is perfectly acceptable to a procedure being executed only once and off-line.

The algorithm to find the triangle edges is the main contribution of this work. It demonstrated to be very fast (around 1ms to process an image, using the previously described computer) and robust to illumination changes, being appropriate for real-time applications. It works in many different robot configurations, provided the triangle is still well visible.

In next works, the proposed feedback system will be used to actually control the manipulator robot and the results will be compared to conventional control techniques based on measuring the joint angles. Finally, we can investigate the fusion on the measurements of two cameras, to avoid the singularity when the triangle is not well visible by a single camera.

References

Altug, E., Ostrowisk, J. P., and Tayllor, C. J. (2003). Quadrator control using dual camera visual feedback. In *ICRA - IEEE International Conference on Robotics and Automation*, Taipei, Taiwan.

Ansar, A., Rodrigues, D., Desai, J., Daniilidis, K., Kumar, V., and Campos, M. F. (2001). Visual and haptic collaborative tele-presence. *Computer and Graphics*, 25(5).

Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6).

Gurdjos, P. and Sturrn, P. (2003). Methods and geometry for plane-based self-calibration. In *CVPR - IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 491–496, Madison, WI, USA.

Haralick, R. M., Lee, C.-N., Ottenberg, K., and Nölle, M. (1991). Analysis and solutions of the three point perspective pose estimation problem. In *CVPR - IEEE Conference on Computer Vision and Pattern Recognition*, Maui, HA, USA.

Hough, P. V. C. (1959). Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, CERN, Geneva, Switzerland.

Juang, J.-G. (1997). Parameter estimation in the three point perspective projection problem in computer vision. In *ISIE - IEEE International Symposium on Industrial Electronics*, volume 3, Guimarães, Portugal.

Kamata, S.-i., Eason, R. O., Tsuji, M., and Kawaguchi, E. (1992). A camera calibration using 4 point-targets. In *11th IAPR - International Conference on Pattern Recognition*, Hague, Netherlands.

Nomura, Y., Sagara, M., Naruse, H., and Ide, A. (1992). Simple calibration algorithm for high-distortion-lens camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1095–1099.

Soares, A. A. A. F., Alsina, P. J., and Medeiros, A. A. (2001). Desenvolvimento de esquema de controle com realimentação visual para um robô manipulador. In *SBAI - Simpósio Brasileiro de Automação Inteligente*, Canela, RS, Brasil.