International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:10, 2009

Robot Vision Application based on Complex 3D Pose Computation

F. Rotaru, S. Bejinariu, C. D. Niță, R. Luca, I. Păvăloi and C. Lazăr

Abstract**—**The paper presents a technique suitable in robot vision applications where it is not possible to establish the object position from one view. Usually, one view pose calculation methods are based on the correspondence of image features established at a training step and exactly the same image features extracted at the execution step, for a different object pose. When such a correspondence is not feasible because of the lack of specific features a new method is proposed. In the first step the method computes from two views the 3D pose of feature points. Subsequently, using a registration algorithm, the set of 3D feature points extracted at the execution phase is aligned with the set of 3D feature points extracted at the training phase. The result is a Euclidean transform which have to be used by robot head for reorientation at execution step.

Keywords**—**features correspondence, registration algorithm, robot vision, triangulation method.

I. Introduction

A lot of robot vision applications are based on 3D pose calculation using a single view [1, 3, 4, 5].

In a preliminary phase the robot is trained to do the required operation on the specific object, located on an ideal position. Also a set of image features of the object are selected. For each chosen feature is allocated a pattern matching classifier providing feature centroid. The object coordinates of the chosen point features in a Euclidean system attached to the object are known from CAD object model or by direct measurements. These coordinates are transferred to the camera Euclidean system, via extrinsic parameters of the camera calibration.

The features chosen in the preliminary step are recognized during execution phase of the process. A Euclidean transformation (rotations and translations) of the transferred points is then computed. This transformation hypothetically overlaps in the image the preliminary features on the pertaining features recognized in the execution phase. In order to execute the learned robot operation, this Euclidean transform, correlated with the eye-hand transformation, will provide all the information to properly move the robot head in the right position.

The described technique implies an easy to perform exact correspondence of the features extracted during execution phase with the features established in the preliminary phase. Usually this is done allocating a classifier for each chosen feature. Obviously, the features points established in the first step not recognized during execution phase are ignored. To compute the object 3D pose at least three of the chosen features have to be recognized at the execution step.

For some cases, as the one depicted in Fig. 1, the exact point correspondence is not possible. The proposed solution is to compute in a preliminary step the 3D pose of the point features from two views. At the execution phase the 3D pose of the recognized features is again computed. A Euclidean transform is computed using a registration algorithm. This time the transform registers the 3D points computed in the preliminary phase to 3D points extracted at the execution phase.

II. 3D Reconstruction Algorithm

It is obvious that the image object area which provides the chosen feature may differ from the robot action area. In all the figures are depicted only the object part which provide image features. The key is to choose those features that lead to the right Euclidean transform of the registration algorithm.

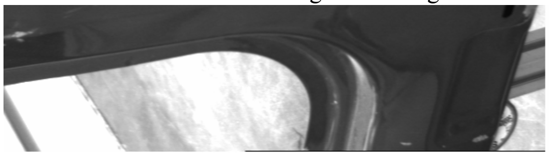
Fig. 1 An object the robot has to work on. There are no point features (corners, line intersections etc.) easily to recognize in various images of the object. The curved edge is chosen as feature. From two images of the object, by stereo techniques, the 3D coordinates of the edge points are computed

The 3D coordinates of the edge points, chosen as features, are computed using a stereo system with cameras $\mathbf{P} = \mathbf{K}[\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = \mathbf{K}'[\mathbf{R}|\mathbf{t}]$. The $\mathbf{K}, \mathbf{K}', \mathbf{R}$ matrices and translation vector **t** are known as result of the calibration process. The 3D edge points may be computed also from the two positions of the single robot camera but this approach is less accurate than the classic stereo one.

The fundamental matrix **F** of the stereo system is: $\mathbf{F} = \mathbf{K}'^{-t}[\mathbf{t}]_{\mathbf{x}}\mathbf{R}\mathbf{K}^{-1}$, where $[\mathbf{t}]_{\mathbf{x}}$ is the skew-symmetric matrix of the translations vector **t**.

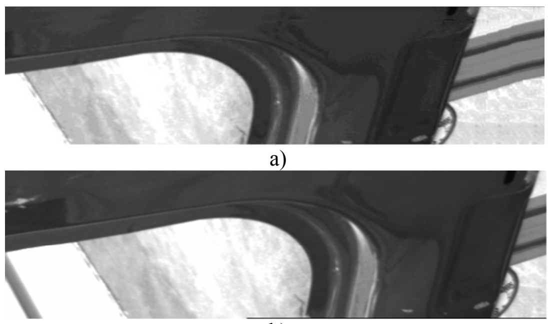

Fig. 2 The stereo system images: a) the left image; b) the right image

The $(\mathbf{x}_i, \mathbf{x'}_i)$ pairs of the correspondent edge points in the two images of the stereo system are computed as follows:

1. The edges are extracted in each image;
2. For each edge pixel **x** of the left image:
   2.1. Compute the epipolar line in the right image: $\mathbf{l} = \mathbf{F}x$.
   2.2. The correspondent pixel of **x** in the right image is computed as the intersection of **l** with the right image contour.

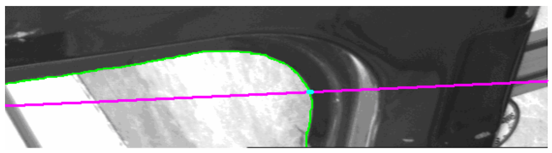

Fig. 3 Epipolar line in the right image of an edge pixel **x** of the left image. The correspondent pixel **x'** in the right image is the intersection of the epipolar line of **x** with the right contour chosen as feature

The 3D edge points $\mathbf{X}_i(X_i, Y_i, Z_i)$ with $\mathbf{P}\mathbf{X}_i = \mathbf{x}_i$ and $\mathbf{P'}\mathbf{X}_i = \mathbf{x'}_i$ are calculated from the pairs $(\mathbf{x}_i, \mathbf{x'}_i)$, using the well known triangulation method illustrated by Fig. 4.

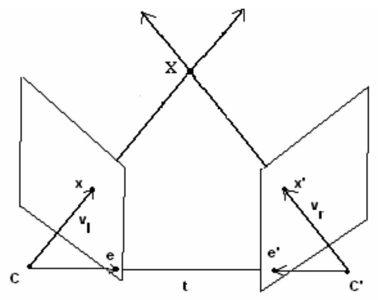

Fig. 4 Triangulation method. Due the fact the correspondent pixels $(\mathbf{x}, \mathbf{x'})$ satisfy the epipolar constraint, the rays of **x** and **x**' intersect in space point **X**.

For a Tsai like camera calibration, [6], the conversion from the pixel **x**(x, y) of the left image to the vector $\mathbf{v}_l(x_l, y_l, z_l)$ of the left camera system is:

$$
\begin{aligned}
x_d &= \frac{d'_x}{s_x}(x - C_x) \\
y_d &= d_y(y - C_y) \\
x_u &= x_d(1 - k_1(x_d^2 + y_d^2)) \\
y_u &= y_d(1 - k_1(x_d^2 + y_d^2)) \\
x_l &= x_u \\
y_l &= y_u \\
z_l &= f
\end{aligned}
\qquad (1)
$$

where (x_u, y_u) are the undistorted coordinates, (x_d, y_d) distorted coordinates of the pixel and $d'_x, C_x, C_y, s_x, d_y, f, k_1$ are the intrinsic calibration parameters.

The vector $\mathbf{v}_r(x_r, y_r, z_r)$ of the left image pixel $\mathbf{x'}(x', y')$ is calculated the same way.

Usually, the 3D edge points $\mathbf{X}_i(X_i, Y_i, Z_i)$ are computed using triangulation method, [5], expressed by:

$$a\mathbf{v}_l - b\mathbf{R}^t\mathbf{v}_r + c(\mathbf{v}_l x \mathbf{R}^t\mathbf{v}_r) = \mathbf{t} \qquad (2)$$

Due the fact the pairs $(\mathbf{x}_i, \mathbf{x'}_i)$ satisfy the epipolar condition $(\mathbf{x'}^t_i \mathbf{F}\mathbf{x}_i = 0)$ the rays of the **x** and **x'** pixels will intersect in 3D space. So, the vector $(\mathbf{v}_r x \mathbf{R}^t \mathbf{v}_r)$, perpendicular on the rays of **x** and **x'**, is of null length. Equation (2) becomes:

$$a\mathbf{v}_l - b\mathbf{R}^t\mathbf{v}_r = \mathbf{t} \qquad (3)$$

Let a_i and b_i be the solutions of (3) for the pairs $(\mathbf{x}_i, \mathbf{x'}_i)$. Let the left camera system vectors of the $\mathbf{x}_i$ pixels be noted with $\mathbf{v}_{li}(x_{li}, y_{li}, z_{li})^t$. Using also the notations $\mathbf{R}^t\mathbf{v}_{ri} = (vx_i, vy_i, vz_i)$, $\mathbf{t} = (t_x, t_y, t_z)^t$ the point $\mathbf{X}_i$ coordinates are:

$$
\begin{aligned}
X_i &= (a_i x_{li} + t_x + b_i vx_i)/2; \\
Y_i &= (a_i y_{li} + t_y + b_i vy_i)/2; \\
Z_i &= (a_i z_{li} + t_z + b_i vz_i)/2.
\end{aligned}
\qquad (4)
$$

Let $\mathbf{X}_{mi}(X_{mi}, Y_{mi}, Z_{mi}), i = 1, M$, be the 3D edge points from training step and $\mathbf{X}_{di}(X_{di}, Y_{di}, Z_{di}), i = 1, N$, the edge points extracted at the execution phase. Consider (**R**, **t**) the Euclidean transform that registers the $\mathbf{X}_{di}$ points over the $\mathbf{X}_{mi}$ edge points:

$$\mathbf{R} = \begin{bmatrix} \cos\phi\cos\theta & \cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \sin\phi\sin\theta & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi \\ -\sin\theta & \cos\theta\sin\psi & \cos\theta\cos\psi \end{bmatrix} \qquad (5)$$

$$\mathbf{t} = (t_x, t_y, t_z)^t$$

where $\psi, \theta, \text{and } \phi$ are the rotation angles about OX, OY, respectively OZ axes and **t** is the translation vector.

Starting from an initial transform $(\mathbf{R}_0, \mathbf{t}_0)$ a Levenberg-Marquardt optimization method computes a Euclidean transformation $(\mathbf{R}, \mathbf{t})$ which registers 3D edge points calculated at the execution step on the 3D points of the training phase.

For each contour point $\mathbf{X}_{di}(X_{di}, Y_{di}, Z_{di}), i = 1, N$, an edge point $\mathbf{X}_{mk_i}(X_{mk_i}, Y_{mk_i}, Z_{mk_i})^t$ of the trained contour $\mathbf{X}_{mj}(X_{mj}, Y_{mj}, Z_{mj})^t$, $j = 1, M$, is chosen according with:

$$d_i = \|\mathbf{X}_{ti} - \mathbf{X}_{mk_i}\| = \|(\mathbf{R}_c\mathbf{X}_{di} + \mathbf{t}_c) - \mathbf{X}_{mk_i}\| = \min_{j=1,M}\|(\mathbf{R}_c\mathbf{X}_{di} + \mathbf{t}_c) - \mathbf{X}_{mj}\| \quad (6)$$

where $(\mathbf{R}_c, \mathbf{t}_c)$ is the current transform, $\mathbf{X}_{ti} = \mathbf{R}_c\mathbf{X}_{di} + \mathbf{t}_c$.

The transform computed by Levenberg-Marquardt method minimizes the following cost function:

$$E(\psi,\theta,\phi,t_x,t_y,t_z) = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N}\|(\mathbf{R}\mathbf{X}_{di} + \mathbf{t}) - \mathbf{X}_{mk_i}\| \quad (7)$$

For N edge points, $\mathbf{X}_{di}(X_{di}, Y_{di}, Z_{di})$, computed at the execution phase, the Jacobian of $E(\psi,\theta,\phi,t_x,t_y,t_z)$ is an Nx6 matrix:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial d_1}{\partial X_1}\frac{\partial X_1}{\partial \psi} + \frac{\partial d_1}{\partial Y_1}\frac{\partial Y_1}{\partial \psi} + \frac{\partial d_1}{\partial Z_1}\frac{\partial Z_1}{\partial \psi} & \frac{\partial d_1}{\partial X_1}\frac{\partial X_1}{\partial \theta} + \frac{\partial d_1}{\partial Y_1}\frac{\partial Y_1}{\partial \theta} + \frac{\partial d_1}{\partial Z_1}\frac{\partial Z_1}{\partial \theta} & \frac{\partial d_1}{\partial X_1}\frac{\partial X_1}{\partial \phi} + \frac{\partial d_1}{\partial Y_1}\frac{\partial Y_1}{\partial \phi} + \frac{\partial d_1}{\partial Z_1}\frac{\partial Z_1}{\partial \phi} & \frac{\partial d_1}{\partial X_1} & \frac{\partial d_1}{\partial Y_1} & \frac{\partial d_1}{\partial Z_1} \\ \frac{\partial d_N}{\partial X_N}\frac{\partial X_N}{\partial \psi} + \frac{\partial d_N}{\partial Y_N}\frac{\partial Y_N}{\partial \psi} + \frac{\partial d_N}{\partial Z_N}\frac{\partial Z_N}{\partial \psi} & \frac{\partial d_N}{\partial X_N}\frac{\partial X_N}{\partial \theta} + \frac{\partial d_N}{\partial Y_N}\frac{\partial Y_N}{\partial \theta} + \frac{\partial d_N}{\partial Z_N}\frac{\partial Z_N}{\partial \theta} & \frac{\partial d_N}{\partial X_N}\frac{\partial X_N}{\partial \phi} + \frac{\partial d_N}{\partial Y_N}\frac{\partial Y_N}{\partial \phi} + \frac{\partial d_N}{\partial Z_N}\frac{\partial Z_N}{\partial \phi} & \frac{\partial d_N}{\partial X_N} & \frac{\partial d_N}{\partial Y_N} & \frac{\partial d_N}{\partial Z_N} \end{bmatrix} \quad (8)$$

where:

$$X_i = (\cos\phi\cos\theta)X_{di} + (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)Y_{di} + $$
$$+ (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)Z_{di} + t_x$$
$$Y_i = (\sin\phi\cos\theta)X_{di} + (\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi)Y_{di} + $$
$$+ (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)Z_{di} + t_y$$
$$Z_i = (-\sin\theta)X_{di} + (\cos\theta\sin\psi)Y_{di} + (\cos\theta\cos\psi)Z_{di} + t_z.$$

The $\psi,\theta,\phi,t_x,t_y,t_z$ values minimizing the cost function (7) are computed in each step of the optimization process by solving the equation:

$$(\mathbf{J}^t\mathbf{J} + \lambda\,\mathbf{I})\Delta = -\mathbf{J}^t\boldsymbol{\varepsilon} \quad (9)$$

where $\mathbf{J}$ is Jacobian (8), λ is a real parameter varying from step to step according to Levenberg-Marquardt method, $\mathbf{I}$ is unit matrix and $\boldsymbol{\varepsilon}$ is:

$\boldsymbol{\varepsilon} = [d_1, d_2, \ldots d_N]^t$.

The solutions $\Delta = (\Delta\psi, \Delta\theta, \Delta\phi, \Delta t_x, \Delta t_y, \Delta t_z)$ lead to minimization of the cost function (7) using the new parameter values:

$\psi_{i+1} = \psi_i + \Delta\psi$; $\theta_{i+1} = \theta_i + \Delta\theta$; $\phi_{i+1} = \phi_i + \Delta\phi$; $t_{x(i+1)} = t_{xi} + \Delta t_x$; $t_{y(i+1)} = t_{yi} + \Delta t_y$; $t_{z(i+1)} = t_{zi} + \Delta t_z$.

Before registration, the edge points $\mathbf{X}_{mi}(X_{mi}, Y_{mi}, Z_{mi}), i = 1, M$ and $\mathbf{X}_{di}(X_{di}, Y_{di}, Z_{di}), i = 1, N$ are scaled to fit in a cube 100x100x100 as depicted by Fig. 5. The scaling method is described in [2].

The registration result is figured in Fig. 6.

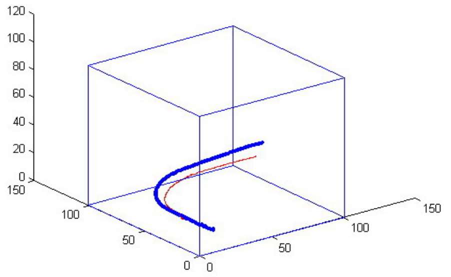

Fig. 5 Training 3D edge (red) and edge extracted in the execution step (blue)

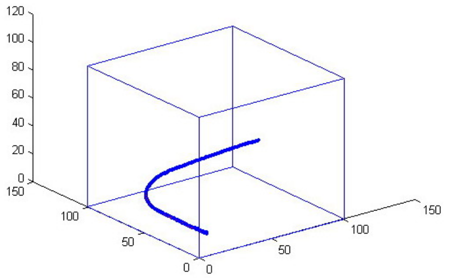

Fig. 6 The registration algorithm result. The training edge overlaps the edge extracted in the execution step

A problem for great volume data applications could be the computation of Nx6 matrix $\mathbf{J}$. The members $\mathbf{J}^t\mathbf{J}$ and $\mathbf{J}^t\boldsymbol{\varepsilon}$ of $(\mathbf{J}^t\mathbf{J} + \lambda\,\mathbf{I})\Delta = -\mathbf{J}^t\boldsymbol{\varepsilon}$ equation may be computed as follows.

Using the notations $\mathbf{J}^t\mathbf{J} = [a_{ij}]$, with i, j =1,6; $\mathbf{J} = [J_{kl}]$ with l=1,6, k = 1,N and $\mathbf{J}^t\boldsymbol{\varepsilon} = [e_1, e_2, \ldots e_6]^t$, a_{ij} and e_i are:

$$a_{ij} = \sum_{l=1}^{N} J_{li}J_{lj}; \quad e_i = \sum_{l=1}^{N} J_{li}d_l\,. \quad (10)$$

After considering k execution edge points $\mathbf{X}_{di}(X_{di}, Y_{di}, Z_{di})$, let the partial sums in (10) be noted as:

$$a_{ij}^k = \sum_{l=1}^{k} J_{li}J_{lj}; \quad e_i^k = \sum_{l=1}^{k} J_{li}d_l\,\cdot$$

For the point $\mathbf{X}_{dk+1}(X_{dk+1}, Y_{dk+1}, Z_{dk+1})$, a_{ij}^{k+1} and e_i^{k+1} are:

$$a_{ij}^{k+1} = a_{ij}^k + J_{(k+1)i}J_{(k+1)j};\; e_i^{k+1} = e_i^k + J_{(k+1)i}d_{k+1};\; i,j = 1,6 \quad (11)$$

For the current edge point $\mathbf{X}_k(X_k, Y_k, Z_k)$, in previous relations, the Jacobian components are:

$$\mathbf{J_{kl}} = \frac{\partial d_k}{\partial X_k}\frac{\partial X_k}{\partial \psi} + \frac{\partial d_k}{\partial Y_k}\frac{\partial Y_k}{\partial \psi} + \frac{\partial d_k}{\partial Z_k}\frac{\partial Z_k}{\partial \psi}$$

$$\mathbf{J_{k2}} = \frac{\partial d_k}{\partial X_k}\frac{\partial X_k}{\partial \theta} + \frac{\partial d_k}{\partial Y_k}\frac{\partial Y_k}{\partial \theta} + \frac{\partial d_k}{\partial Z_k}\frac{\partial Z_k}{\partial \theta}$$

$$\mathbf{J_{k3}} = \frac{\partial d_k}{\partial X_k}\frac{\partial X_k}{\partial \phi} + \frac{\partial d_k}{\partial Y_k}\frac{\partial Y_k}{\partial \phi} + \frac{\partial d_k}{\partial Z_k}\frac{\partial Z_k}{\partial \phi}$$

$$\mathbf{J_{k4}} = \frac{\partial d_k}{\partial X_k}; \quad \mathbf{J_{k5}} = \frac{\partial d_k}{\partial Y_k}; \quad \mathbf{J_{k6}} = \frac{\partial d_k}{\partial Z_k}.$$

III. CONCLUSIONS

The registration process is an $O(n^2)$ algorithm, with n=max(M,N). It is acceptable for applications with edges of hundreds of points. For applications involving a great amount of edge data it is recommended the registration method presented in [2].

The method was tested for robot vision applications on various objects. The precision of head robot pose is the same like in single camera robot applications. The comparison was done using accuracy tests for the two methods applied on different objects and also applied on the same object. This was possible because the proposed method works also for objects suited in single view robot applications.

REFERENCES

[1] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2004, Second Edition.
[2] A. Fitzgibbon, Robust Registration of 2D and 3D point sets, Image and Vision Computing 2003, BMVC 2001.
[3] D Lowe, Fitting parameterized three-dimensional models to images, IEEE PAMI 13(5): 441-450, May 1991.
[4] F Martin, R.Horaud, Multiple-camera tracking of rigid objects, International Journal of Robotics Research, Vol. 21, No.2, pp.97-113, February 2002.
[5] E. Trucco, A. Verri, Introductory techniques for 3D Computer Vision, Prentice Hall, 1998.
[6] R.Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, pp. 323-344.