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

The \((x_i, x'_i)\) 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: \(l_x = y_x (y - c_x)\);
   2.2. The correspondent pixel of \(x\) in the right image is computed as the intersection of \(l_x\) with the right image contour.

The \(3D\) edge points \((X,Y,Z)\) with \(x_P = c_x\) and \(x'_P = c_x'\) are calculated from the pairs \((x,x')\), using the well known triangulation method illustrated by Fig. 4.

For a Tsai like camera calibration, \([6]\), the conversion from the pixel \((x,y)\) of the left image to the vector \((z,y,x)\) of the left camera system is:

\[
x_x = \frac{d_x}{s_x} (x - C_x)
\]
\[
y_y = \frac{d_y}{s_y} (y - C_y)
\]
\[
x_x = x_y (1 - k_x (x_x^2 + y_y^2))
\]
\[
y_y = y_x (1 - k_x (x_x^2 + y_y^2))
\]
\[
x_x = x_x
\]
\[
y_y = y_y
\]
\[
z_x = f
\]

where \((x_x,y_y)\) are the undistorted coordinates, \((x_x',y_y')\) distorted coordinates of the pixel and \(d_x, C_x, C_y, d_y, f, k_x\) are the intrinsic calibration parameters.

The vector \(v_i(x,y,z)\) of the left image pixel \((x',y')\) is calculated the same way.

Usually, the \(3D\) edge points \((X,Y,Z)\) are computed using triangulation method, \([5]\), expressed by:

\[
\begin{align*}
v_i & = -e R v_i + c(v_i R v_i) = t \\
\end{align*}
\]

Due the fact the pairs \((x,x')\) satisfy the epipolar condition \((x'Fx = 0)\) the rays of the \(x\) and \(x'\) pixels will intersect in \(3D\) space. So, the vector \((v_i X v_i R v_i)\), perpendicular on the rays of \(x\) and \(x'\), is of null length. Equation (2) becomes:

\[
\begin{align*}
v_i & = -e R v_i = t \\
\end{align*}
\]

Let \(a\) and \(b\) be the solutions of (3) for the pairs \((x,x')\).

Let the left camera system vectors of the \(x\) pixels be noted \(x_P = (x_x, y_y, z_z)\). Using also the notations \(x'_P = (x'_x, y'_y, z'_z)\), \(t = (t_x, t_y, t_z)\) the point \(X\) coordinates are:

\[
\begin{align*}
X_x & = (a x_x + t_x + b y_x) / 2; \\
Y_y & = (a y_y + t_y + b y_y) / 2; \\
Z_z & = (a z_z + t_z + b y_z) / 2. \\
\end{align*}
\]

Let \(X_n(X,n_z, Z_m, Z_m)\), \(i = 1,M\), be the \(3D\) edge points from training step and \(X_n(X_n, Y_n, Z_n)\), \(i = 1,N\), the \(3D\) edge points extracted at the execution phase. Consider \((R,t)\) the Euclidean transform that registers the \(X_n\) points over the \(X_{ns}\) edge points:

\[
R = \begin{bmatrix}
cos \psi cos \theta & cos \phi sin \psi & sin \phi sin \psi & -sin \theta & -sin \phi cos \theta \\
cos \psi sin \theta & sin \phi cos \psi & -cos \phi cos \psi & cos \phi sin \psi & cos \theta \\
-sin \psi & cos \phi & sin \phi & 0 & 0 \\
cos \theta & -sin \phi & sin \phi & 0 & 0 \\
0 & cos \phi & -sin \phi & 0 & 0 \\
\end{bmatrix}
\]

\[
t = (t_x, t_y, t_z)\]

where \(\psi, \theta, \phi\) are the rotation angles about \(OX, OY\), respectively \(OZ\) axes and \(t\) is the translation vector.

Starting from an initial transform \((R_0, t_0)\) a Levenberg-Marquardt optimization method computes a Euclidean transformation \((R,t)\) which registers \(3D\) edge points calculated at the execution step on the \(3D\) points of the training phase.

For each contour point \(X_n(X_n, Y_n, Z_n)\), \(i = 1,N\), an edge
point $X_{nk} (X_{nk}, Y_{nk}, Z_{nk})'$ of the trained contour $X_{nk}(X_{nk}, Y_{nk}, Z_{nk})^*$, $j = 1, M$, is chosen according with:

$$d_j = |X_{nk} - X_{nk}'| = |R X_{nk} + t| - X_{nk}'$$

(6)

where $(R, t)$ is the current transform, $X_{nk} = R X_{nk} + t$.

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

$$E(\psi, \phi, t, t, t, t) = \sum_{j=1}^{N} d_j = \sum_{j=1}^{N} |R X_{nk} + t| - X_{nk}'$$

(7)

For $N$ edge points, $X_{nk}(X_{nk}, Y_{nk}, Z_{nk})$, computed at the execution phase, the Jacobian of $E(\psi, \phi, t, t, t, t)$ is an $N \times 6$ matrix:

$$J = \begin{bmatrix}
\frac{\partial E}{\partial \psi} & \frac{\partial E}{\partial \phi} & \frac{\partial E}{\partial t} & \frac{\partial E}{\partial t} & \frac{\partial E}{\partial t} & \frac{\partial E}{\partial t}
\end{bmatrix}
$$

(8)

where:

$$X_i = (\cos \psi \cos \theta) X_{di} + (\cos \psi \sin \theta \sin \psi - \sin \psi \cos \theta) Y_{di} +$$

$$+ (\cos \theta \sin \psi) Z_{di} + t_x$$

$$Y_i = (\sin \psi \cos \theta) X_{di} + (\sin \psi \sin \theta \sin \psi + \sin \psi \cos \theta) Y_{di} +$$

$$+ (\sin \theta \sin \psi) Z_{di} + t_y$$

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

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

$$(J^T J + \lambda I) \Delta = -J^T e$$

(9)

where $J$ is Jacobian (8), $\lambda$ is a real parameter varying from step to step according to Levenberg-Marquardt method, $I$ is unit matrix and $e$ is:

$$e = [d_1, d_2, \ldots, d_N]^T.$$

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

$$\psi_{i+1}^* = \psi_i + \Delta \psi; \quad \theta_{i+1}^* = \theta_i + \Delta \theta; \quad \phi_{i+1}^* = \phi_i + \Delta \phi; \quad t_{i+1}^* = t_i + \Delta t;$$

$$t_{i+1}^* = t_i + \Delta t; \quad t_{i+1}^* = t_i + \Delta t.$$

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

The registration result is figured in Fig. 6.
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