VISUAL TRACKING FOR MOVING MULTIPLE OBJECTS: AN INTEGRATION OF VISION AND CONTROL

Metin Sitti, Işıl Bozma, and Ahmet Denker

Abstract — This paper addresses the use of a vision sensor in the feedback loop for tracking an object which is selected from multiple objects that are unknown and randomly moving on a conveyor belt. For achieving such a tracking task, a position-based visual tracking system is proposed. This system has the visual processing part in which multiple objects are segmented. Here we address the segmentation problem as a clustering problem. In its control part, PD and self-tuning controllers are designed and compared with respect to their tracking speed and noise immunity performances. As the tracking device a pan-tilt camera mount is proposed at first and the performance of the proposed visual servoing system is evaluated by the simulations. Then the real-time applicability of the introduced vision system is experimented on Manutec R15 robotic manipulator in picking parts on a conveyor belt and the results show that the introduced system can be used as a real-time system.

1 INTRODUCTION[1]

We have built a system that tracks an unknown randomly moving object which is selected from a group of objects on a conveyor belt as shown in Fig. 1. There are no assumptions regarding object shape, position in the conveyor plane, orientation and velocity. The tracking algorithm underlying this behavior relies on a continuous stream of object position and velocity estimates delivered by a camera system. Our visual servoing problem is to control the *pose* of a camera -based on information obtained from processing visual feedback- so that the camera tracks an object located on a conveyor belt. By tracking, it is meant that the intersection of the optical axis of the camera with the image plane corresponds to an *a priori* defined "characterizing" point of the object.

For the problem of visual servoing, some image features should be selected for getting the object position and velocity information. First main approach in selecting the image feature is using the image brightness values and computing motion from the optical flow. Papanikolopoulos *et al.* [1] use the sum-of-squared differences optical flow for the computation of the vector of discrete displacements within multiple small windows and Luo *et al.* [2] do a modified version of Horn-Schunk [3] algorithm. Allen *et al.* [4] use stereo cameras for computing the the optical flow and from these flow fields, a motion energy profile is obtained that is utilized in recovering the 3-D position of a moving object.

[1]The authors are with the Department of Electrical and Electronics Engineering, Boğaziçi University, 80815, İstanbul and M. Sitti and A. Denker are also with TÜBİTAK Marmara Research Center, CAD/CAM Robotics Department, Kocaeli, Turkey.

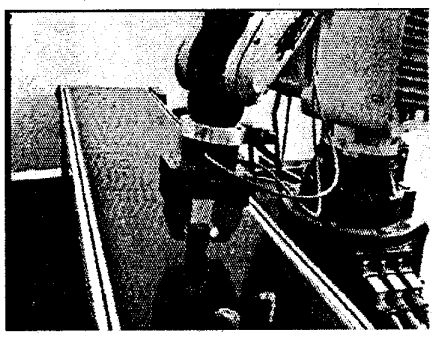

Fig. 1. Conveyor belt and robot setup.

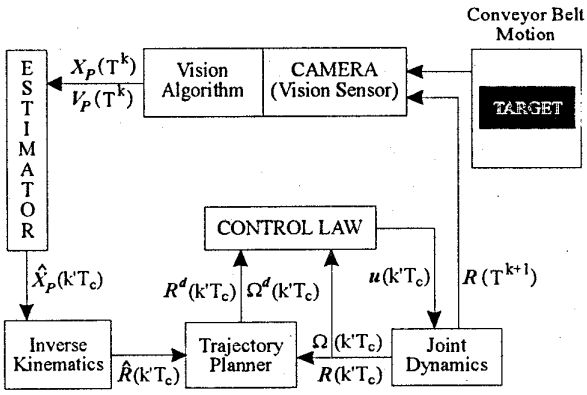

Fig. 2. Architecture of the visual tracking system.

The other approach is getting a characterizing feature of objects that is unambiguous in each image frame and most preferred feature is the centroid of the objects. Koivo *et al.* [5] compute the centroid positions of a rectangular object after preprocessing and get the object velocities. Hunt and Sanderson [6] present algorithms for visual tracking based on mathematical prediction of the object centroids.

As different from the previous works on tracking or grasping objects [1]-[6] where the image processing is simplified by assuming a single object in the scene, a real-time vision system with multiple objects in the scene is introduced. This is accomplished by a modified version of agglomerative hierarchical clustering algorithm that is used for segmenting the incoming image data. PD and self-tuning control laws are compared with respect to their tracking speed and noise immunity in the tracking control of a pan-tilt camera mount joints using the vision information.

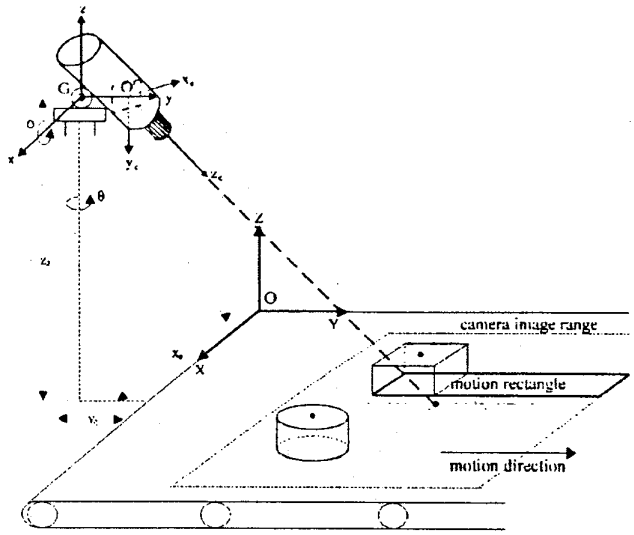

Fig. 3. Camera setup with coordinate frames.

1.1 Solution Approach

We use a visual tracking system as shown in Fig. 2. The visual data from the camera is processed in order to compute the position $\mathbf{x_p}(T^k) \in \mathcal{R}^2$ and velocity $\mathbf{v_p}(T^k) \in \mathcal{R}^2$ of a target object in the image frame coordinates where $T^k = \sum_{i=1}^{k} T_v^i$ and $T_v^i \in \mathcal{R}$ is the visual processing time between $(i-1)^{th}$ and i^{th} frames. Assuming a priori knowledge of approximate depth and using the inverse perspective transformation, these values are mapped to the target feature positions $\mathbf{X_p}(T^k) \in \mathcal{R}^3$ and velocities.$\mathbf{V_p}(T^k) \in \mathcal{R}^3$ Due to digitization and processing latency, the image measurements generated by the visual processsing section are results that are at least T_v^k seconds old. It follows that we ought to construct an estimator which operates on this delayed data and the position values are predicted one-step-ahead with the intervals of T_c where $T_c \in \mathcal{R}$ is the constant control sampling time. Assuming $T_v^k > T_c$ relation holds everytime, the estimator provides the positions of $\hat{\mathbf{X}}_\mathbf{p}(k'T_c)$ where $k' = 1, \ldots, m^k$, $m^k = T_v^k/T_c$, m^k is an integer, and estimating positions with the intervals of T_c enables the generation of a smooth trajectory for the joints. For the simplicity of the notation $k'T_c$ will be denoted as k' in the sequel. The estimated values are mapped to camera mount joint angles utilizing the inverse kinematics of the camera mount as $\hat{\mathbf{R}}(k') \in \mathcal{R}^2$. These angles correspond to the subgoal positions $\mathbf{R^d}(k')$ for the trajectory planner and the subgoal joint velocities $\mathbf{\Omega^d}(k') \in \mathcal{R}^2$ are computed using the first-order Taylor Series approximator. A trajectory planner adjusts the required joint velocities and implied joint acceleration within the limits of the joint motors limits. A controller generates the necessary inputs $\mathbf{u}(k') \in \mathcal{R}^2$ for tracking these reference values and these inputs move the joints to a new position as $\mathbf{R}(T^{k+1})$ at the end.

2 SETUP

A a pan-tilt camera mount shown in Fig. 3 is used. Consider a target that moves in a plane with a feature, located at point $\mathbf{P}$, that we want to track. The projection of $\mathbf{P}$ on the image plane is the point $\mathbf{p} \in \mathcal{R}^2$. Consider also a neighborhood $\mathcal{N}_\mathbf{m}$ of $\mathbf{P}$ in the conveyor belt plane and that $\mathcal{N}_\mathbf{i}$ of $\mathbf{p}$ in the image plane. The problem of 2-D visual tracking of a single feature point is defined as: Find the camera rotation vector $\mathbf{R} = (\mathbf{R_1}, \mathbf{R_2})$ with respect to the camera frame that keeps $\mathcal{N}_\mathbf{i}$ stationary in the area $\mathcal{N}_\mathbf{m}$ around the origin of the image frame. There is no need to know the depth of the point $\mathbf{P}$ since a 2-D tracking task is proposed. Taking the target feature at the point $\mathbf{P}$ as its centroid point, $\mathbf{P}$ is assumed to have no height on the motion plane hereafter. Furthermore, at the outset of the tracking process, there is no condition of intersection of $\mathcal{N}_\mathbf{i}$ with $\mathcal{N}_\mathbf{m}$. From the position of the point $\mathbf{p}$ at time T^k $\mathbf{x_p}(T^k)$, the velocity $\mathbf{v_p}(T^k)$ is computed using the first-order Taylor Series approximation as

$$\mathbf{v_p}(T^k) = \frac{\mathbf{x_p}(T^k) - \mathbf{x_p}(T^{k-1})}{T_v^k} . \tag{1}$$

Equation (1) will be used with the assumption that the motion of the object feature is smooth and continuous and the vision sampling period T_v can change from one frame to another in our system.

2.1 Perspective Transformation

The positions $\mathbf{p}(T^k) = (x(T^k)\ y(T^k))^T$ obtained from the vision algorithm are in the camera image plane coordinates and they are mapped to the motion plane coordinates as $\mathbf{P}(T^k) = (X(T^k)\ Y(T^k)\ Z(T^k))^T$, where $Z(T^k) = 0$ is taken using the perspective transformation which is given as [7]

$$x(T^k)\Delta = f[(X(T^k) - X_0)\cos\theta + (Y(T^k) - Y_0)\sin\theta - l_1] , \tag{2}$$

$$y(T^k)\Delta = f[(X(T^k) - X_0)\sin\phi\sin\theta - (Y(T^k) - Y_0)4 \sin\phi\cos\theta + (Z(T^k) - Z_0)\cos\phi - l_3] , \tag{3}$$

where $\Delta = -(X(T^k) - X_0)\cos\phi\sin\theta + (Y(T^k) - Y_0)\cos\phi\cos\theta + (Z(T^k) - Z_0)\sin\phi - l_2$, $(X_0\ Y_0\ Z_0)^T$ is the motion plane coordinates of the camera stand gimbal center G as shown in Fig. 3, the vector $(l_1\ l_2\ l_3)^T$ is from G to the image plane center O', and f is the camera focal length.

2.2 Dynamical Model of the Camera Stand

The physical setup of the camera stand consists of the pan and tilt rotations along two joints of the camera mount which include DC motors. Taking two motors independent of each other, the plant function of the joints can be derived as [7]

$$T_m^1\ddot{\theta}(t) + \theta(t) = K_m^1 u_\theta , \tag{4}$$

$$T_m^2\ddot{\phi}(t) + \phi(t) = K_m^2 u_\phi , \tag{5}$$

where K_m^i and T_m^i, $i = 1, 2$, stand for *motor gain constant* and *motor time constant* of each DC motor successively and u_θ and u_ϕ are input joint torques or voltages.

2.3 Inverse Kinematics of the Camera Stand

As $\hat{\mathbf{X}}_\mathbf{P}(k') = (\hat{X}(k')\ \hat{Y}(k')\ 0)^T$ refer to the estimated positions in the motion plane coordinates, they are mapped to camera mount joint angles $\hat{\mathbf{R}}(k') = (\hat{\theta}(k')\ \hat{\phi}(k'))^T$ utilizing the inverse kinematics of the camera mount such that

$$\hat{\theta}(k') = tan^{-1}\left(\frac{\hat{Y}(k') - Y_0}{\hat{X}(k') - X_0}\right), \quad (6)$$

$$\hat{\phi}(k') = tan^{-1}\left(\frac{\sqrt{(\hat{X}(k') - X_0)^2 + (\hat{Y}(k') - Y_0)^2}}{-Z_0}\right) \quad (7)$$

2.4 Trajectory Planner

The task of the trajectory planner is to generate the subgoal/desired points $\mathbf{\Omega^d}[(k'+1)T_c]$ and $\mathbf{R^d}[(k'+1)T_c]$ for the controller such that a stable and smooth motion of the camera mount joints can be obtained. Since the trajectory points are determined in every T_c by the estimator using the vision information, there is no need to generate a smooth trajectory here; but, the generated trajectories are checked in terms of the motor maximum velocity and acceleration limits, and adjusted accordingly [7].

3 GENERATING VISUAL FEEDBACK

There are multiple objects on the conveyor belt and the one among them which is closest to the camera needs to be tracked. In order to accomplish this, three features of the objects are computed: *centroids* which are used in generating subgoal trajectory points for the tracking control, *enclosing rectangles* that enable the selection of the target object and the determination *motion rectangle* - the elongated area of the enclosing rectangle in the conveyor belt motion direction - which is utilized in local window operation for a less amount of computation. The conveyor motion direction is assumed to be from left to right hereafter.

The segmentation problem is posed as a clustering problem. The set $\mathbf{X}$ of N pixels constitute the sample points $\mathbf{s}_i$, $i = 1, \ldots, N$ that need to be clustered into L disjoint subsets $\mathbf{X}^1, \ldots, \mathbf{X}^L$ where the resulting clusters constitute the unknown objects in the camera image frame, and the centroids of each object k $\mathbf{x_p}^k$ is the center of each cluster as

$$\mathbf{x_p}^k = \frac{1}{n^k} \sum_{\mathbf{s}_i \in \mathbf{X}^k} \mathbf{s}_i , \quad (8)$$

where $n^k \in \mathcal{R}$ is the number of samples in the cluster k and $k = 1, \ldots, L$.

In order to expedite the processing, a lower resolution image is used and the edge pixels of an image - obtained via Sobel operators and thresholded with $T_e \in \mathcal{R}$ - are used as the samples to be clustered. Let us suppose that N edge samples remain. Edges in the first image frame are first roughly clustered using initial clustering algorithm that is a nearest centroid classification approach without any iterative optimization [8]. Assuming that the objects do not overlap, the algorithm assigns each sample $\mathbf{s}_i$ to the cluster $\mathbf{X}^l$ where the relation $|\mathbf{s}_i - \mathbf{X}^l| < T_d$ is valid and $T_d \in \mathcal{R}$ is the distance threshold. This algorithm works well for objects which are similar in size, well-separated and have rotationally symmetric locations. However, in a conveyor belt system, these conditions may not be satisfied where there is no *a priori* knowledge about the object location and geometry. Therefore, in the initial clustering part, T_d is selected to have a small value so that any edge pixels of objects with different sizes and close locations are not grouped in the same cluster. Then the resulting clusters can belong to the same object and clusters belonging to the same object are merged using a modified agglomerative hierarchical clustering algorithm. As different from the agglomerative (bottom-up) hierarchical clustering (AHC) technique in where N samples are merged hierarchically, we merge clusters X^l, $l = 1, \ldots, L$, that result from the initial clustering algorithm according to the rule: If $\exists \mathbf{s_i} \in \mathbf{X^k}$, $\exists \mathbf{s_j} \in \mathbf{X^l}$ such that $|\mathbf{s}_i - \mathbf{s}_j| < T_m$, then merge X^k and X^l, where $T_m \leq T_d \in \mathcal{R}$ is the merging threshold.

Letting $\hat{L}$ denotes for the remaining number of clusters after merging, the modified AHC algorithm becomes as

Step 1. Let $\hat{L} = L$, $k = 0$.
Step 2. If any $\mathbf{s}_i \in X^m$ and $\mathbf{s}_j \in X^n$ obeys the merging rule, $k = k + 1$, merge X^m and X^n into the cluster X^k, and delete X^m and X^n from $\mathbf{X}$ where $m = 1, \ldots, \hat{L}-1, n = m+1, \ldots, \hat{L}$.
Step 3. If any merging is done, $\hat{L} = k$, $k = 0$, and go to Step 2; otherwise, compute $\mathbf{x_p}^l$ of the resulting clusters by the equation (8) where $l = 1, \ldots, \hat{L}$ and stop.

Resulting merged $\hat{L}$ clusters constitute the objects at the image. Then, enclosing rectangle features of each object are computed. The object having the largest right enclosing rectangle side is selected as the target t. Since the target enclosing rectangle is in the reduced size image, it is approximated in the original size by multiplying rectangle coordinates by four. Correcting these coordinates by the algorithm given in [7], accurate centroid is calculated by averaging the edge pixels in the new rectangle.

The motion rectangle of the target at k^{th} frame is computed and resulting rectangle region is denoted as A_m^k. Letting the enclosing rectangle region that the target will be at T_v^{k+1} seconds later is A_e^{k+1}, it is assumed that

$$A_e^{k+1} \subseteq A_m^k . \quad (9)$$

Then the target object can be searched in the region A_m^k in the $(k+1)^{th}$ image frame which enables a local window search and reduces the computation time significantly. For a moving camera such as in our case, this relation is valid only after direct and inverse transformations given in [7]; the computed A_m^k corresponds to another region $\hat{A}_m^k$ in the image plane T_v^{k+1} seconds later.

Assuming no object enters to the front of the target object in the region $\hat{A}_m^k$ and letting the enclosing rectangle coordinates as $(x_1{}^k, y_1{}^k)$ and $(x_2{}^k, y_2{}^k)$ in the k^{th} frame, the motion rectangle algorithm at k^{th} frame is as follows:

Step 1. $\Delta x^k = x_2^{k-1} - x_1^{k-1}$ is known from $k-1^{th}$ image frame.
Step 2. Find the first vertical i that crosses the target edge pixels in the region $\hat{A}_m^k$ by scanning vertically from the end of $\hat{A}_m^k$ to the beginning; $x_2^{k+1} = i$ and x_1^{k+1} is approximated as $x_2^{k+1} - \Delta x^k$.
Step 3. Apply the correction algorithm [7] for finding the accurate x_1^{k+1}, y_1^{k+1} and y_2^{k+1}.
Step 4. Compute $\mathbf{x_p}^t(T^{k+1})$ and $\mathbf{v_p}(T^{k+1})$ from the equations (12) and (1) respectively.

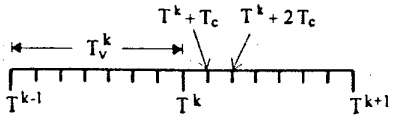

Fig. 4. Vision and control sampling periods on time axis.

4 ESTIMATION OF THE POSITIONS

In order to compensate for the inherent delay in the visual feature computations, the positions of the target object are predicted $(T^k + m^k T_c)$ ahead for the trajectory planner based on the available information at time T^{k-1}. Fig. 4 illustrates the relation between the vision and control sampling periods.

The estimation of the target position is based on its velocity $\mathbf{V_P}$ where $\mathbf{V_P}$ can be modeled by the linear parametrization form [9] as

$$\mathbf{V_P}(t) = \alpha^T \mathbf{M}(t) , \tag{10}$$

where $\alpha \in \mathcal{R}^{m\times 2} = (\alpha_1 \alpha_2)$ is unknown parameter matrix to be estimated where $\alpha_i \in \mathcal{R}^m$, $i = 1, 2$, and $\mathbf{M} \in \mathcal{R}^m = (\mathbf{V_P}(t-1)^T \ldots \mathbf{V_P}(t-d)^T)^T$ is a signal matrix where $m = 2d$ and d is the *order* of the model. The aim is to calculate estimated values of $\hat{\alpha}_\mathbf{i}$ where the estimated output can be written as

$$\hat{\mathbf{V}}_\mathbf{P}(t) = \hat{\alpha}^T \mathbf{M}(t) . \tag{11}$$

Due to the time-varying nature of the target velocity, *least-squares estimator with exponential forgetting* technique [10] is used in getting $\hat{\mathbf{V}}_\mathbf{P}(T^k)$ and $\hat{\mathbf{X}}_\mathbf{P}(k')$. This algorithm discards old data exponentially under the assumption that the most recent data contain more information. The algorithm is the result of the minimization of the following cost function J

$$J = \sum_{k=1}^{t} \lambda_0^{t-k} \left[\mathbf{V_P}(k) - \hat{\alpha}^T \mathbf{M}(k)\right]^2 \tag{12}$$

where $0 < \lambda_0 \leq 1$ is the *exponential forgetting factor*. Minimizing J with respect to $\hat{\alpha}$, α_i's are estimated on-line by recursive equations as follows

$$\mathbf{P}(t) = \frac{1}{\lambda_0}\left[\mathbf{P}(t-1) - \frac{\mathbf{P}_{(t-1)}\mathbf{M}^T(t)\mathbf{M}(t)\mathbf{P}_{(t-1)}}{\lambda_0 + \mathbf{M}(t)\mathbf{P}_{(t-1)}\mathbf{M}^T(t)}\right] , \tag{13}$$

$$\hat{\alpha}_i^t = \hat{\alpha}_i^{t-1} + \mathbf{P}(t)\mathbf{M}^T(t)\left[V_P(t) - (\hat{\alpha}_i^{t-1})^T \mathbf{M}(t)\right] , \tag{14}$$

where $i = 1, \ldots, n$ and $\mathbf{P}(t)$ is a positive definite symmetric gain matrix. It is assumed that the vector $\hat{\alpha}_i(0)$ and matrix $\mathbf{P}(0)$ are given. $\mathbf{P}(0)$ can be selected as small when accurate knowledge of the vector $\hat{\alpha}(0)$ is known.

The predicted positions of the target for the planning of the camera motion can now be computed from the successive images as [7]

$$\hat{\mathbf{X}}_\mathbf{P}(k') = \mathbf{X_P}(T^{k-1}) + \mathbf{V_P}(T^{k-1})T^{k-1} + \hat{\alpha}^T \mathbf{M}(k) m T_c . \tag{15}$$

5 CONTROL LAWS

For tracking the subgoal trajectory points generated by the planner, PD and self-tuning control laws are used. The former is a simple and fast technique while the latter is expensive but does not need the exact knowledge of the joint dynamics and can compensate for unknown dynamics and payloads without deteriorating its performance.

Taking $\mathbf{u}(t) = (u_\theta(t), u_\phi(t))^T$, the control law for a PD control can be written as

$$\mathbf{u}(t) = \mathbf{K_p}(\mathbf{R^d}(t) - \mathbf{R}(t)) + \mathbf{K_d}(\Omega^\mathbf{d}(t) - \Omega(t)), \tag{16}$$

where $\mathbf{K_p} = diag[K_p^1 K_p^1 K_p^2 K_p^2]$ and $\mathbf{K_d} = diag[K_d^1 K_d^1 K_d^2 K_d^2]$ denote for the proportional and derivative gain matrices. For a minimum overshoot, a critically damped PD controller is proposed and a condition is computed as

$$K_d^i = \frac{2}{K_p^i}\sqrt{T_m^i (K_p^i K_m^i + 1)} , \tag{17}$$

where $i = 1, 2$.

Self-tuning controller is designed using a multivariable exogenous ARX-model for the joint velocity [5]

$$\Omega(k'T_c) = \mathbf{a_0^c} + \mathbf{A^c}(q^{-1}) \quad \Omega(k') + \mathbf{B^c}(q^{-1})\mathbf{u}[(k'-1)T_c] + \mathbf{e^c}(k'), \tag{18}$$

where $k' = T^k/T_c + m^k - 1$, $m^k = 1, \ldots, T_v^k/T_c$. Operator q^{-1} causes a delay of one control sampling period, i.e., $q^{-1}\mathbf{V_P}(k') = \mathbf{V_P}(k'-1)$. The vector $\mathbf{a_0^c} \in \mathcal{R}^2$ accounts for the gravitational effects. Moreover, $\mathbf{A^c}(q^{-1}) \in \mathcal{R}^{2\times 2} = \mathbf{A_1^c} q^{-1} + \ldots + \mathbf{A_{n'}^c} q^{-n'}$ and $\mathbf{B^c}(q^{-1}) \in \mathcal{R}^{2\times 2} = \mathbf{B_0^c} + \mathbf{B_1^c} q^{-1} + \ldots + \mathbf{B_{n'-1}^c} q^{-n'+1}$ where n' specifies the order of the model. The equation error $\mathbf{e^c}(k') \in \mathcal{R}^2$ is a white Gaussian zero mean random variable with known finite variance. For the centroid velocity $\Omega(k') = (\Omega_\theta(k')\ \Omega_\phi(k'))^T$ and position $\mathbf{R}(k') = (\theta(k')\ \phi(k'))^T$, the unknown parameters in $\mathbf{a_0^c}$, $\mathbf{A^c}$ and $\mathbf{B^c}$ are written as:

$$\mathbf{A_1^c} = \begin{bmatrix} a_{11}^c & a_{12}^c \\ a_{21}^c & a_{22}^c \end{bmatrix} , \tag{19}$$

$$\mathbf{B_0^c} = \begin{bmatrix} b_{11}^c & b_{12}^c \\ b_{21}^c & b_{22}^c \end{bmatrix} , \tag{20}$$

$$\mathbf{a_0^c} = [a_{01}^c\ a_{02}^c]^T , \tag{21}$$

Writing the equation (18) in a different notation, the velocity model becomes as $(n' = 1)$

$$\Omega(k') = \alpha^T M(k') + \mathbf{e}(k') , \tag{22}$$

$$\alpha \in \mathcal{R}^{5\times 2} = [\alpha^1\ \alpha^2] , \tag{23}$$

$$\alpha_1 = [a_{11}^c\ a_{12}^c\ b_{11}^c\ b_{12}^c\ a_{01}^c]^T , \tag{24}$$

$$\alpha_2 = [a_{21}^c\ a_{22}^c\ b_{21}^c\ b_{22}^c\ a_{02}^c]^T , \tag{25}$$

$$\mathbf{M}(t) = [\Omega_\theta(t-1)\ \Omega_\phi(t-1)\ u_\theta(t-1)\ u_\phi(t-1)\ 1]^T \tag{26}$$

The recursive equations in (13) and (14) are used for estimating α_i's. The controller is designed on the basis of the velocity model (22) in which the estimates are used for the unknown parameters. The control $\mathbf{u}(k')$ is determined by minimizing

$$I_{k'}[\mathbf{u}] = E\Big[\|\Omega(k'+1) - \Omega^\mathbf{d}(k'+1) + \gamma\big[\mathbf{R}(k') + \Omega(k')T_c - \mathbf{R^d}(k'+1)\big]\|_\mathbf{Q}^2 + \|\mathbf{u}(k')\|_\mathbf{W}^2 \ \ |\Phi(k')\Big], \tag{27}$$

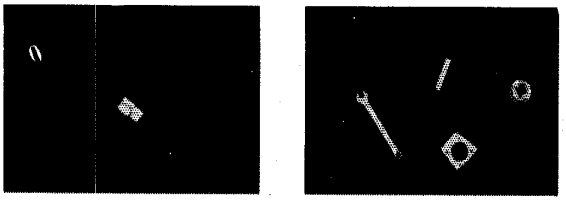

Fig. 5. One and four objects to be tracked.

where $\Omega^{\mathbf{d}}(k'+1)$ and $\mathbf{R}^{\mathbf{d}}(k'+1)$ are the desired velocity and angle vectors of the joint motors at time $(k'+1)T_c$ as specified by the output of the trajectory planner. γ, $\mathbf{Q}$, and $\mathbf{W}$ are symmetrical positive definite weighting matrices and $\|\mathbf{B}\|_{\mathbf{A}}$ denotes for $\mathbf{A}^T\mathbf{B}\mathbf{A}$.

Substituting $\Omega(k'+1)$ by its estimated model (22), expanding $\|.\|$'s and minimizing $I_{k'}[\mathbf{u}]$ with respect to $\mathbf{u}(k')$, $\mathbf{u}(k')$ is computed as

$$\mathbf{u}(k') = [\mathbf{W} + (\hat{\mathbf{B}}^{\mathbf{c}})^T\mathbf{Q}\hat{\mathbf{B}}^{\mathbf{c}}]^{-1}(\hat{\mathbf{B}}^{\mathbf{c}})^T\mathbf{Q}[\Omega^{\mathbf{d}}(k'+1) - \hat{\mathbf{A}}^{\mathbf{c}}\Omega(k') - \hat{\mathbf{a}}_{\mathbf{0}}^{\mathbf{c}} + \gamma(\mathbf{R}^{\mathbf{d}}(k'+1) - \mathbf{R}(k') - \Omega(k')T_c)] \quad (28)$$

6 EXPERIMENTS

For assessing the performance of the complete visual servoing system with respect to real-time applicability, tracking accuracy and speed, two sets of experiments are conducted: (i) experiments based on a simulation of the camera mount tracking system, (ii) experiments performed using a 6-DOF robotic manipulator.

6.1 Simulated Camera Mount Tracking

The task of the camera mount shown in Fig. 2 is chosen as supplying the position and velocity information of a moving target object for a robotic manipulator. For achieving this task, the camera mount tracks a target object, follows it for a duration t_g, and do the same processes for the other objects successively until no object remains in the field of view of the camera.

In the simulations, the gimbal center vector is so that the camera image plane is parallel to the conveyor motion direction initially. The vector (l_1, l_2, l_3) is $(0, 75, 20)$ mm and $f = 22$ mm. In the vision algorithm, T_e, T_d and T_m are chosen as 200, 10 and $T_d/2$ respectively. The initial parameter values in predicting the target centroid velocity are $\mathbf{A_1}(0) = 0.7I$, $\mathbf{A_2}(0) = 0.3I$, and $\mathbf{P}(0) = 100I$, and the forgetting factor is $\mu = 0.99$. For the planner, the maximum velocity and acceleration of the joint motors are taken as 0.5 rad/sec and 0.6 rad/sec^2 where $K_m^1 = K_m^2 = 13.26$, $T_m^1 = T_m^2 = 0.0135$ for the joint motors. T_c for self tuning and PD controllers are taken as 12 $msec$ and 10 $msec$ respectively. For the self tuning controller, $n' = 1$, $\mathbf{A_1^c}(0) = 0.5I$, $\mathbf{B_0^c}(0) = I$, and $\mathbf{a_0^c}(0) = 0$, and $\mu = 0.95$. When the relative distance between the camera optical axis and object position is less than the threshold value 3 mm, the tracking is achieved and $t_g = 2$ sec is chosen.

The first simulation is realized with a metal single object displayed in Fig. 5. For the controllers, $\mathbf{Q} = 0.012I$, $\gamma = 300I$, $\mathbf{K_p} = I$ are selected. The resulting camera optical axis positions with PD and self tuning controllers and the measured object trajectory are shown in Fig. 6. From the figure, it is observed that self tuning control results in a quicker tracking of the trajectory with less tracking error as compared to PD control.

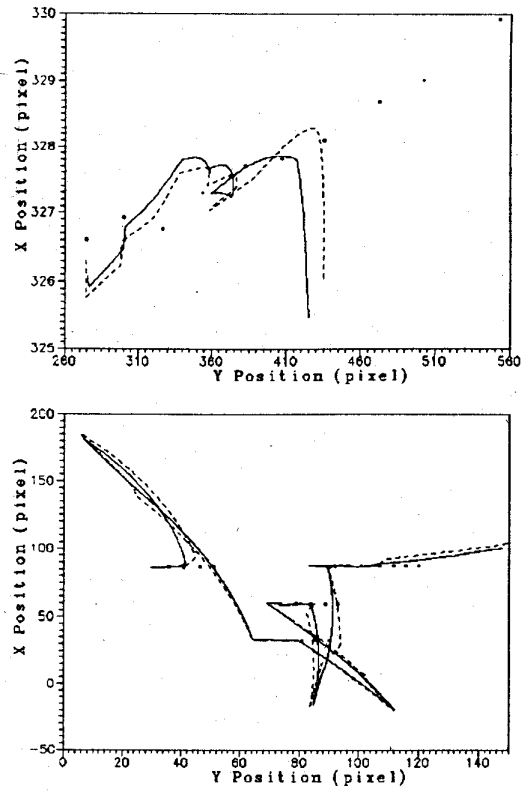

Fig. 6. Trajectories resulting from PD (dotted) and self-tuning (solid) controllers for one and four object cases (black points show the measured target positions).

In the second simulation, four objects as shown in Fig. 5 are used in the simulation. Using the same parameters with the one object case, the resulting trajectories for PD and self-tuning controllers are displayed in Fig. 6. In this case, the effect of the abrupt changes can be observed clearly where larger peaks exist. However, these peaks can be reduced by making T_c small which is a large value for both controllers in the simulations. In spite of the largeness of T_c, both controllers are successful in tracking while self tuning control performs better with respects to generating smooth trajectories and having less tracking error and PD is faster in tracking.

6.2 Robot Grasping Experiments

The applicability of the vision system in real-time is assessed by experiments in which one moving target object is to be grasped by the gripper of a Manutec R15 6-DOF industrial manipulator as shown in Fig. 1. The system hardware configuration is given in Fig. 7. Scorpion frame-grabber card acquires images that are taken from a stationary camera and Eagle board filters them with the Sobel edge mask. The vision algorithm is run on an IBM 486 DX-66 PC using the Borland $C++$ Compiler.

In the experiment, a set of five spongy objects as shown in Fig. 8 are to be picked by the manipulator successively. At any k^{th} edge filtered image of the size 640 x 480 with 256 grey levels which include any combination of the objects is thresholded by $T_e = 100$. The size of the resulting edge map is reduced to 160 x 120. Then, for $T_d = 8$ and $T_m = T_d/2$, the objects are

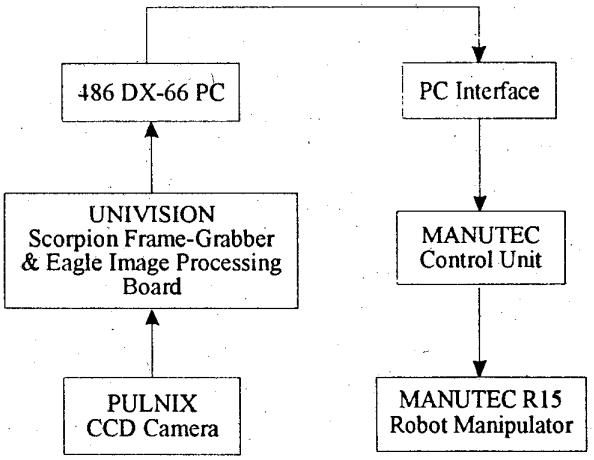

Fig. 7. Hardware used in the experiments.

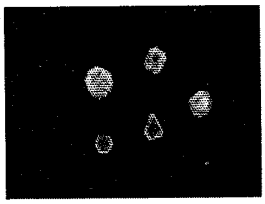

Fig. 8. The spongy objects used in the experiments.

segmented The clusters having less than eight pixels are considered to be the pointwise background noise and eliminated.

Computing $\mathbf{X_P}(T^k)$ target centroid position, $\mathbf{V_P}(T^k)$ is calculated using the equation (1) for the k^{th} frame where $k = 0, \ldots, K$. Assuming the conveyor speed is constant, target constant velocity $\mathbf{V}$ is estimated as

$$\hat{\mathbf{V}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{V_P}(T^k) \,. \tag{29}$$

This estimation reduces the measurement noise in the velocity values with a large K. But since the camera field of view is fixed and limited, K has a limit and for the experiments $K = 4$ is taken For the grasping, the target object centroid position is estimated $t = t_g + t_c$ seconds later as

$$\hat{\mathbf{X}}_\mathbf{P}(T^K + t) = \mathbf{X_P}(T^K) + t\hat{\mathbf{V}}, \tag{30}$$

where for the experiment setup, $t_g = 2.1$ *sec* is the grasping time which is determined empirically and $t_c = 0.205$ sec is the communication time between the PC and robot controller. Then, $\hat{\mathbf{X}}_\mathbf{P}(T^K + t)$ is converted to the robot coordinates and sent to the robot controller. Robot manipulator goes to that point, grasps the object and puts it into a box.

In order to assess the real-time application performance of the vision system, the maximum allowable speeds of the conveyor belt $V^* \in \mathcal{R}$ are found for grasping different number of objects from a set of objects successfully. For the set of objects shown in Fig. 8, V^* values for grasping one, two, three, four and five objects are tabulated in Table 1. In a static camera case, V^* is determined by two main factors: the delay inherent in the vision algorithm and K.

Number of Successive Grasped Objects	*Number of Objects in the Scene*	V^* *(cm/sec)*
1	1	13.84
1	2	13.13
1	3	11.38
1	4	10.23
1	5	8.02
2	2	4.25
2	3	3.89
2	4	3.69
2	5	2.85
3	3	2.37
3	4	2.10
3	5	1.98
4	4	1.43
4	5	1.35
5	5	1.18

Table 1. Maximum allowable velocities.

7 CONCLUSION

A position-based visual servoing structure for the on-line servoing of a pan-tilt camera mount with multiple objects in the camera field of view case is presented. In the camera mount simulations, it is observed that self-tuning controller is less sensitive to the noise and changes in the target motion while PD controller is fast when vision measurements are accurate. The applicability of the introduced vision algorithm is tested by the picking task of a Manutec R17 robotic manipulator which gets its visual information from a stationary camera over a conveyor belt. The experimental results are successful in terms of picking the objects in the determined speed limits for different cases.

REFERENCES

[1] N. P. Papanikolopoulos, P. K. Khosla and Takeo Kanade, "Visual tracking of moving target by a camera mount on a robot: A combination of control and vision," *IEEE Trans. on Robotics and Autom.*, Vol. 9, No. 1, pp. 14-35, February 1993.

[2] R.C. Luo, R.E. Mullen Jr., and D.E. Wessel, "An adaptive robotic tracking system using optical flow," *Proc. IEEE Int. Conf. Robotics Avotomat.*, 1988, pp. 568-573.

[3] B.K.P. Horn and B.G. Schunck, "Determining optical flow," *Artificial Intell.*, vol. 17, pp. 185-204, 1981.

[4] P.K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Real-time visual servoing," *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1850-1856, 1992.

[5] A. J. Koivo and N. Houshangi, "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller," *IEEE Trans. Syst. Man Cybern.*, vol. 21, no. 1, pp. 134-142, 1991.

[6] A.E. Hunt and A.C. Sanderson," Vision based predictive tracking of a moving target," Carnegie Mellon Univ., The Robotic Inst., Tech. Rep. CMU-RI-TR-82-15, Jan. 1982.

[7] M. Sitti, "Visual Tracking: An Integration of Vision and Control," Master's thesis, Dept. Elec. and Electro. Eng., Boğaziçi University, İstanbul, Sept. 1994.

[8] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, Inc., New York, 1973.

[9] Jean-Jacques E. Slotine and Weiping Li, *Applied Nonlinear Control*, Prentice Hall, New Jersey, 1991.

[10] Ljung Lennart, *System Identification: Theory for the User*, Prentice-Hall, New Jersey, 1987.