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

Metin Sitti, İşil 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 Manutech 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

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 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.

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.
2 SETUP

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 \( P \), that we want to track. The projection of \( P \) on the image plane is the point \( p \in \mathbb{R}^2 \). Consider also a neighborhood \( \mathcal{N}_m \) of \( P \) in the conveyor belt plane and that \( \mathcal{N}_t \) of \( p \) in the image plane. The problem of 2-D visual tracking of a single feature point is defined as: Find the camera rotation vector \( \mathbf{R} = (R_1, R_2) \) with respect to the camera frame that keeps \( \mathcal{N}_t \) stationary in the area \( \mathcal{N}_m \) around the origin of the image frame. There is no need to know the depth of the point \( P \) since a 2-D tracking task is proposed. Taking the target feature at the point \( P \) as its centroid point, \( P \) 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 \( \mathcal{N}_t \) with \( \mathcal{N}_m \). From the position of the point \( p \) at time \( T_k \), \( x_P(T_k) \), the velocity \( v_P(T_k) \) is computed using the first-order Taylor Series approximation as

\[
v_P(T_k) = \frac{x_P(T_k) - x_P(T_k-1)}{T_k}.
\]

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

2.1 Perspective Transformation

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

\[
x(T_k) = f[(X(T_k) - X_0) \cos \theta + (Y(T_k) - Y_0) \sin \theta - l_1] \, (2)
\]

\[
y(T_k) = f[(X(T_k) - X_0) \sin \phi \sin \theta + (Y(T_k) - Y_0) \cos \phi \cos \theta + (Z(T_k) - Z_0) \sin \phi - l_2] \, (3)
\]

where \( \Delta = -\left( X(T_k) - X_0 \right) \cos \phi \sin \theta + \left( Y(T_k) - Y_0 \right) \cos \phi \cos \theta + \left( Z(T_k) - Z_0 \right) \sin \phi - l_2 \), \( X_0, Y_0, Z_0 \) are the motion plane coordinates of the camera stand gimbal center \( G \) as shown in Fig. 3, the vector \( (l_1 l_2 l_3) \) 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]

\[
T_1^m \ddot{\theta}(t) + \dot{\theta}(t) = K_m^1 u_\theta, \quad (4)
\]

\[
T_2^m \ddot{\phi}(t) + \dot{\phi}(t) = K_m^2 u_\phi, \quad (5)
\]

where \( K_m^i, T_m^i, i = 1, 2 \), stand for motor gain constant and motor time constant of each DC motor successively and \( u_\theta \) and \( u_\phi \) are input joint torques or voltages.
2.3 Inverse Kinematics of the Camera Stand

As \( \hat{X}\mathbf{P}(k^t) = (\hat{X}(k^t) \hat{Y}(k^t) 0)^T \) refer to the estimated positions in the motion plane coordinates, they are mapped to camera mount joint angles \( \hat{R}(k^t) = (\hat{\theta}(k^t) \hat{\phi}(k^t))^T \) utilizing the inverse kinematics of the camera mount such that

\[
\hat{\theta}(k^t) = \tan^{-1}\left(\frac{\hat{Y}(k^t) - X_0}{\hat{X}(k^t) - X_0}\right),
\]

\[
\hat{\phi}(k^t) = \tan^{-1}\left(\frac{\sqrt{(\hat{X}(k^t) - X_0)^2 + (\hat{Y}(k^t) - Y_0)^2}}{Z_0}\right).
\]

2.4 Trajectory Planner

The task of the trajectory planner is to generate the subgoal/desired points \( \Omega^d[(k^t+1)T_c] \) and \( \mathbf{R}^d[(k^t+1)T_c] \) 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 of 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 \( s_i, i = 1, \ldots, N \) 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 \( k \) \( x^k \) is the center of each cluster as

\[
x^k = \frac{1}{n^k} \sum_{s_i \in \mathbf{X}^k} s_i,
\]

where \( n^k \in \mathbb{R} \) 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 \( T_r \in \mathbb{R} \) - 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 \( s_i \) to the cluster \( \mathbf{X}^k \) where the relation \( |s_i - \mathbf{X}^k| < T_d \) is valid and \( \mathbf{X}^k \in \mathbb{R} \) 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 which \( N \) 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 \( 3 s_i \in X^m, 3 s_j \in X^n \) such that \( |s_i - s_j| < T_m \), then merge \( X^m \) and \( X^n \), where \( T_m \leq T_d \in \mathbb{R} \) is the merging threshold.

Letting \( L \) 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 \( s_i \in X^m \) and \( s_j \in X^n \) 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 \( X \) where \( m = 1, \ldots, \hat{L} - 1, n = m + 1, \ldots, L \).
Step 3. If any merging is done, \( \hat{L} = k, k = 0 \), and go to Step 2; otherwise, compute \( x^k \) 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 \( k^t \) frame is computed and resulting rectangle region is denoted as \( A^k(t) \). Letting the enclosing rectangle region that the target will be at \( t^k + 1 \) seconds later is \( A^k_{t+1} \), it is assumed that

\[
A^k_{t+1} \subseteq A^k \cdot
\]

Then the target object can be searched in the region \( A^k \) 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 \( A^k_{t+1} \) corresponds to another region \( A^k_{t+1} \) in the image plane \( T_{t+1} \) seconds later.

Assuming no object enters to the front of the target object in the region \( A^k_{t+1} \) and letting the enclosing rectangle coordinates as \( (z_{t+1}^1, y_{t+1}^1) \) and \( (z_{t+1}^2, y_{t+1}^2) \) in the \( k^t \) frame, the motion rectangle algorithm at \( k^t \) frame is as follows:

Step 1. \( \Delta x^k = x_{t+1}^1 - x_{t+1}^1 \) is known from \( k - 1^{th} \) image frame.
Step 2. Find the first vertical \( i \) that crosses the target edge pixels in the region \( A^k_{t+1} \) by scanning vertically from the end of \( A^k_{t+1} \) to the beginning; \( z_{t+1}^{i+1} = i \) and \( z_{t+1}^{i+1} \) is approximated as \( z_{t+1}^{i+1} - \Delta z^k \).
Step 3. Apply the correction algorithm [7] for finding the accurate \( z_{t+1}^{i+1}, y_{t+1}^{i+1} \) and \( y_{t+1}^{i+1} \).
Step 4. Compute \( x^k_T(T + 1) \) and \( y^k_T(T + 1) \) from the equations (12) and (1) respectively.
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 $(T^{k+1} + m^2 T_c)$ 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 $V_P$ where $V_P$ can be modeled by the linear parametrization form [9] as

$$ V_P(t) = \alpha^T M(t), \quad (10) $$

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

$$ \hat{V}_P(t) = \hat{\alpha}^T M(t). \quad (11) $$

Due to the time-varying nature of the target velocity, least-squares estimator with exponential forgetting technique [10] is used in getting $\hat{V}_P(T^k)$ and $\hat{X}_P(k')$. 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 $J$

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

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

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

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

where $i = 1, \ldots, n$ and $P(t)$ is a positive definite symmetric gain matrix. It is assumed that the vector $\hat{\alpha}(0)$ and matrix $P(0)$ are given. $P(0)$ 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]

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

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 $u(t) = (u_k(t), u_s(t))^T$, the control law for a PD controller can be written as

$$ u(t) = K_P(R(t) - R(t)) + K_D(R(t) - \Omega(t)), \quad (16) $$

where $K_P = \text{diag}[K_{p1}^1 K_{p2}^1 K_{p3}^2 K_{p4}^2]$ and $K_D = \text{diag}[K_{d1}^1 K_{d2}^1 K_{d3}^2 K_{d4}^2]$ 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

$$ K_D = \frac{2}{K_P^2} \sqrt{T_n^e (K_p^1 K_n^1 + 1)}, \quad (17) $$

where $i = 1, 2$.

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

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

where $k' = T^{k+1}/T_c + m - 1, m = 1, \ldots, T^{k+1}/T_c$. Operator $q^{-1}$ causes a delay of one control sampling period, i.e., $q^{-1} V_P(k') = V_P(k' - 1)$. The vector $a_0^e \in \mathbb{R}^{2n}$ accounts for the gravitational effects. Moreover, $A^e(q^{-1}) \in \mathbb{R}^{2n \times 2} = A_{1}^e q^{-1} + \ldots + A_{n_1}^e q^{-n_1}$ and $B^e(q^{-1}) \in \mathbb{R}^{2n \times 2} = B_{1}^e q^{-1} + \ldots + B_{n_1}^e q^{-n_1}$, where $n_1$ specifies the order of the model. The equation error $e^e(k') \in \mathbb{R}^{2}$ is a white Gaussian zero mean random variable with known finite variance. For the centroid velocity $\Omega(k') = (\Omega_x(k') \, \Omega_y(k'))^T$ and position $R(k') = (\theta(k') \, \phi(k'))^T$, the unknown parameters in $a_0^e$, $A^e$ and $B^e$ are written as:

$$ A_1^e = \begin{bmatrix} a_{11}^e & a_{12}^e \\ a_{21}^e & a_{22}^e \end{bmatrix}, \quad (19) $$

$$ B_0^e = \begin{bmatrix} b_{11}^e & b_{12}^e \\ b_{21}^e & b_{22}^e \end{bmatrix}, \quad (20) $$

$$ a_0^e = \begin{bmatrix} a_{01}^e \ a_{02}^e \end{bmatrix}^T, \quad (21) $$

Writing the equation (18) in a different notation, the velocity model becomes as ($n_1 = 1$)

$$ \Omega(k') = a^T M(k') + e^e(k'), \quad (22) $$

$$ a \in \mathbb{R}^{2 \times 2} = [a_{11} \ a_{21}^2], \quad (23) $$

$$ a_1 = [a_{11}^e \ a_{12}^e \ b_{11}^e \ b_{12}^e \ a_{01}^e]^T, \quad (24) $$

$$ a_2 = [a_{21}^e \ a_{22}^e \ b_{21}^e \ b_{22}^e \ a_{02}^e]^T, \quad (25) $$

$$ M(t) = [\Omega(t-1) \, \Omega(t-1) \, u_\theta(t-1) \, u_\phi(t-1)]^T. \quad (26) $$

The recursive equations in (13) and (14) are used for estimating $a_1$'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 $u(k')$ is determined by minimizing

$$ I_k[u] = E \left[ \| \Omega(k' + 1) - \Omega^d(k' + 1) + \gamma [R(k')]^2 \right], \quad (27) $$

$$ \Omega(k')T_c - R^d(k' + 1) ||^2_Q + ||u(k')||^2_W |\Phi(k')|, \quad (27) $$
where $\Omega^d(k' + 1)$ and $R^d(k' + 1)$ 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. $\gamma$, $Q$, and $W$ are symmetrical positive definite weighting matrices and $\|B\|_A$ denotes for $A^TBA$.

Substituting $\Omega(k' + 1)$ by its estimated model (22), expanding $||\cdot||$'s and minimizing $\epsilon_T(\omega)$ with respect to $u(k')$, $u(k')$ is computed as

$$u(k') = \left[ W + (\dot{\theta}^c)^T Q\dot{\theta}^c \right]^{-1}(\dot{\theta}^c)^T Q[\Omega^d(k' + 1) - \dot{\theta}^c - \gamma(R^d(k' + 1) - R(k') - \Omega(k')T_c)]$$

### 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 $t_f$, 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 $(l_1, l_2, d)$ is $(0, 75, 20)$ mm and $f = 22$ mm.

In the vision algorithm, $T_c$, $T_d$ and $T_m$ are chosen as 200, 10 and $T_d/2$ respectively. The initial parameter values in predicting the target centroid velocity are $A_1(0) = 0.7I$, $A_2(0) = 0.3I$, and $P(0) = 100I$, 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/sec² where $K_m^v = K_m^a = 13.26$, $T_m^v = T_m^a = 0.0135$ 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, $n' = 1$, $A_1^s(0) = 0.5I$, $B_1^s(0) = I$, and $a_1^s(0) = 0$, 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 $t_f = 2$ sec is chosen.

The first simulation is realized with a metal single object displayed in Fig. 5. For the controllers, $Q = 0.012I$, $\gamma = 300I$, $K_p = I$ 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.

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 $640 \times 480$ with 256 grey levels which include any combination of the objects is thresholded by $T_r = 100$. The size of the resulting edge map is reduced to $160 \times 120$. Then, for $T_d = 8$ and $T_m = T_d/2$, the objects are
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

\[
\hat{\mathbf{V}} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{V}_p(T^k).
\]  (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 \( t = t_g + t_r \) seconds later as

\[
\hat{\mathbf{X}}_p(T^k + t) = \mathbf{X}_p(T^k) + t\hat{\mathbf{V}}.
\]  (30)

where for the experiment setup, \( t_g = 2.1 \) sec is the grasping time which is determined empirically and \( t_r = 0.205 \) sec is the communication time between the PC and robot controller. Then, \( \hat{\mathbf{X}}_p(T^k + t) \) 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 \( K \).

<table>
<thead>
<tr>
<th>Number of Successive Grasped Objects</th>
<th>Number of Objects in the Scene</th>
<th>( V^* ) (cm/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>13.84</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>13.13</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>11.38</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>10.23</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>8.62</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4.25</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3.89</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3.69</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2.85</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2.37</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2.10</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1.98</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1.43</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1.35</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1.18</td>
</tr>
</tbody>
</table>

Table 1. Maximum allowable velocities.

7 CONCLUSION

A position-based visual servoing structure for the on-line servicing of a pan-tilt camera mounted 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 Manutek 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