Vision-based control of UR5 robot to track a moving object under occlusion using Adaptive Kalman Filter

K. Ramachandruni*
S. Jaiswal*
ramachandruni.1@iitj.ac.in
jaiswal.1@iitj.ac.in
Mechanical Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, Rajasthan, 342037 India

S. V. Shah
surilshah@iitj.ac.in
Assistant Professor,
Mechanical Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, Rajasthan, 342037 India

ABSTRACT

This paper presents a robust method to track a moving object under occlusion using an off-the-shelf monocular camera and a 6 Degree of Freedom (DOF) articulated arm. The visual servoing problem of tracking a known object using data from a monocular camera can be solved with a simple closed loop controller. However, this system frequently fails in situations where the object cannot be detected and to overcome this problem an estimation based tracking system is required. This work employs an Adaptive Kalman Filter (AKF) to improve the visual feedback of the camera. The role of the AKF is to estimate the position of the object when it is occluded/out of view and remove the noise and uncertainties associated with visual data. Two estimation models for the AKF are selected for comparison and among them, the Mean-Adaptive acceleration model is implemented on a 6-DOF UR5 articulated arm with a monocular camera mounted in eye-in-hand configuration to follow the known object in 2D cartesian space (without using depth information).

CCS CONCEPTS

• **Computer systems organization → Robotic control**.

KEYWORDS

Adaptive Kalman Filter, Visual Servoing, Target following, Robotic arm

1 INTRODUCTION

Moving object tracking is an important problem as it is required in many systems such as mass production and fully automated factories, where pick-and-place and assembly operations of various components are required on the assembly line and sports and wildlife filming, where fast moving targets need to be followed and recorded accurately. The tracking problem can be solved using data from a variety of sensors like thermal/IR cameras, stereo cameras (depth or disparity maps), or monocular cameras (raw pixel data). The advantage of using raw pixel data is that it can be obtained from a simple monocular camera and standard image processing methods can be used to easily detect the object and measure its state. However, in the visual servoing problem, where visual feedback is used to control a robot, this raw data cannot be used directly, especially when the position error of the object to be tracked is large. This might lead to severe oscillations and jerky motions of the follower robot which is undesirable. Hence, a closed loop controller is required for fast and accurate visual servoing.

A problem which arises with general closed-loop controllers in noisy and occluded environments is the frequent loss of detection which causes abrupt stopping of the robot. To improve the performance of these controllers in such situations, a Kalman Filter is generally used to estimate and correct the state data. The use of a Kalman filter along with a suitable motion model allows the system to provide reliable state data even in situations of occlusion/no visibility of the object. Among the known visual servoing techniques(Position based Visual Servoing (PBVS) and Image based Visual Servoing (IBVS)), the IBVS method is simpler as it defines the current and desired positions of the target in pixel coordinates (u, v) and does not need any information about the pose of the camera. In addition to a naive implementation of a closed-loop controller for object tracking such as a PD/PID controller, many additional methods have been applied to make the system more robust. Of these methods, Kalman filter is a common choice for object tracking problems in many contexts. [1] used an Extended Kalman filter (Kalman filter for non-linear models) with unconstrained brownian motion model for object tracking from visual data. Other papers such as [2] and [3] use Adaptive Kalman filters (AKFs), where the estimation parameters are adjusted according to the current motion. [2] uses a constant velocity model to model the AKF while [3] uses a zero mean acceleration model known as Singer model. Other mathematical models were presented in [4] to use for tracking moving targets.

The problem tackled in this paper is to follow a moving object (object known to us) in a noisy and cluttered environment using a 6 Degree of Freedom (DOF) articulated arm with an attached visual sensor. The sensor is an off-the-shelf USB monocular camera attached in an eye-in-hand configuration (attached to the end-effector of the arm) and the object is freely moving in 2D cartesian space (no depth information measured). It is assumed that image coordinates of the object can be easily extracted and hence the current solution does not try to tackle the issue of feature extraction. The

*Both authors contributed equally to this research.

AIR '19, July 2–6, 2019, Chennai, India

ACM ISBN 978-1-4503-6650-2/19/07…$15.00
https://doi.org/10.1145/3352593.3352675

visual servoing system explained in this paper uses a PD controller coupled with an Adaptive Kalman Filter (AKF).
The first half of this paper describes the controller being used in visual servoing for tracking an object and following it using the UR5 robotic arm. The second half explains the development of a robust state estimator for aiding the visual servoing problem using an Adaptive Kalman Filter (AKF). Two motion models which are selected for the AKF are compared and the more suitable one is implemented on the UR5 arm. Implementation results are presented and discussed in the latter sections.

2 FEATURE EXTRACTION AND GEOMETRY

Vision-based control methods try to minimize the error e_s between the current image feature of an object s_t and its desired feature value $s*$. Here, Image based Visual Servoing (IBVS) is used along with an eye-in-hand monocular camera system to do the same. Image features are measured in the image plane using color thresholding and blob detection and tracking is done using an appropriate control law. The object considered for tracking is a 2-D circular object of a known color which is assumed to be moving in a fixed plane parallel to the camera frame. This section provides some background on the image processing and feature transformation techniques used to do the same.

2.1 Feature extraction from Image

For this problem, a green circular cap is being used as the known target. Feature extraction is implemented using the **OpenCV** library [5] of Python to perform blob detection. Color thresholding is done based on the HSV (Hue, Saturation, Value) range of object in order to form a blob and further thresholds are applied based on area, circularity and convexity to eliminate background noise. The center of the blob (u, v) and area are obtained in pixel coordinates, which are the final image features.

2.2 Camera Model and Interaction Matrix

Using the pin-hole camera model, an interaction matrix $L(s, a)$ can be defined that relates the camera velocity ξ to the image feature velocity vector $\dot{s}$ as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{z} & -\frac{\lambda^2+u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2+u^2}{\lambda} & -\frac{uv}{z} & -u \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (1)$$

This can also be written as:

$$\dot{s} = L(s, q)\xi, \quad where \dot{s} = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix}, \xi = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad (2)$$

$$L(s, q) = \begin{bmatrix} -\frac{\lambda}{z} & 0 & \frac{u}{z} & \frac{uv}{z} & -\frac{\lambda^2+u^2}{\lambda} & v \\ 0 & -\frac{\lambda}{z} & \frac{v}{z} & \frac{\lambda^2+u^2}{\lambda} & -\frac{uv}{z} & -u \end{bmatrix}$$

As depth information (z cartesian coordinate) cannot be directly measured using a monocular camera, the target object (fixed size) was used to obtain a correlation between the pixel area of blob and the object distance from camera in Cartesian space. This approximation is, however, only valid in the case of circular objects with known dimensions. The data obtained for correlation is shown below:

Z distance (cm)	10	20	30	40	50	60	70
Blob Area (Pixels)	495	253	164	123	97	85	72

The relation obtained between object pixel area and z distance from camera plane upon curve fitting is:

$$Z(m) = \frac{55.4}{Area_{pixel}} \quad (3)$$

The assumption that the motion of the ball will remain in a plane parallel to the camera frame ensures that the value of Z will remain fixed for a given target trajectory.

3 CONTROL LAW FOR IBVS

After object detection and feature measurement, a closed-loop controller must be applied on the image features obtained.If $\begin{bmatrix} u_t \\ v_t \end{bmatrix}$ denotes the feature vector of an image (coordinates of center of object in image space) at time t, then we can define the error functions as $e_u(t) = |u_t - u_0|$ and $e_v(t) = |v_t - v_0|$, where u_0 and v_0 are the desired feature values of u and v respectively.
PID is a popular closed-loop controller used in many applications. It applies a correction to the proportional, integral, and derivative error terms (P, I, and D respectively). The control equation is:

$$CV(\text{control output}) = -(K_p \cdot e(t) + K_d \cdot \dot{e}(x) + K_i \cdot \int e(t)dt) \quad (4)$$

where K_p, K_d and K_i are the proportional, derivative and integral constants respectively.
As explained in majority of the visual servoing literature such as [6], PD controllers have a less settling time than PID contollers and hence for tracking a continuously moving object, it is preferred to use a PD controller i.e, remove the integral term completely. The final PD control law can be written as ([7], [8]):

$$\dot{u}(t) = -[K_{pu} \cdot e_u(t) + K_{du} \cdot \frac{de_u(t)}{dt}] \quad (5)$$

$$\dot{v}(t) = -[K_{pv} \cdot e_v(t) + K_{dv} \cdot \frac{de_v(t)}{dt}] \quad (6)$$

From Eq. 1, by considering the 2D motion of camera i.e, only V_x and V_y are mapped to image feature velocity, the following relation is obtained:

$$V_x = \frac{-z}{\lambda} \cdot \dot{u}; V_y = \frac{-z}{\lambda} \cdot \dot{v} \quad (7)$$

where λ is the focal length of camera and z is the Z estimate obtained from the pixel area of blob, using Eq. 3. From equations 1-2 and 5-7, and by using the kinematic jacobian (i.e, relation between joint velocities and end-effector velocity of robot), joint velocities can be calculated for visual servoing ($\dot{q} = J_v^{-1} \cdot \dot{V}$).

PD Tuning. The PD gains K_p and K_d decide the dependence of control output on the error function. After studying literature on how PID/PD controllers are tuned, it was found that many prefer to use a trial and error method to manipulate the gain values and observe the control response of each trial to check for desired behaviour (e.g. settling time, accurate convergence, overshoot, etc.). Hence a similar approach was followed.

3.1 Shortcomings of using current model

After tuning the PD controller, the UR5 robot was able to perform fast and smooth tracking. However, due to external factors such as bad lighting condition or obstruction of another object, the blob detection tends to fail frequently (Fig 3). In addition, the low FPS (Frames Per Second) of the camera creates a unique problem: once the detection is missed for a few frames, the object sometimes moves out of the visible region of the camera if it is moving very fast. This problem is being called as the **out of view** problem (Fig 2). This leads to the abrupt stopping of the robot and complete failure of tracking. In order to circumvent these (occlusion and out of view) problems, an Adaptive Kalman Filter (AKF) was coupled with the existing controller, which is a state estimator used to provide reliable data in noisy environments. Implementation details are explained in the next section.

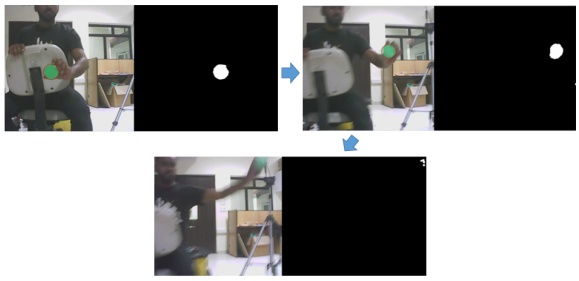


Figure 2: Sequence of frames showing the 'Out of view' problem. The right side (black & white) shows the detected blob while the left side (RGB) shows the raw image in which object is detected with a red circle. Although ball is still detected in the first two images, image blur (due to fast motion) and sudden escape from the camera view leads to detection failure.

Figure 3: Image frames showing the object being blocked by the chair, similar to Fig.2. In the first image object is completely detected and hence no occlusion, while in the second image object is partially visible so occlusion occurs and in the last image no sight of the object so it is completely occluded.

4 ADAPTIVE KALMAN FILTER

In order to create a robust tracking model that can handle the noise and uncertainties of visual measurements, an Adaptive Kalman Filter (AKF) was implemented on the image coordinate data u, v. The following section discusses the basic equations related to a basic Kalman Filter and compares the performance of two estimation models selected from the existing literature.

4.1 Kalman filter Equations

The Kalman filter, as explained in [9], is a set of mathematical equations that recursively implements a predictor-corrector estimator by minimizing the estimated error covariance in order to provide new values for the state variables.

Let $X(k)$ be the state of a discrete time-controlled process at time t_k, governed by the equations:

$$X(k) = F(k-1) \cdot X(k-1) + W(k-1) \tag{8}$$

and $Z(k)$ be the measurement of state, then:

$$Z(k) = H(k) \cdot x(k) + V(k) \tag{9}$$

where $F(k-1)$ is the state transition matrix, $H(k)$ is measurement matrix and $W(k)$ and $V(k)$ are the process and measurement noises respectively. The noises are usually modelled as independent white Gaussian noises with covariances $Q(k)$ and $R(k)$ respectively. Q, R are the process noise covariance and measurement noise covariance respectively. It is assumed that they remain constant for the given time-step.

Let $X(k/k-1)$ be the *a priori* estimate i.e, prediction of state $X(k)$ using the measurements $X(0), X(1)$ ..... $X(k-1)$. The *a priori* and *a posteriori* estimate errors are then defined respectively as:

$$E^{-}(k) = X(k) - X(k/k-1) \tag{10}$$

$$E(k) = X(k) - X(k/k) \tag{11}$$

where $X(k/k)$ is the *a posteriori* estimate of state $X(k)$. The *a priori* error covariance and *a posteriori* error covariance are defined using the above error terms respectively.

The kalman filter consists of two phases in order to update the *a priori* and *a posteriori* terms (state and covariance): Prediction and Correction (refer Figure 1):

Prediction. In the prediction step an *a priori* estimate is obtained by forwarding the current state as shown:

$$X(k/k-1) = F(k-1) \cdot X(k-1/k-1) \tag{12}$$

A priori error covariance can then be written as:

$$P(k/k-1) = F(k-1)P(k-1/k-1)F(k-1)^T + Q(k-1) \tag{13}$$

Correction. The correction step acts as a feedback mechanism by incorporating the actual measurement into the *a priori* estimate $X(k/k-1)$ to obtain an *a posteriori* estimate $X(k/k)$:

$$X(k/k) = X(k/k-1) + K(k) \cdot (Z(k) - H(k) \cdot X(k/k-1)) \tag{14}$$

where $K(k)$ is known as the Kalman gain given by:

$$K(k) = \frac{P(k/k-1) \cdot H(k)^T}{H(k) \cdot P(k/k-1) \cdot H(k)^T + R(k)} \tag{15}$$

A posteriori error covariance becomes:

$$P(k/k) = (1 - K(k) \cdot H(k)) \cdot P(k/k-1) \tag{16}$$

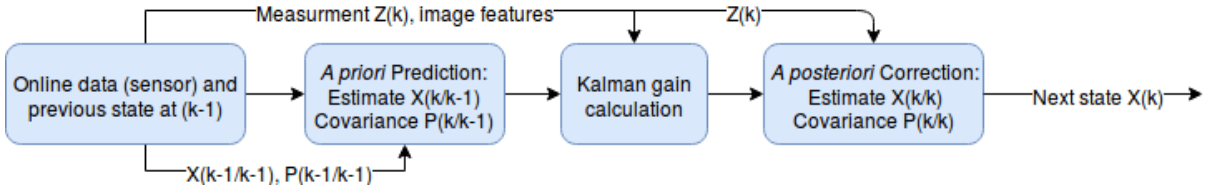


Figure 1: Flowchart showing various Prediction-Correction steps of state estimation for visual data executed by an Adaptive Kalman Filter.

These two steps are repeated for each time step.

Adaptive kalman filter. Adaptive Kalman Filter or AKF allows estimation parameters of the Kalman filter to adjust automatically, as shown in Fig. 1, where Z(k) and image features are used to update the KF states.

4.2 Estimation models

In order to implement the adaptive Kalman filter for visual servoing two models have been selected for comparison, after which one of the models was implemented on the UR5 robot.

4.2.1 ***Model 1- Uniform Velocity model.*** This system was implemented in [2] for tracking moving objects in videos. The system state of model is defined as:

$$\text{State}X(k) = \begin{bmatrix} d(k) \\ d(k-1) \end{bmatrix} \tag{17}$$

where $d(k)$ is image feature at time t_k, in this case the image coordinates u, v.
For a very short time interval T, velocity of the moving object is considered to be constant. The estimated feature can then be written as:

$$d(k) = d(k-1) + (d(k-1) - d(k-2)) \tag{18}$$

State model is then represented by:

$$\begin{aligned} X(k) &= F(k-1)x(k-1) + W(k-1) \\ &= \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} d(k-1) \\ d(k-2) \end{bmatrix} + \begin{bmatrix} W(k-1) \\ 0 \end{bmatrix} \end{aligned} \tag{19}$$

where F(k-1) =$\left[\begin{smallmatrix} 2 & -1 \\ 1 & 0 \end{smallmatrix}\right]$.

Z(k) is taken as [d(k)] and H(k)= [1 0], giving:

$$Z(k) = H(k) \cdot X(k) + V(k) = d(k) + V(k) \tag{20}$$

Adjustment of process noise covariance and measurement noise covariance using image features is explained below.

Occlusion Rate. As explained in [2], the occlusion rate is the ratio of occlusion area (in pixels) in frame t to that in frame $t-1$.

$$\alpha(t) = \begin{cases} |\frac{PN(t)}{PN(t-1)} - 1| & \text{if } |\frac{PN(t)}{PN(t-1)} - 1| \leq 1 \\ 1 & \text{if } |\frac{PN(t)}{PN(t-1)} - 1| \geq 1 \end{cases} \tag{21}$$

where $PN(t)$ and $PN(t-1)$ are the pixel number of the moving object in frame t and $t-1$, respectively. Referring to Figure 3, in the first frame as the object is clearly visible, there is no occlusion and hence $\alpha_t = 0$. In the second frame, as occlusion is observed, $\alpha_t < 1$ and in the last frame, as object cannot be detected at all, there is complete occlusion and $\alpha_t = 1$.
The occlusion rate can be used as a measure of noise covariance as described in [2], where the measurement error is considered directly proportional to the occlusion rate (more occlusion leads to less accurate measurement). Hence the following estimation parameters are defined:

$$R(t) = \begin{cases} \alpha(t) & \text{if } \alpha(t) < \alpha^*(t) \\ \infty & \text{else} \end{cases} \tag{22}$$

$$Q(t) = \begin{cases} 1 - \alpha(t) & \text{if } \alpha(t) < \alpha^*(t) \\ 0 & \text{else} \end{cases} \tag{23}$$

where $\alpha^*(t)$ is the threshold value of $\alpha(t)$.

In the case when $\alpha(t)$ exceeds the threshold (i.e. when object is completely invisible) $K(t)$ will become 0 (refer Eq. 15) and hence the system will use the predicted value obtained from equation 19. The final estimation equations can now be written as:

Prediction

1. $X(k/k-1) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} d(k-1) \\ d(k-2) \end{bmatrix}$

2. $P(k/k-1) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} \cdot P(k-1/k-1) \cdot \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}^T + \begin{bmatrix} 1-\alpha(t) & 0 \\ 0 & 1-\alpha(t) \end{bmatrix}$

Correction

1. $K(k) = \frac{P(k/k-1) \cdot [1\ 0]^T}{[1\ 0] \cdot P(k/k-1) \cdot \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right] + \alpha(k)}$

2. $X(k/k) = X(k/k-1) + K(k) \cdot (d(k) - d(k/k-1))$

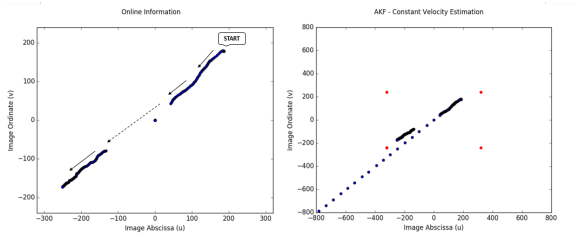
3. $P(k/k) = (1 - K(k) * \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]) \cdot P(k/k-1)$

where d is any coordinate of the image plane. By using AKF to estimate both directions of motion we can predict its motion in image plane.

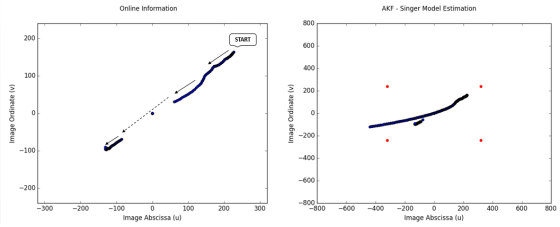
4.2.2 ***Model 2- Mean adaptive acceleration model.*** This model was proposed in [4] as one of a number of dynamic models suitable for tracking maneuvering targets. This acceleration model is a Singer model with an adaptive mean. While the singer model (used in [3]) considers acceleration to be a zero mean first order Markov process i.e, $E[a(t+\tau)a(t)] = \sigma^2 * e^{-\alpha * |\tau|}$, the mean adaptive acceleration model modifies the singer model to have a non-zero mean of the acceleration satisfying the equation:

$$\dot{a}(t) = -\alpha \cdot \widetilde{a}(t) + W(t) \text{ or } \dot{a}(t) = -\alpha \cdot a(t) + \alpha \cdot \bar{a}(t) + W(t) \tag{24}$$

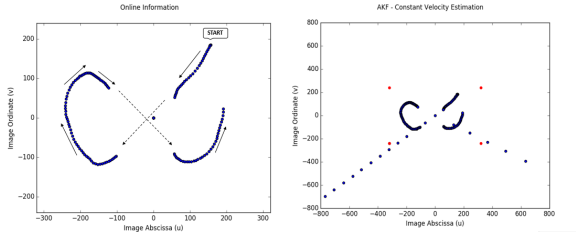
where $\bar{a}(t)$ is the estimate of $a(t)$ from all available online information until time t.



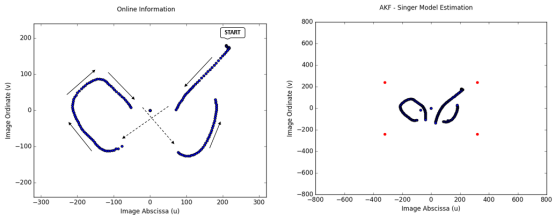
(a) Uniform Velocity Model



(b) Mean Adaptive Acceleration Model



(c) Uniform Velocity Model



(d) Mean Adaptive Acceleration Model

Figure 4: Comparing linear(4a, 4b) and curvi-linear(4c, 4d) estimation performance between both the models under occlusded motion. X and Y axis denote the image plane axes. The left image is motion captured by the camera (image dimension 640x480). Right image is the data output of Kalman filter, where the dimensions of the camera frame are shown using the four red dots in the image (Kalman filter sometimes overshoots estimation data). Start position of the motion is labelled and visible motion of the object is denoted by arrow lines. Occluded motions (object blocked from view) is shown using dotted arrow lines. Refer Discussion section for more explanation.

The discrete-time system state model is given by:

$$X(k/k-1) = F_a \cdot X(k-1/k-1) + U_a \cdot \bar{a}_{k-1} \tag{25}$$

where $X(k) = [d(k), \dot{d}(k), \ddot{d}(k)]^T$ is the state of system at time t_k, $d(k)$ being the image coordinates (u, v) and U_a is a matrix defined to apply the effect of non-zero mean acceleration. From Eq. 24:

$F_a = \begin{bmatrix} 1 & T & \frac{\alpha \cdot T - 1 + \beta}{\alpha^2} \\ 0 & 1 & \frac{1-\beta}{\alpha} \\ 0 & 0 & \beta \end{bmatrix}$, $U_a = \left[\begin{bmatrix} \frac{T^2}{2} \\ T \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{\alpha \cdot T - 1 + \beta}{\alpha^2} \\ \frac{1-beta}{\alpha} \\ \beta \end{bmatrix} \right]$ and $\beta = e^{-\alpha \cdot T}$.

Based on this model (similar to formulation in [3]), the process noise covariance matrix is given by:

$$Q(k) = E[w(k)w(k)^T] = 2 \cdot \alpha \cdot \sigma_a^2 \cdot \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \tag{26}$$

where:

$q_{11} = \frac{1}{2 \cdot \alpha^5} \cdot (1 - \beta^2 + 2 \cdot \alpha \cdot T + \frac{2 \cdot \alpha^3 \cdot T^3}{3} - 2 \cdot \alpha^2 \cdot T^2 - 4 \cdot \alpha \cdot T \cdot \beta)$

$q_{12} = \frac{1}{2 \cdot \alpha^4} \cdot (\beta^2 + 1 + 2 \cdot \beta + 2 \cdot T \cdot \beta - 2 \cdot \alpha \cdot T - \alpha^2 \cdot T^2)$

$q_{13} = \frac{1}{2 \cdot \alpha^3} \cdot (1 - \beta^2 - 2 \cdot \alpha \cdot T \cdot \beta)$

$q_{22} = \frac{1}{2 \cdot \alpha^3} \cdot (4 \cdot \beta - 3 - \beta^2 + 2 \cdot \alpha \cdot T)$

$q_{23} = \frac{1}{2 \cdot \alpha^2} \cdot (\beta^2 + 1 - 2 \cdot \beta)$

$q_{33} = \frac{1}{2 \cdot \alpha} \cdot (1 - \beta^2)$.

[3] also proposed an update rule for the acceleration variance σ_a^2:

$$\sigma_a^2 = 2 \cdot \frac{X(k/k) - X(k/k-1)}{T^2} \tag{27}$$

By retaining the use of measurement noise error as occlusion ratio, we have $R(k) = \alpha(k)$.

Mean Acceleration Estimation. As a modification to the mean acceleration update rule proposed in [4], in the current implementation an exponential moving average (EMA, [10]) was taken instead:

$$\bar{a}_{k+1} = \eta \cdot (\hat{a}_k) + (1 - \eta) \cdot \bar{a}_k \tag{28}$$

where $\bar{a}_k$ is mean of acceleration until time t_k and $\hat{a}_k$ is the online information about current acceleration at time t_k.

Though this update rule is similar to the acceleration update suggested in [4] (i.e, $\bar{a}_{k+1} = \beta \cdot \hat{a}_k + (1-\beta) \cdot \bar{a}_k$, $\beta = e^{-\alpha \cdot T}$), in the EMA mean acceleration is not a function of the number of data points (i.e, not a function of time) measured unto t_k. For an application such as continuously following a moving target, many data points will be received and hence mean acceleration should not be a function of time (recent values and old values must be given equal weightage during update). Being a constant, the value of η has been manually adjusted to give desirable results.

5 RESULTS & DISCUSSION

The figures 4a, 4b and 4c, 4d show a comparison between the estimation provided by Uniform (constant) Velocity model and Mean-Adaptive Acceleration model. Two different motions have been generated with an occluded region in between the motion (object is undetectable in this region) and both prediction models were applied to compare their accuracy in predicting the object's next position.

As shown in the results (refer figures 4a, 4b and 4c, 4d), the main

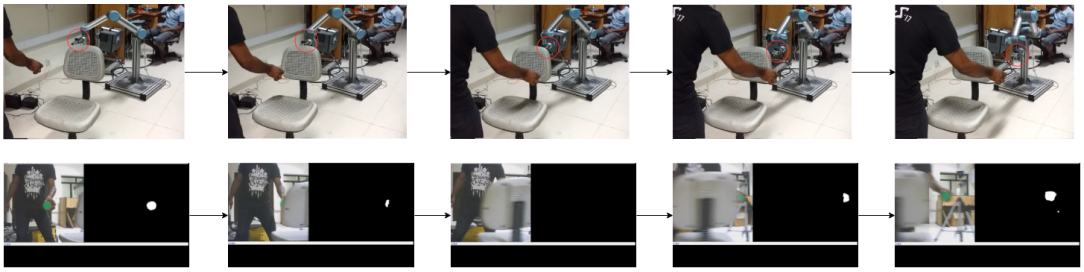


Figure 5: Series of frames showing performance of AKF with Mean Adaptive Acceleration model during occlusion. The top row images show the green cap (target) being tracked by the robot end effector(red circle). The bottom row images shows the camera feed from eye-in-hand camera mounted on UR5 arm (left images) and corresponding blob being detected from that image (right images).

issue with the Uniform Velocity model is that the estimated position points are highly overshot with a sharp linear trajectory, while those of the Mean Adaptive Acceleration model are more dense and a smooth trajectory is generated. Also, the estimation data given by the Mean Adaptive Acceleration model during occlusion lies within the camera's field of view unlike that of the Uniform Velocity model (much lower data overshooting as shown in 4, the four red dots in right image represent the view of the camera). Though the exact curvature of the trajectory is not perfect, the authors argue that for a randomly generated motion which does not follow any particular trajectory, it is impossible to predict that motion even for a human observer.

Using the Mean-Adaptive Acceleration model, visual servoing was carried out on the UR5 robotic arm. It was observed that the problems of occlusion and out of view were reduced in most cases, as shown in Fig. 5. In this series of frames, the target object (green cap) is being blocked from the camera's view by the chair (in the bottom row images blob is not detected during occlusion), however the robot is still able to follow the object due to data obtained from the estimation model. Also, it is worth noting that significant overshooting was not observed while estimating the ball trajectory during occlusion.

By studying the various applications for which similar models have been used([2] and [3]), it can be suggested that the Uniform Velocity model works better for applications where the acceleration of motion is very low i.e, non-erratic and unidirectional motions. Also, the advantage of Uniform Velocity model is that none of the parameters of the AKF require manual tuning. However, for accelerated motions with randomly changing directions, it is better to use the Mean Adaptive Acceleration model model and carefully tune its constants (η and β) to achieve required accuracy.

6 CONCLUSION

An Adaptive Kalman Filter was used along with a closed loop PD controller to perform a visual servoing task on a UR5 robotic arm and its performance was studied. Mathematical models were formulated for the estimation model used known as Mean-Adaptive Acceleration model and it was compared with another existing model (Uniform Velocity model) used for similar tasks.

In order to thoroughly improve the accuracy of the system, the frequency of visual measurements needs to be improved. Frequencies obtained using a high-speed camera (around 400 FPS) are generally required for tracking and hitting a randomly thrown object (such as a robot playing Table Tennis). With some modifications in formulations the same AKF formulation can be improved to provide information at a higher frequency than the incoming visual feedback. Also, the solution proposed in this paper can be implemented using a high-speed camera to see whether there is any improvement in performance as more image data will allow the AKF to make more accurate predictions, especially during occlusion and other cases of failed detection. Further, it is worth noting that as the Kalman filter directly receives extracted coordinate features from the image data, the solution proposed can be extended to more complex 3-D objects as well provided the camera is able to capture this information such as a stereo camera and there is a feature extraction module that can extract raw image coordinates from the data collected.

REFERENCES

[1] E. V. Cuevas, D. Zaldivar, and R. Rojas, "Kalman filter for vision tracking," 2005.

[2] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, "Video object tracking using adaptive kalman filter," *Journal of Visual Communication and Image Representation*, vol. 17, no. 6, pp. 1190–1208, 2006.

[3] C. Liu, X. Huang, and M. Wang, "Target tracking for visual servoing systems based on an adaptive kalman filter," *International Journal of Advanced Robotic Systems*, vol. 9, no. 4, p. 149, 2012.

[4] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on aerospace and electronic systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[5] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[6] M. Elena, M. Cristiano, F. Damiano, and M. Bonfé, "Variable structure pid controller for cooperative eye-in-hand/eye-to-hand visual servoing," in *Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003.*, vol. 2, pp. 989–994, IEEE, 2003.

[7] Z. Qu, R.-q. Wen, X.-y. Wang, and B.-b. Zhou, "Image recognition and tracking algorithm based on pid fuzzy control," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 2A, pp. 403–412, 2016.

[8] I. Siradjuddin, L. Behera, T. M. McGinnity, and S. Coleman, "Image-based visual servoing of a 7-dof robot manipulator using an adaptive distributed fuzzy pd controller," *IEEE/ASME Transactions On Mechatronics*, vol. 19, no. 2, pp. 512–523, 2014.

[9] G. Welch and G. Bishop, "An introduction to the kalman filter. department of computer science, university of north carolina," *ed: Chapel Hill, NC, unpublished manuscript*, 2006.

[10] A. Raudys, V. Lenčiauskas, and E. Malčius, "Moving averages for financial data smoothing," in *International Conference on Information and Software Technologies*, pp. 34–45, Springer, 2013.