A Novel Vision-Based Tracking Algorithm for a Human-Following Mobile Robot

Meenakshi Gupta, Swagat Kumar, *Member, IEEE*, Laxmidhar Behera, *Senior Member, IEEE*, and Venkatesh K. Subramanian

***Abstract*—The ability to follow a human is an important requirement for a service robot designed to work along side humans in homes or in work places. This paper describes the development and implementation of a novel robust visual controller for the human-following robot. This visual controller consists of two parts: 1) a robust algorithm that tracks a human visible in its camera view and 2) a servo controller that generates necessary motion commands so that the robot can follow the target human. The tracking algorithm uses point-based features, like speeded up robust feature, to detect human under challenging conditions, such as, variation in illumination, pose change, full or partial occlusion, and abrupt camera motion. The novel contributions in the tracking algorithm include the following: 1) a dynamic object model that evolves over time to deal with short-term changes, while maintaining stability over long run; 2) an online K-D tree-based classifier along with a Kalman filter is used to differentiate a case of pose change from a case of partial or full occlusion; and 3) a method is proposed to detect pose change due to *out-of-plane rotations*, which is a difficult problem that leads to frequent tracking failures in a human following robot. An improved version of a visual servo controller is proposed that uses feedback linearization to overcome the chattering phenomenon present in sliding mode-based controllers used previously. The efficacy of the proposed approach is demonstrated through various simulations and real-life experiments with an actual mobile robot platform.**

***Index Terms*—Feedback linearization, K-D tree, Kalman filter, out-of-plane rotations, speeded up robust feature (SURF)-based human tracking, visual servo controller.**

I. Introduction

THE current decade has seen robots moving out of their confinement in factory assembly lines to our houses and offices. These robots, termed as *service robots*, can operate in an unstructured environment while sharing their workspaces with humans. These service robots are required to have capabilities that would enable them to effectively interact with humans and, the ability to follow them is one such important attribute for such robots. For instance, a personal assistant robot [1] may carry baggage and follow an elderly person in an airport. Similarly, a robot in a retail store [2] may carry the merchandise of a customer and follow him/her to a checkout counter. This paper looks into the challenges of building a reliable and robust visual controller for such robots. The visual controller for a human-following robot comprises of two parts: 1) a robust vision-based algorithm that tracks a human visible in its camera view and 2) a servo controller that generates necessary motion commands so that the robot can follow the human while maintaining a constant distance from the target.

The human walks in front of the mobile robot with natural and unconstrained gait in an upright pose. The visual tracking of the target human offers several challenges due to effects such as, camera motion, variation in illumination, scaling, in-plane pose variation and pose change due to *out-of-plane* rotation, partial or full occlusion, etc. On the other hand, visual servoing has to deal with challenges like camera calibration errors [3], [4], field of view (FOV) constraints [5] and localization errors due to inaccurate robot odometry [6] and nonholonomic constraints. A number of methods have been proposed in the literature to overcome these challenges, an overview of which is provided in the next section.

This paper is also an attempt to address some of the above-mentioned challenges and provide a robust visual servoing algorithm for a human-following robot. The tracking algorithms makes use of the tracking-by-detection framework [7] which is based on locally invariant point features like speeded up robust features (SURFs) [8] to recognize target in each frame. Point-based features like SURF are known to be robust to photometric and geometric distortions and are less sensitive to abrupt object motion or low frame rate video [9]. The main problem of interest-point-based methods lies in the fact that the availability of matching points may vary significantly from one frame to another leading to tracking failure over a long run. This problem becomes more difficult for a nonrigid target like human that may undergo shape or pose change during its motion. So, a tracking method based on a fixed target template may not work for a long duration. In order to account for the temporal changes that might accrue into the target appearance, the object model is updated over time by selecting new templates and projecting stable point features from previous templates onto the current template through affine transformation (AT).

Manuscript received April 19, 2016; accepted September 20, 2016. Date of publication October 24, 2016; date of current version June 22, 2017. This paper was recommended by Associate Editor S. Nahavandi. *(Corresponding author: Swagat Kumar.)*

M. Gupta, L. Behera, and V. K. Subramanian are with the Department of Electrical Engineering, Indian Institute of Technology at Kanpur, Kanpur 208016, India (e-mail: meenug@iitk.ac.in; lbehera@iitk.ac.in; venkats@iitk.ac.in).

S. Kumar is with TCS Research, Tata Consultancy Services, New Delhi 201309, India (e-mail: swagat.kumar@tcs.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2016.2616343

A method is also proposed to detect pose change due to *out-of-plane* rotations which is a very difficult problem to solve with interest point-based tracking methods. A K-D tree-based classifier combined with a Kalman filter is used to differentiate between a case of occlusion and a pose change due to out-of-plane rotations whenever a tracker failure is detected. The output of the tracking algorithm is fed to the servoing module that generates necessary motion commands to keep the target within the robot camera's FOV. The visual servoing algorithm follows the same formulation as given in [4], but uses feedback linearization technique to overcome the chattering effect arising out of sliding mode control approach. The efficacy of the approach is demonstrated through several simulations as well as real experiment on an actual mobile robot platform.

In short, the contributions made in this paper can be summarized as follows.

1) An innovative application of the region growing algorithm is demonstrated to point-based methods, where it helps in removing background descriptors from the object template.
2) Unlike previous approaches [10], [11], an online update mechanism for the object model is proposed, where the stable descriptors are projected onto the current template using an AT model. AT takes into account the physical displacement of the object in the scene.
3) The new templates provide *temporal information*, while the projected points from previous templates provide *stable information*, and hence, leads to more robust tracking under challenging situations. This solves the so-called plasticity-versus-stability dilemma [7] in tracking to some extent.
4) A novel K-D tree-based classifier is used to differentiate between a case of occlusion from that of a pose change due to out-of-plane rotations. It is one among very few papers that attempt to deal with pose change due to out-of-plane rotations. A new method is proposed for detecting pose change due to out-of-plane rotations which is based on the spread of point features projected through AT.

The rest of this paper is organized as follows. An overview of related works is provided in the next section. The problem definition is provided in Section III. The proposed tracking method is explained in Section IV. Section V describes the design of visual servo controller. The simulation and experimental results are provided in Section VI followed by conclusion in Section VII.

II. Related Works

One of the earliest version of human-following robot used a camera to detect and track LED lights carried by the person being tracked [12]. The robot by Hirai and Mizoguchi [13] tracked human back and shoulder to follow a person. The robustness is achieved through template matching. Morioka *et al.* [14] used a intelligent sensor network comprising of multiple external cameras to allow a robot to follow a human. They later use laser range finder and Kinect to improve the detection of humans in the environment [15], [16]. Yoshimi *et al.* [17] developed a human-following robot which uses a vision-based algorithm for identifying individuals from color and texture of their clothes. Hu *et al.* [18] have used 3-D meanshift tracking and LRF-based leg tracking to locate humans. They also use anticipatory human walking model to improve the following behavior. In a very recent work, Han and Jin [19] used sound signal to locate and track humans. Among all these approaches, vision-based approach is most popular, thanks to the recent advancement in human detection and tracking algorithms [20], [21] and the availability of low cost cameras which could be easily mounted on a robot.

This paper focuses on improving the performance of currently available vision-based approaches for a human following robot. The controller for such a robot consists of two parts: 1) vision-based algorithm for tracking a person seen through its on-board camera and 2) a servo controller that drives the motor to keep the person in its FOV while maintaining a constant distance from the target in the environment. Readers can refer to [22] and [23] for a better understanding of visual servoing principles. An overview of the current state-of-the-art in visual servoing techniques could be found in [24] and [25]. Visual servoing methods are broadly classified into three categories. In position-based visual servoing (PBVS), the control objectives and control law are specified in the actual 3-D Cartesian space. The control law could be decoupled from the image processing part that is required for detecting and tracking features. On the other hand, in image-based visual servoing (IBVS), the control law is defined directly in image space. The third category of methods use a combination of the above two methods to overcome each other's limitation to some extent [23].

Visual servoing has been applied extensively to wheeled mobile platforms and a brief overview is provided here to set the context for this paper. Localizing a dynamic target from a moving platform is always a problem. It can be solved to some extent by using a velocity estimator for the target as reported in [26]. The authors use an extended Kalman filter-based velocity estimator to determine the range and bearing of the observed leader. In [27], a homography-based visual servo control strategy is proposed for a mobile robot with on-board camera (eye-in-hand problem) to track a desired time varying trajectory defined by a prerecorded sequence of images. By comparing the feature points of an object from a reference image to that in the current image and the prerecorded sequence of images, geometric relationships are formulated to reconstruct the Euclidean coordinates of the target points with respect to the mobile robot. The controller objectives are then defined in Euclidean space and the unmeasurable translation error is corrected through a Lyapunov-based adaptive control strategy. Similarly, Mariottini *et al.* [28] have proposed an IBVS method that exploits epipolar geometry for driving a nonholonomic robot toward to the desired target. This allows them to work without 3-D information. It still requires having at least partially calibrated cameras. This limitation was removed in [29], where a Lyapunov-based feedback control law is proposed that can compensate for the lack of depth information and the lack of precise visual parameters. Similarly, Jean and Lian [4] developed a visual servo controller based on sliding mode control technique. In another work, Fang *et al.* [5] developed an adaptive active visual servoing system that solves the FOV problem in mobile robot navigation. Active vision systems have separate actuators for cameras which are controlled separately from those of wheels. In yet another recent work, Wang *et al.* [6] have proposed a PBVS system that does not require direct position measurement. The robot positions are estimated using visual features combined with directions and velocity measured using IMU and compass sensors.

The main focus of this paper is on nonholonomic robots that primarily use vision sensors for detecting and following a human. As stated earlier, the visual controller for such robots consists of two parts: 1) vision-based algorithm for tracking a person present in the camera's FOV and 2) a servo controller that attempts to keep the person within the camera's FOV while following him in the actual environment. The servo-controller implements a feedback linearization-based controller to overcome the limitations of a sliding mode controller [4]. It is shown through experiment that the resulting controller is free from chattering unlike the sliding-mode-based controller.

As far as human tracking is concerned, *tracking-by-detection framework* [7], [9], [30] is becoming increasingly popular that uses locally invariant features like SURF [8] or scale invariant feature transformation (SIFT) [31], [32] to recognize target in each frame. Since SURF is known to be computationally more efficient compared to SIFT features, the current review is limited only to SURF-based methods. However, the discussion is equally valid for other point features as well. The point-based methods suffer from several limitations. For instance, the availability of matching points may vary significantly from one frame to another. This becomes even more difficult with nonrigid targets like human which can undergo variation in shape, size, and pose during its motion. The problem is partially remedied by maintaining a good set of point features in the object model over time as suggested in [10] and [11]. Since the target may undergo significant variation over time, it would be necessary to generate new templates that can prevent tracking failure over a long run. However, choice of a wrong or a bad template may corrupt the source model thereby leading to drifting of the tracker [33]. It is also expected that the new template should be selected so as to reconcile the *stability-versus-plasticity* dilemma [7] which is a common problem in long term tracking. Gupta *et al.* [34] tried to solve the drifting problem by imposing a graphical structure on the selected template. Similarly, tracking failure due to out-of-plane rotations is considered as a difficult problem [35]–[37] which has not received adequate attention in the literature. Specifically, no effort has been made to either detect pose change due to out-of-plane rotations or to take appropriate actions to ameliorate its effect. The usual approach is to treat such occurrence as partial or full occlusions and then rely on some motion predictor to localize target.

Many of these problems are revisited in this paper and new approaches are proposed to solve these problems effectively. For instance, color-based region growing algorithm is used for removing point features corresponding to the background. This, in turn, solves the drifting problem. The problem of *stability-versus-plasticity* dilemma is resolved by using an online update mechanism that projects stable features from past templates onto the current template. This affine projection takes into account the physical motion of the target between two frames. An attempt is also made to detect out-of-plane rotations which is a novel contribution in the field of interest point-based tracking methods. Rather than treating such occurrences as partial or full occlusions, a K-D tree-based classifier is used to distinguish them from later effects. This helps us in having much finer control over the target tracking leading to lesser failures. The problem definition for the proposed work is presented next.

III. Problem Definition

The overall scheme for a human-following robot is shown in Fig. 1. It primarily constitutes two modules: 1) a tracking module that detects and tracks a target human in the video sequence collected from a camera mounted on the mobile robot and 2) a visual servo controller that drives the robot so as to keep the human within its view and following it maintaining a fixed distance from the target.

The algorithm for solving the tracking problem is described in the next section. The visual servoing controller will be described later in this paper.

IV. Tracking Algorithm

The tracking problem could be described as follows. Consider a set of frames $I_i, i = 0, 1, 2, \ldots, N$ of a video sequence, where an object identified by the user in the first frame is to be tracked by a mobile robot over all the frames. The object is identified by the user by selecting a rectangular region on the first frame. Let this rectangular region be denoted by W_0 corresponding to the first image I_0. Let $V(W_i) = \{(\mathbf{x}_1, \mathbf{v}_1, \omega_1), (\mathbf{x}_2, \mathbf{v}_2, \omega_2), \ldots, (\mathbf{x}_n, \mathbf{v}_n, \omega_n)\}$ be the set of SURF features of an image I_i within the window W_i, where $\mathbf{x}_i$ is the 2-D key point location of the 64-D SURF descriptor $\mathbf{v}_i$ and ω_i is the weight assigned to the SURF descriptor $\mathbf{v}_i$. V_x is used to denote the set of key point locations in a given window or a frame, and V_v is used to denote the corresponding set of descriptors. The tracking window W is represented by $W = (\mathbf{c}, w, h)$, where $\mathbf{c} = (c_x, c_y)$ center of the window with width w and height h. Given I_0, W_0, and $V(W_0)$, the task is to compute the tracking window $W_i(\mathbf{c}_i, w_i, h_i)$ for all image frames $i = 1, 2, \ldots, N$.

The proposed method for visual tracking of human using SURF is explained in the flowchart provided in Fig. 2. It primarily consists of four parts: 1) initialization; 2) tracking; 3) template update; and 4) error-recovery. In the *initialization* part, an object model is defined for the target, which is to be tracked in subsequent frames of the video. The initialization is carried out once in the first frame for a given video sequence. In the *tracking* module, an SURF-based tracker is used to locate the target in the next frame. A Kalman filter motion predictor is updated whenever the target is successfully detected by the SURF-based tracker. The *template update module*, selects a new template and updates the object model

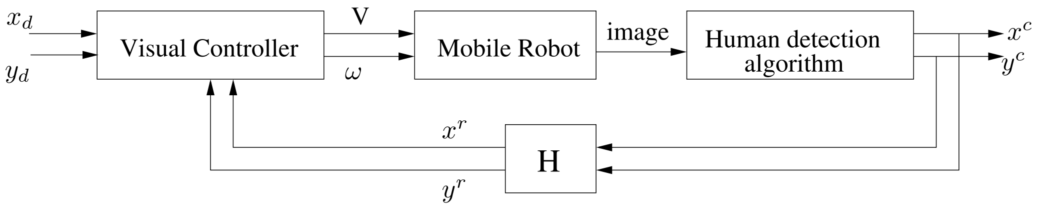


Fig. 1. Schematic of a human-following robot. Human detection and tracking algorithm locates the target to be followed by the robot. The location of the target in the image plane is used by the visual controller block to generate necessary motion commands for the mobile robot which allows it to follow the human. The depth value obtained from Kinect allows the robot to maintain a constant distance from the target.

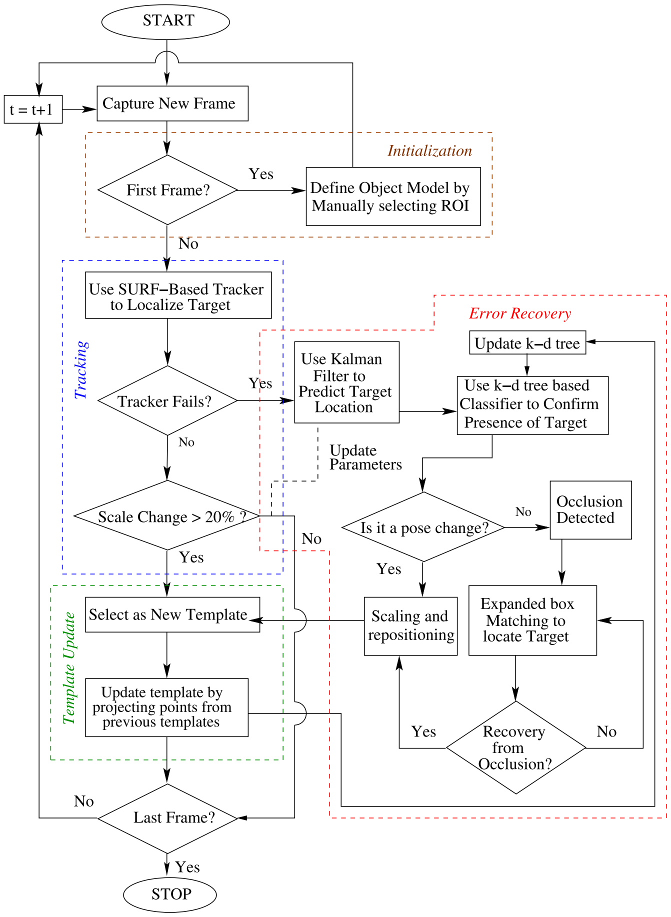

START
t = t+1
Capture New Frame
Initialization
First Frame?
Yes
Define Object Model by Manually selecting ROI
No
Tracking
Use SURF–Based Tracker to Localize Target
Tracker Fails?
Yes
No
Scale Change > 20% ?
Yes
No
Error Recovery
Use Kalman Filter to Predict Target Location
Update k–d tree
Use k–d tree based Classifier to Confirm Presence of Target
Update Parameters
Is it a pose change?
No
Occlusion Detected
Yes
Scaling and repositioning
Expanded box Matching to locate Target
Recovery from Occlusion?
Yes
No
Template Update
Select as New Template
Update template by projecting points from previous templates
No
Last Frame?
Yes
STOP

Fig. 2. Flowchart of the proposed human tracking algorithm.

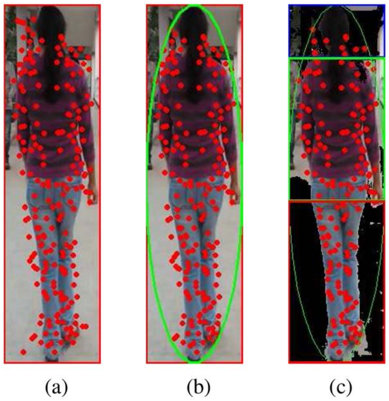


Fig. 3. Object template selection—set of SURF descriptors (a) in the selected rectangle region (b) after fitting the ellipse in rectangle region (c) after removing background inside the ellipse using region growing method.

by projecting the points from previous template to current template. The *error recovery module* provides the way to deal with pose change and occlusion. All these modules are described in detail in the following sections.

A. Template Initialization

In this module, the target human to be tracked is selected manually by drawing a rectangle box W_0 around it in the first frame I_0. The SURF descriptors present in this box, represented by $V(W_0)$ also include some descriptors from the background region. The inclusion of background descriptors into object template are most hazardous, since they increase the probability of negative correspondence during the tracking phase. In order to remove the background SURF descriptors from the object template, an ellipse that fits inside the rectangle W_0 is drawn, as shown in Fig. 3(b). The SURF descriptors present within this elliptical region (e_0) is represented as $V(e_0)$. The set $V(e_0)$ may still contain a few descriptors that belong to the background. In order to remove these remaining background descriptors from the object template, a region growing algorithm (flood-fill) is applied to the key-points that lie outside the elliptical region [38]. The flood-fill algorithm fills the adjacent cells having same pixel intensities with same color. In this way, the background region is segmented from the foreground, as shown in Fig. 3(c). All the descriptors, which belong to this segmented background region (B_R) are now removed from the object template. Mathematically, the object template (OT_0) can be defined as

$$\mathrm{OT}_0 = \big\{(\mathbf{x}_i, \mathbf{v}_i, \omega_i) | (\mathbf{x}_i, \mathbf{v}_i, \omega_i) \in V(W_0) \wedge \mathbf{x}_i \in e_0 \wedge \mathbf{x}_i \notin B_R,\ \omega_i = 15, i = 1, 2, \dots, p \le m\big\} \quad (1)$$

where m is the number of SURF descriptors present inside the window W_0. The object model (OM_0) is initialized with the object template. All the SURF descriptors of the object model are assigned an initial weight ω_i. The initial weight is selected so as to ensure that the descriptors survive at least for a few frames. In this paper, the initial weight for a descriptor is chosen as 15. This step is repeated whenever a new template is selected. This is a crucial step that helps in avoiding tracker drift for a longer duration.

B. Tracking

In this module, the target is localized in the next frame using direct SURF matching. The SURF correspondences are computed between the current object model and the new frame within a bounded rectangular region around the last target location. The bounded region is 10% bigger than the last target window. The wrong matches are removed by using random sample consensus based on homography [32], [39]. The tracking is considered successful if the percentage match between the source and destination region is found to be greater than a user defined threshold (θ). The percentage match is defined as

$$M_p = \frac{N_m}{N_s} \times 100 \tag{2}$$

where N_s is the number of SURF key-points present in the last selected template, and N_m is the number of matching key-points obtained through SURF correspondence. The match threshold (θ) used in this paper is 20. If the tracking is successful, the weights of the descriptors of the object model is updated as

$$\omega_i(t+1) = \begin{cases} \omega_i(t) + 2 & \text{if match is found} \\ \omega_i(t) - 1 & \text{otherwise.} \end{cases} \tag{3}$$

The tracker window is scaled and repositioned, as described in the following section. The new tracker location is used to train the Kalman filter motion predictor.

1) Scaling and Repositioning the Tracking Window: As the size of human being tracked may vary over time, the tracking window needs to be scaled and repositioned in order to avoid spurious background descriptors getting included into the template information. The bounding box around the human is divided into three parts. The torso region having maximum number of matching descriptors is used for computing the scaling factor, as explained in [40]. The repositioning of the tracker window is done by obtaining the center of the points in the torso region and then shifting it by using the body ratio of the human.

C. Template Update Module

The object model used for tracking the target by SURF-based tracker needs to be updated in order to accommodate for the temporal changes that the target may undergo during its motion. The update of object model in this case includes two steps.

1) A new template is selected only when the current template is inadequate to detect the target in the next frame using SURF correspondences. This happens when substantial movement has accumulated over time rendering the current template unfit for detecting the target in the new frame. Therefore, a new template is selected whenever the K-D tree-based classifier confirms the pose change case in the predicted tracker window. To avoid the frequent failure of the tracker, a new template is selected whenever the tracking window obtained from SURF-based tracker undergoes a scale change of more than 20% or the target recovered from occlusion.
2) Selecting a new template can detect the target over short-term, it might fail over longer run in the absence of stable descriptors, which matched frequently in previous frames. Hence, to maintain stability over long run, these stable features need to be incorporated into the new template so as to resolve the stability-versus-plasticity dilemma [7]. All the SURF descriptors, which have non-negative weight, along with their key-point locations are added to the new template by using the AT. AT reflects the displacement between the new template and the last template. Although the human motion is nonaffine in nature, the torso region under the assumption of upright human position can be considered as a rigid object and its motion can be considered, as an affine motion. The AT [41] model is given by

$$\begin{pmatrix} f_{x,k+1} \\ f_{y,k+1} \end{pmatrix} = \begin{pmatrix} a_0 & a_1 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} f_{x,k} \\ f_{y,k} \end{pmatrix} + \begin{pmatrix} a_2 \\ a_5 \end{pmatrix}. \tag{4}$$

The six parameters in the model is estimated using least-square method for the set of points obtained in the torso region through SURF correspondence between the two templates.

Therefore, the updated object model can be defined as

$$\begin{aligned} \mathrm{OM}_n = \big\{ &\mathrm{OT}_n \cup (\mathbf{x}_i, \mathbf{v}_i, \omega_i) | (\mathbf{x}_i, \mathbf{v}_i, \omega_i) \in \mathrm{OT}_{n-1} \\ &\wedge \mathbf{x}_i \in g({V_x}^{n-1}) \; \forall \; \omega_i \geq 0, i = 1, 2, \ldots, m \big\} \end{aligned} \tag{5}$$

where m is the number of SURF-descriptors present in the OT_{n-1} and $g(.)$ represents the AT. Note that a subset of the past features, which are occurring more frequently are projected onto the current template. This reduces the computational and memory requirement by avoiding linearly increasing number of descriptors in the object model with the increasing frames in a video. The computational and memory requirement of the proposed algorithm is much less compared to methods reported in [10] and [11]. The projected descriptors with their key-point locations lead to better matches in subsequent frames.

D. Error Recovery Module

This module is executed whenever the SURF-based tracker fails to detect the human. The tracker fails to localize the target in a frame, when the number of matching points between the source and the target windows falls below a certain threshold. Such a case might arise under two conditions: 1) the target is present but its appearance has changed significantly from the current object template due to pose change such as *out-of-plane rotations* and 2) the target is partially or fully occluded by other objects in the environment. Compared to other effects like variation in illumination or scaling, the pose change due to out-of-plane rotations lead to frequent tracking failures. In order to differentiate a case of pose change from that of an occlusion, a K-D tree-based classifier is used is this algorithm.

1) K-D Tree Classifier: In this algorithm, a K-D tree is used for feature matching or to classify a tracker window whether its belongs to the foreground or to the background. Directly matching raw features extracted from the current tracker window with the older templates represents an reasonable similarity measurement and can be used for classification. The problem is that the direct feature matching via linear search can soon become intractable with the increasing number of templates to be processed and prevent its real-time implementation. In contrast, a tree structure is an efficient data structure for feature matching [42] and can provide online processing in classification due to its efficiency. In the proposed algorithm, a K-D tree is built over all the template features and a tree search is performed for the features extracted from the query tracker window. The idea of using K-D tree for feature matching is inspired by the following facts.

The K-D tree reduces the search complexity from linear to logarithmic and compares one dimension of the high-dimensional features each time, and thus, avoids the distance computations (the most time-consuming part in finding the correct nearest neighbor for a query feature with linear search). The second fact is that the distance ratio method [43] is an effective way for verifying putative matches. In distance ratio method, a correct match requires the ratio between the distances of the closest and second closest neighbor to the query feature to be below some given threshold. This paper focus more on the applicability of the tree structure in identifying foreground descriptors by using distance ratio technique.

2) Construction of K-D Tree: The K-D tree is initially build with the descriptors of the first template and these descriptors are pushed into the feature pool D. Whenever a new template is selected, its descriptors are inserted to the feature pool D and the K-D tree is reconstructed over all the features in D. Hence, D always includes the features extracted from all the templates and the tree is updated every time a new template is selected. When the SURF-based tracker fails to converge due to unavailability of sufficient number of matching points, the descriptors of the tracker window, whose location is predicted by the Kalman filter is subjected to a K-D tree-based classifier to distinguish the above two situations. Let Q be the set of SURF descriptors extracted from the current tracker window. For each descriptor in Q, it will go through a tree search in the current K-D tree and top two nearest neighbors are returned. Distance ratio is applied to determine whether the closest one is a good match or not. If it is a good match, then it is considered as a foreground descriptor, otherwise, it is considered as a background descriptor. If the number of foreground descriptors is more than 1.5 times of background descriptors in the tracker window, then it is considered a case of pose change, otherwise, it is considered a case of occlusion.

3) Dealing With Pose Change: Once the K-D tree-based classifier confirms that the human is present in the tracker window predicted by the Kalman filter, it is necessary to figure out that the pose change is due to in-plane-rotations or out-of-plane rotations. If the target undergoes a significant amount of out-of-plane rotation (side pose) and the object model is updated with this template, then a number of background descriptors will get include into the object model, and results in failure of the tracker. Therefore, the object model is only updated with the template, in which pose change is either due to in-plane rotations or due to small amount of out-of-plane rotations. The out-of-plane rotations can be checked by projecting the points of the previous template onto the current template through the AT obtained between the two templates, as shown in Fig. 4. As one can see, the projected points get concentrated on a line during significant out-of-plane rotations and this information could be utilized to avoid tracking failure.

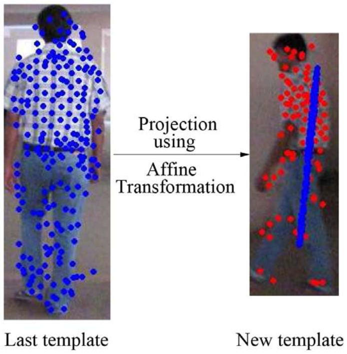


Fig. 4. Detecting out-of-plane rotations by using the aspect ratio of points projected using AT.

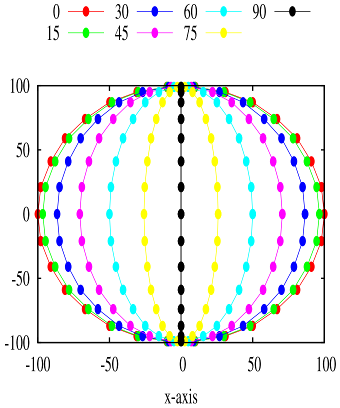


Fig. 5. Thirty points are taken on the boundary of a circle in x–y plane. With increasing amount of out-of-plane rotation, these points get closer to each-other and finally merged in a straight line at 90° rotation.

To prove that the points get concentrated with increasing amount of out-of-plane rotations, 30 points are taken on the boundary of a circle in x–y plane, as shown in Fig. 5 by red circles. In an image, the out-of-plane rotation is the rotation about y-axis and its rotation matrix is given by

$$R(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}. \tag{6}$$

In this experiment, a rotation of 0, 15, 30, 45, 60, 75, and 90° is applied on these points and these points are plotted, as shown in Fig. 5. The value of z-coordinate is taken as zero. One can see in Fig. 5 that these points get concentrated with increasing amount of rotations and finally at 90° rotation, these points get concentrated on a straight line. This is pictorially demonstrated on data-set D6 in Fig. 6.

4) Tracker Recovery From Full Occlusions: Once the occlusion is detected, the location of the tracker window is provided by a Kalman filter-based predictor. The confirmation about occlusion is obtained from the K-D tree-based classifier which identifies it as a background. However, if the target is occluded for several consecutive frames, motion predictor might drift from the actual trajectory of the target in absence of update over these frames. Therefore, in such cases, the SURF-based tracker is used to search around the location predicted

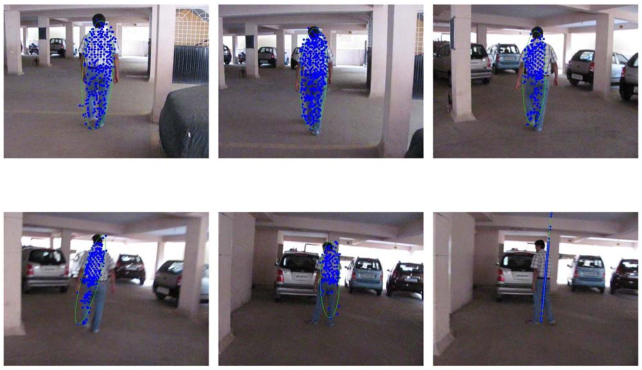


Fig. 6. Demonstration of projected points during out-of-plane rotation on data-set D6.

by the Kalman filter with increasing size of the tracker window. In the worst case, a frame-to-frame matching between the current image and the past template image could be used to locate the target once it recovers from the occlusion. In most cases, local search around predicted position is adequate to recover the target.

In order to implement the proposed visual human tracking algorithm on a mobile robot, a visual servo controller is designed in the next section which takes data from the tracking algorithm and gives motion commands to the mobile robot.

V. Visual Servo Controller

To design a visual servo controller for the mobile robot, equations are derived from the kinematic model of the robot and the pin-hole model of the camera. Then, the problem is formulated using these equations.

Let us consider a differential drive mobile robot (P3-DX) carrying a fixed camera (Kinect). The task of the robot is to track a moving human on a 2-D plane (x–y plane). As shown in Fig. 7, the camera is mounted along the heading direction of the mobile robot. A coordinate frame $\{R_c\}$ is attached to the optical center of the camera P_c. The x-axis being along the optical axis of the camera. z-axis is in the direction pointing out of this paper. Using Fig. 7, the kinematic equations of a mobile robot can be obtained as follows.

A. Kinematic Model of the Mobile Robot

The posture of a mobile robot is presented by its position that is the middle point of the two driving wheels, and the heading direction θ. Fig. 7 shows the position of the robot expressed in the X–Y coordinates. The posture vector of mobile robot is presented as $P_c = (x_c, y_c, \theta)^T$ with respect to initial frame $\{R_0\}$. The mobile robot has 2° of freedom. It can move along the x-axis and rotate around the z-axis. The mobile robot's motion is controlled by the vector $\mathbf{q} = (v, \omega)^T$, where v is the linear velocity of the robot and ω is the angular velocity of the robot. Based on the assumption that the mobile robot moves on a 2-D plane without slips, the kinematic equation of a mobile robot can be written as

$$\dot{P}_c = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{q}. \tag{7}$$

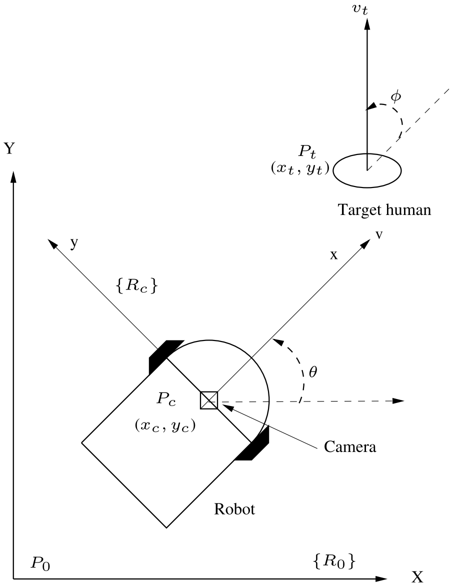


Fig. 7. Mobile robot mounted with a fixed camera tracks a moving human on a 2-D plane.

B. Pinhole Camera Model

The pinhole camera model describes the mathematical relationship between the coordinates of a 3-D point and its projection onto the image plane of an ideal pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light [44]. Let $({}^{c}x_t, {}^{c}y_t, h)$ is the position vector of the human center P_t with respect to the frame $\{R_c\}$, h is a known positive constant, (α_t, β_t) is the image coordinate of the point projected on the image plane, and f is the focal length of the camera. Using the pin-hole model for the camera as shown in Fig. 7, the following relationships are obtained:

$$\frac{{}^{c}x_t}{f} = \frac{{}^{c}y_t}{\alpha_t} = \frac{h}{\beta_t}. \tag{8}$$

C. Problem Formulation

Let the error posture between the target human and the robot be $P_e = (x_e, y_e, \theta_e)$. The error posture is a transformation of the target posture P_t from frame $\{R_0\}$ in a local coordinate frame $\{R_c\}$ with an origin of (x_c, y_c) and z-axis is the direction of θ_e

$$\begin{aligned} P_e &= \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} {}^{c}x_t \\ {}^{c}y_t \\ \phi \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t - x_c \\ y_t - y_c \\ \omega_t - \omega \end{bmatrix}. \end{aligned} \tag{9}$$

The equation of relative motion between the target human and the mobile robot can be calculated as follows [45]:

$$\begin{aligned} \dot{P}_e &= \begin{bmatrix} {}^{c}\dot{x}_t \\ {}^{c}\dot{y}_t \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -v + v_t\cos\phi + {}^{c}y_t.\omega \\ v_t\sin\phi - {}^{c}x_t.\omega \\ \omega_t - \omega \end{bmatrix} \\ &= \begin{bmatrix} -v + v_{tx} + {}^{c}y_t.\omega \\ v_{ty} - {}^{c}x_t.\omega \\ \omega_t - \omega \end{bmatrix} \end{aligned} \tag{10}$$

where v_t is the linear velocity of the target human, ϕ is the angle between moving directions of the target human and the mobile robot, ω_t is the angular velocity of the target human, and v_{tx} and v_{ty} are the target human velocity components, $v_{tx} = v_t \cos\phi$ and $v_{ty} = v_t \sin\phi$, respectively.

The objective of human tracking is to drive the mobile robot to keep the target human always in sight of the camera. This objective can be achieved if we keep the position of the human center P_t on the heading direction and within a short distance of the mobile robot, that is

$$^{c}x_t = x_d \qquad ^{c}y_t = 0. \tag{11}$$

D. Visual Servo Control Design

For the objective of human tracking (11), the tracking errors, e_x and e_y can be defined as

$$e = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} ^{c}x_t - x_d \\ ^{c}y_t \end{bmatrix}. \tag{12}$$

The system (10) can be rewritten in terms of tracking errors as

$$\dot{e} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix} = \begin{bmatrix} -v + v_{tx} + {}^{c}y_t.\omega \\ v_{ty} - {}^{c}x_t.\omega \end{bmatrix} \tag{13}$$

or

$$\dot{e} = \begin{bmatrix} \dot{e}_x \\ \dot{e}_y \end{bmatrix} = \begin{bmatrix} -1 & ^{c}y_t \\ 0 & -^{c}x_t \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} v_{tx} \\ v_{ty} \end{bmatrix}. \tag{14}$$

In this paper, the visual servo controller is designed using the approach of dynamic inversion. In dynamic inversion approach, the controller is synthesized such that the following stable linear error dynamics are satisfied:

$$\dot{e} + K.e = 0 \tag{15}$$

where $K = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$ is the gain matrix. Substituting the values of $\dot{e}$ and e, we get

$$\begin{bmatrix} -1 & ^{c}y_t \\ 0 & -^{c}x_t \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} + \begin{bmatrix} v_{tx} \\ v_{ty} \end{bmatrix} + \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} ^{c}x_t - x_d \\ ^{c}y_t \end{bmatrix} = 0. \tag{16}$$

After rearranging the equation, the desired control inputs are

$$v = v_{tx} + k_1\left({}^{c}x_t - x_d\right) + \frac{^{c}y_t\left(k_2{}^{c}y_t + v_{ty}\right)}{^{c}x_t} \tag{17}$$

$$\omega = \frac{\left(k_2{}^{c}y_t + v_{ty}\right)}{^{c}x_t}. \tag{18}$$

The values of $({}^{c}x_t, {}^{c}y_t)$ can be obtained using the pin-hole camera (8) and the relative target velocities (v_{tx}, v_{ty}) are obtained using the Kalman filter. The Kalman filter provides the estimate of the velocities in image plane, which can be converted to real-world velocities using the interaction matrix as explained in [46].

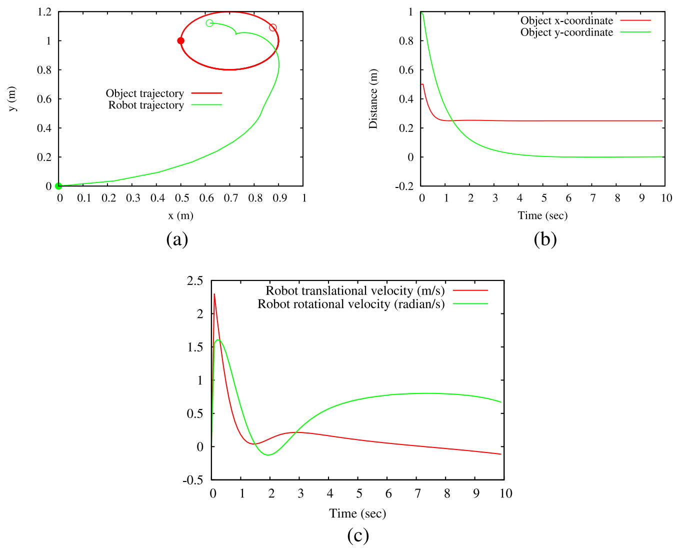


Fig. 8. Simulation results for tracking a circular trajectory. The filled and empty circles show the starting and ending point of the trajectory, respectively. (a) 2-D position trajectories in the world coordinate frame. (b) Target coordinate in robot frame. (c) Velocity trajectories.

E. Simulation Results

To illustrate the performance of the proposed control scheme, simulations were performed in which the mobile robot was required to track various trajectories. Three kinds of object trajectories, namely, straight-line, circle, and eight-shape curve motion, were generated by a path planner in the frame of the x–y coordinates. The units of x and y coordinates are in meter in all the experiments. In the simulations, for using the proposed control scheme, object trajectories were transformed into the robot local coordinate frame $\{R_c\}$ before generating control commands. Only later two trajectories are described below in the interest of the space limitation.

1) Example 1 (Tracking Object Which Moves in Circular Trajectory): Let the object moves counterclockwise in a circle around (0.7, 1) with respect to $\{R_0\}$ [4] and set the motion as follows:

$$x_t = 0.7 - 0.2\cos(t) \quad y_t = 1.0 - 0.2\sin(t). \tag{19}$$

Therefore, the velocity of moving object is described as follows:

$$\frac{dx_t}{dt} = 0.2\sin(t) = v_t\cos(\theta + \phi) \tag{20}$$

$$\frac{dy_t}{dt} = -0.2\cos(t) = v_t\sin(\theta + \phi). \tag{21}$$

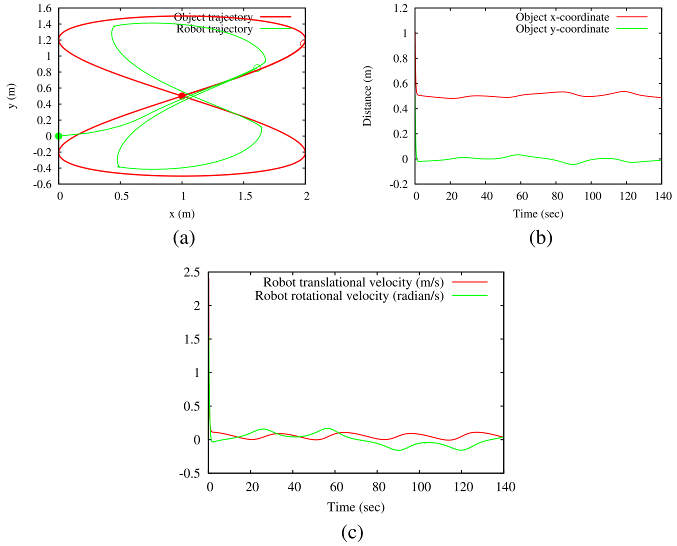
The relative velocity between the object and the mobile robot can be derived as follows:

$$v_{tx} = v_t\cos(\phi) = -0.2\sin(\theta - t) \tag{22}$$

$$v_{ty} = v_t\sin(\phi) = -0.2\cos(\theta - t). \tag{23}$$

The robot initial pose is taken as $P_c(0) = (0, 0, 0.0)^T$. The desired tracking goal is set as $x_d = 0.25$ and $y_d = 0$. The values of k_1 and k_2 are taken as 3 and 1, respectively.

Fig. 8 shows the simulation result for tracking a circular trajectory in the world coordinate ($\{R_0\}$ frame). As can be



(a)

(b)

(c)

Fig. 9. Simulation results for tracking an eight-shape trajectory. Filled and empty circles show the starting and ending point of a trajectory, respectively. (a) 2-D position trajectories in world coordinate frame. (b) Target coordinates in the robot coordinate frame. (c) Velocity trajectories.

seen in this figure, the proposed controller results in a smooth trajectory. In order to maintain the desired distance x_d, the robot departs from the target when the target approaches to it. Fig. 8(b) indicates that the tracking result achieves the tracking goal, i.e., ${}^{c}x_t = x_d = 0.25$ and ${}^{c}y_t = y_d = 0.0$, before 4 s. The velocities given to the robot is provided in Fig. 8(c).

2) Example 2 (Tracking Object Which Moves in Eight-Shape Trajectory): Let the object move in an eight-shape trajectory [47], which is defined with respect to $\{R_0\}$ as follows:

$$x_t = 1 + \sin\left(\frac{t}{10}\right) \quad y_t = 0.5 + \sin\left(\frac{t}{20}\right). \tag{24}$$

Therefore, the velocity of moving object is described as follows:

$$\frac{dx_t}{dt} = \frac{1}{10}\cos\left(\frac{t}{10}\right) = v_t\cos(\theta + \phi) \tag{25}$$

$$\frac{dy_t}{dt} = \frac{1}{20}\cos\left(\frac{t}{20}\right) = v_t\sin(\theta + \phi). \tag{26}$$

The relative velocity between the object and the mobile robot can be derived as follows:

$$v_{tx} = v_t\cos(\phi) = \frac{1}{10}\cos\left(\theta - \frac{t}{10}\right) \tag{27}$$

$$v_{ty} = v_t\sin(\phi) = \frac{1}{20}\cos\left(\theta - \frac{t}{20}\right). \tag{28}$$

The robot initial pose is taken as $P_c(0) = (0, 0, 0.0)^T$. The desired tracking goal is set as $x_d = 0.5$ and $y_d = 0$. The values of k_1 and k_2 are taken as 3.

Fig. 9 shows the simulation result of tracking an eight-shape trajectory in the world coordinate ($\{R_0\}$ frame). Fig. 9(b) indicates that the robot fluctuates around the tracking goal, i.e., ${}^{c}x_t = x_d = 0.5$ and ${}^{c}y_t = y_d = 0.0$, as the object changes the pose frequently in making an eight-shape trajectory. The velocities given to the robot is provided in Fig. 9(c).

TABLE I
SUMMARY OF ATTRIBUTES FOR DATA-SETS USED FOR EVALUATING THE PERFORMANCE OF THE PROPOSED ALGORITHM

Data-set		Total no. of frames	Scaling (Upto)	No. of templates generated	Success Rate
ETH	D1	380	179%	25	98.15
	D2	251	214%	21	92.82
Youtube	D3	166	17%	9	80.12
	D4	266	87%	12	90.22
Own	D5	291	14%	19	94.15
	D6	979	51%	29	86.41

VI. EXPERIMENTAL RESULT

The experimental results are provided in two parts. In the first part, the performance of the proposed visual human tracking algorithm is evaluated while in the second part, the results obtained from the actual robot are analyzed.

A. Tracking Results

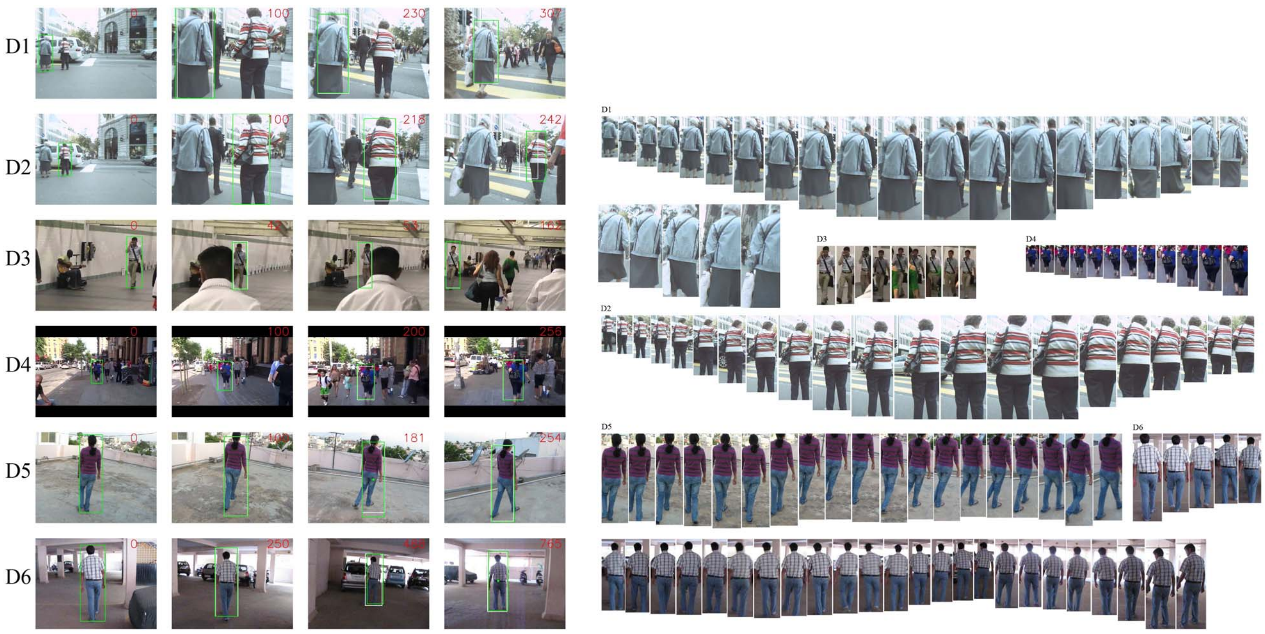
The performance of the proposed tracking algorithm is analyze by testing it on a number of video datasets collected from different sources. These data-sets exhibit different kinds of situations and pose different challenges. This is different from a person detection problem [21] or a video surveillance problem, where one needs to detect and track persons in the scene no matter how small they appear. The summary of various data-sets used in this paper are provided in Table I. In total, six sets of videos are taken, out of which two are from pedestrian data-set of ETH Zurich [48], two sets are from Youtube, and the last two are our own. The videos depict several challenging situations like variation in illumination, scaling, occlusion, camera motion, and change in pose with *out-of-plane rotations*.

The images in each video has a resolution of 640×480. The algorithms is implemented in C/C++ using OpenCV 2.0 library on a system with intel i7 processor running with 8 GB of RAM. The tracking results for these data-sets are available on website [49] for the inspection and a few snapshots are shown in Fig. 10(a).

The templates generated by the proposed algorithm for all video data-sets are shown in Fig. 10(b). As one can see, the templates generated exhibit all kinds of variations as discussed earlier in this paper. Online update of object model helps in tracking a person for a longer duration by avoiding tracking failures arising out of appearance change that may occur over time.

It can be seen in Fig. 11(a) that the maximum number of descriptors in the object model is less than 1000, which is quite less as compared to the methods [11] that save a number of templates in the memory. Therefore, it can be concluded that the memory requirement of the proposed tracker is quite low, as compared to the existing methods. Computational efficiency of the proposed tracker can be seen in Fig. 11(b). Average computation time for a video is around 150 ms, i.e., 6 frames/s, which is comparable to those reported by [30] and sufficient for tracking a human with mobile robot. The tracking results of real-time implementation of visual human tracking algorithm are available online [50].

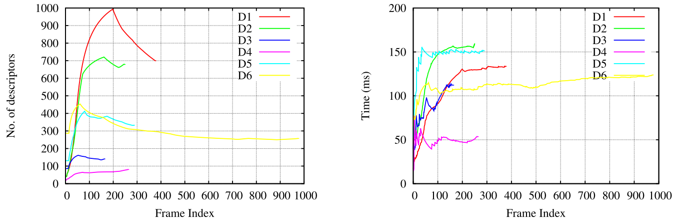
In order to assess the utility of the proposed approach, its performance is compared with the well known algorithms



(a)

(b)

Fig. 10. (a) Tracking results for different datasets. The green window is the main tracking window for the target being tracker while white window is the estimated target location obtained from a predictor. (b) Templates generated by the algorithm for these datasets. As one can see, the templates contains different poses of the human with a varying degree of scaling and illumination.



(a)

(b)

Fig. 11. (a) Average number of descriptors and (b) average computation time per frame for data sets mentioned in Table I.

like mean-shift algorithm [51] and the SURF-based mean-shift algorithm (SBMS) [52].

Three measures are used, namely, percentage overlap, success rate, and computation time to quantify the efficacy a given tracking algorithm. In order to compute these values, the ground truths are labeled manually for all the video datasets. The percentage overlap between the tracker window (W_t) and its ground truth (W_g) is defined as

$$\% \text{ Overlap} = \frac{\text{area}(W_g \cap W_t)}{\text{area}(W_g \cup W_t)} \times 100. \tag{29}$$

The overlap is measured as the percentage of area of the ground truth window that is common with the tracker window.

Similarly, the success rate [53] of the algorithm is defined as the ratio of number of frames, where the target is correctly detected by the tracking algorithm (n) to the total number of frames (N). Mathematically, it is computed as follows:

$$\text{Success Rate} = \frac{n}{N} \times 100. \tag{30}$$

For a given frame, if the overlap between the tracker window and its ground truth exceeds 50%, the target detection is considered to be successful. The success rate for the videos is mentioned in Table I. A tracking algorithm is considered to have a better performance if it has higher values for % overlap and success rate with lower value of computation time per image.

The performance comparison among the tracking algorithms in terms of the above quantitative measures is provided in Table II. It is clear from the table that success rate of the proposed algorithm is high for all the data-sets, while mean-shift tracker tracks the human successfully only for data-set D5, in which human color is different from the background color and human did not get occluded. However, the computation time of the proposed algorithm is greater than the mean-shift algorithm. As compared to the SBMS tracker, the proposed tracker outperforms in all the measures.

B. Human-Following Robot

The schematic block diagram of a human-following robot is shown in Fig. 1. It consists of a P3-DX mobile robot from Adept Mobilerobots [54] equipped with a camera, a human detection algorithm and a visual controller. The desired position of the human center in the robot frame is represented by the pair (x_d, y_d). V and ω are the translational and rotational velocities generated by the visual controller for the mobile robot so that it can reach the desired position. (x_c, y_c) is the human center position obtained by the human detection algorithm in the image frame. $\mathbf{H}$ represents the relationship between the variables defining the relative posture of the follower robot to the target human (x^r, y^r), and the image features.

TABLE II
QUANTITATIVE PERFORMANCE COMPARISON OF VARIOUS TRACKING ALGORITHMS

Data-set	Algorithm	MS [51]	SBMS [52]	PA
D1	Success Rate	0	21.84	98.15
	Average % overlap	11.12	34.3	75.4
	Average time (ms)	121	682	134
D2	Success Rate	0	8.36	92.82
	Average % overlap	8.23	19.78	71.7
	Average time (ms)	121	565	179
D3	Success Rate	22.8	54.81	80.12
	Average % overlap	15.4	41.3	68.9
	Average time (ms)	120	486	101
D4	Success Rate	20.67	60.5	90.22
	Average % overlap	29.1	50.54	61.6
	Average time (ms)	121	423	51
D5	Success Rate	100	53.9	94.15
	Average % overlap	74.9	51.7	69.1
	Average time (ms)	121	748	160
D6	Success Rate	51.68	79.8	86.41
	Average % overlap	41.8	60.3	71.5
	Average time (ms)	124	771	131

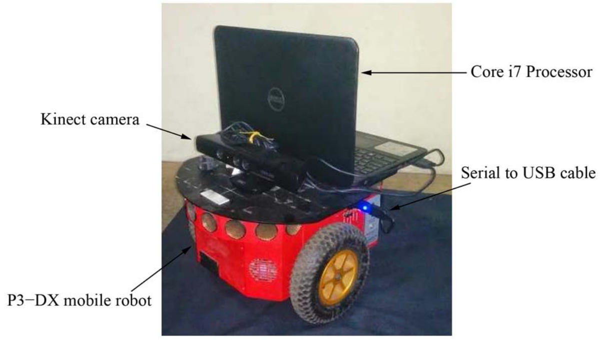
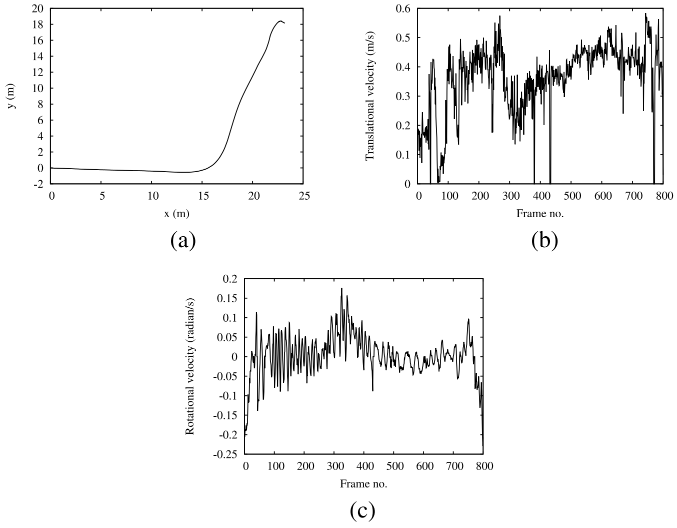


Fig. 12. Experimental setup for a human-following robot. This comprises of a P3DX mobile platform from Adept [54] and a Kinect. An external computer is used to run the tracking algorithm and the visual controller required for this task.

The experimental setup used for implementing the algorithms is shown in Fig. 12. The Kinect is used as the visual sensor for detecting and tracking the human walking in front of the robot. The visual controller takes the human position in the robot frame as the input and with reference to the desired goal generates the velocity commands for the mobile robot. The mobile robot moves with the generated velocities to perform human following. The robot maintains a constant distance from the human target by using Kinect depth measurement. In order to analyze the performance of the robot, an experiment is performed where the subject walks in a corridor in front of the robot. Few snapshots of the person being tracked is shown in Fig. 14 and the resulting trajectory of the robot is shown in Fig. 13(a). The translational and rotational velocities of the robot is shown in Fig. 13(b) and (c), respectively. As one can see, the robot trajectory is smooth and control velocities are well within the limits of the system. The average translational velocity of the robot is 0.4 m/s.

C. Discussion on Performance Comparison

To summarize, the advantages of the proposed approach is demonstrated by providing performance comparison with previous works only at individual module level, rather than for the full system. For instance, the performance of the vision-based tracking algorithm is compared with some of the earlier works as shown in Table II. This improvement is achieved using lesser number of features compared to earlier works. Similarly, for the visual servoing module, an improved version of the controller used in [4] is proposed in this paper which uses feedback linearization to overcome the problem of chattering present in sliding mode controller used in the previous work. Direct comparison for the overall system is difficult due to several factors such as, unavailability of common dataset, nonuniformity of sensors used, etc. A more rigorous comparison with uniform operating conditions will remain as a part of the future work.



(a)

(b)

(c)

Fig. 13. Robot motion trajectories in the actual experiment, where a subject is tracked and followed in an outdoor environment. As one can see the velocity magnitudes are well within the physical limits of robot. The high frequency in the velocity component is due to sensor noise. (a) Robot trajectory. (b) Translational velocity. (c) Rotational velocity.

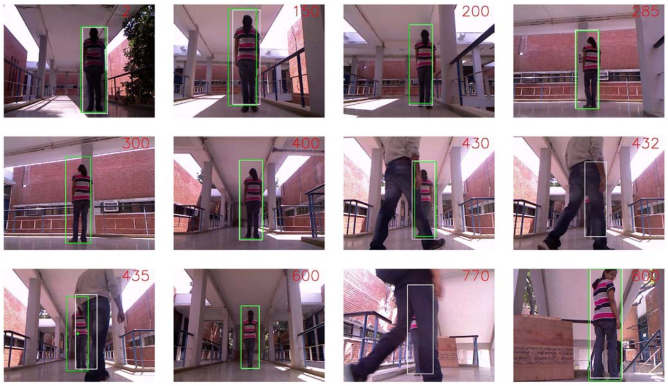


Fig. 14. Snapshots of real-life experiment where the subject is walking in front of a robot in an outdoor environment.

VII. CONCLUSION

In this paper, we revisit the problems associated with a human-following robot that uses a vision sensor for detecting and tracking a human subject. The problem consists of two parts: first, detecting and tracking humans in a video sequence recorded from the onboard camera and second, designing a visual servo controller that generates necessary motion commands required for following the human target. The first part is fraught with several challenges, such as, variation in shape, size, illumination, scale and pose, partial and full occlusion, pose-change due to out-of-plane rotations, etc. On the other hand, the visual servo controller has to deal with problems like inaccurate odometry information, FOV constraints, nonholonomic motion, and camera calibration errors. An attempt is made to solve some of these problems in this paper. The proposed method uses a tracking-by-detection framework which uses point features for detection. The tracking algorithm uses a dynamic object model that evolves over time to accommodate for temporal changes while the stability is ensured by propagating stable features using affine motion model. The spread of the projected points is used for detecting pose change due to *out-of-plane rotations*, which is a novel contribution made in this paper. A K-D tree-based classifier used to differentiate a case of full occlusion from that of partial occlusion and pose change. The visual servo controller uses a feedback-linearization-based technique to ensure stability during motion. The efficacy of the approach is demonstrated through various simulation and experiments.

References

[1] N. Hirose, R. Tajima, and K. Sukigara, "Personal robot assisting transportation to support active human life—Human-following method based on model predictive control for adjacency without collision," in *Proc. IEEE Int. Conf. Mechatronics (ICM)*, Nagoya, Japan, 2015, pp. 76–81.

[2] N. Kejriwal, S. Garg, and S. Kumar, "Product counting using images with application to robot-based retail stock assessment," in *Proc. IEEE Int. Conf. Technol. Pract. Robot Appl. (TePRA)*, Woburn, MA, USA, 2015, pp. 1–6.

[3] W. E. Dixon, D. M. Dawson, E. Zergeroglu, and A. Behal, "Adaptive tracking control of a wheeled mobile robot via an uncalibrated camera system," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 341–352, Jun. 2001.

[4] J.-H. Jean and F.-L. Lian, "Robust visual servo control of a mobile robot for object tracking using shape parameters," *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 6, pp. 1461–1472, Nov. 2012.

[5] Y. Fang, X. Liu, and X. Zhang, "Adaptive active visual servoing of nonholonomic mobile robots," *IEEE Trans. Ind. Electron.*, vol. 59, no. 1, pp. 486–497, Jan. 2012.

[6] K. Wang, Y. Liu, and L. Li, "Visual servoing trajectory tracking of nonholonomic mobile robots without direct position measurement," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 1026–1035, Aug. 2014.

[7] S. Gu, Y. Zheng, and C. Tomasi, "Efficient visual object tracking with online nearest neighbor classifier," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, Queenstown, New Zealand, 2010, pp. 271–282.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[9] W. Kloihofer and M. Kampel, "Interest point based tracking," in *Proc. Int. Conf. Pattern Recogn. (ICPR)*, Istanbul, Turkey, 2010, pp. 3549–3552.

[10] A. M. Gupta, B. S. Garg, C. S. Kumar, and D. L. Behera, "An online visual human tracking algorithm using SURF-based dynamic object model," in *Proc. Int. Conf. Image Process. (ICIP)*, Melbourne, VIC, Australia, 2013, pp. 3875–3879.

[11] D. Zhou and D. Hu, "A robust object tracking algorithm based on SURF," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Hangzhou, China, 2013, pp. 1–5.

[12] Y. Nagumo and A. Ohya, "Human following behavior of an autonomous mobile robot using light-emitting device," in *Proc. 10th IEEE Int. Workshop Robot Human Interact. Commun.*, Bordeaux, France, 2001, pp. 225–230.

[13] N. Hirai and H. Mizoguchi, "Visual tracking of human back and shoulder for person following robot," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, vol. 1. Kobe, Japan, 2003, pp. 527–532.

[14] K. Morioka, J.-H. Lee, and H. Hashimoto, "Human-following mobile robot in a distributed intelligent sensor network," *IEEE Trans. Ind. Electron.*, vol. 51, no. 1, pp. 229–237, Feb. 2004.

[15] F. Hoshino and K. Morioka, "Human following robot based on control of particle distribution with integrated range sensors," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Kyoto, Japan, 2011, pp. 212–217.

[16] K. Morioka, Y. Oinaga, and Y. Nakamura, "Control of human-following robot based on cooperative positioning with an intelligent space," *IEEJ Trans. Electron. Inf. Syst.*, vol. 131, no. 5, pp. 1050–1058, 2011.

[17] T. Yoshimi *et al.*, "Development of a person following robot with vision based target detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, 2006, pp. 5286–5291.

[18] J.-S. Hu, J.-J. Wang, and D. M. Ho, "Design of sensing system and anticipative behavior for human following of mobile robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 4, pp. 1916–1927, Apr. 2014.

[19] J. Han and T. Jin, "Sound source based mobile robot control for human following in a networked intelligent space," *Int. J. Control Autom.*, vol. 8, no. 7, pp. 67–74, 2015.

[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. (CVPR)*, vol. 1. San Diego, CA, USA, 2005, pp. 886–893.

[21] A. A. Mekonnen, C. Briand, F. Lerasle, and A. Herbulot, "Fast HOG based person detection devoted to a mobile robot with a spherical camera," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, 2013, pp. 631–637.

[22] S. Hutchinson, G. D. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Autom.*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[23] E. Malis, F. Chaumette, and S. Boudet, "21/2d visual servoing," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 238–250, Apr. 1999.

[24] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[25] F. Chaumette and S. Hutchinson, "Visual servo control. II. Advanced approaches [tutorial]," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[26] A. K. Das *et al.*, "Real-time vision-based control of a nonholonomic mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2. Seoul, South Korea, 2001, pp. 1714–1719.

[27] J. Chen, W. E. Dixon, M. Dawson, and M. McIntyre, "Homography-based visual servo tracking control of a wheeled mobile robot," *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 406–415, Apr. 2006.

[28] G. L. Mariottini, G. Oriolo, and D. Prattichizzo, "Image-based visual servoing for nonholonomic mobile robots using epipolar geometry," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 87–100, Feb. 2007.

[29] C. Wang, Y. Mei, Z. Liang, and Q. Jia, "Dynamic feedback tracking control of non-holonomic mobile robots with unknown camera parameters," *Trans. Inst. Measur. Control*, vol. 32, no. 2, pp. 155–169, 2009.

[30] W. He, T. Yamashita, L. Hongtao, and S. Lao, "SURF tracking," in *Proc. Int. Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 1586–1592.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Jan. 2004.

[32] L. Juan and O. Gwun, "A comparison of SIFT, PCA-SIFT and SURF," *Int. J. Image Process.*, vol. 3, no. 4, pp. 143–152, 2009.

[33] L. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 810–815, Jun. 2004.

[34] M. Gupta, S. Kumar, S. Garg, N. Kejriwal, and L. Behera, "A novel SURF-based algorithm for tracking a 'Human' in a dynamic environment," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Singapore, 2014, pp. 1004–1009.

[35] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, Providence, RI, USA, 2012, pp. 1838–1845.

[36] S. Oron, A. Bar-Hille, and S. Avidan, "Extended Lucas–Kanade tracking," in *Computer Vision—ECCV 2014*. Springer, 2014, pp. 142–156.

[37] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, 2013, pp. 2411–2418.

[38] R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 6, pp. 641–647, Jun. 1994.

[39] Z. Bing, Y. Wang, J. Hou, H. Lu, and H. Chen, "Research of tracking robot based on SURF features," in *Proc. Int. Conf. Nat. Comput. (ICNC)*, Yantai, China, 2010, pp. 3523–3527.

[40] J. Zhang, J. Fang, and J. Lu, “Mean-shift algorithm integrating with SURF for tracking,” in *Proc. Int. Conf. Nat. Comput.*, Shanghai, China, 2011, pp. 960–963.

[41] I. Leichter, M. Lindenbaum, and E. Rivlin, “Tracking by affine kernel transformations using color and boundary cues,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 1, pp. 164–171, Jan. 2009.

[42] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *Proc. VISAPP Int. Conf. Comput. Vis. Theory Appl.*, vol. 2. Lisbon, Portugal, 2009, pp. 331–340.

[43] Y. Liu and H. Zhang, “Indexing visual features: Real-time loop closure detection using a tree structure,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, St. Paul, MN, USA, 2012, pp. 3613–3618.

[44] P. Sturm, “Pinhole camera model,” in *Computer Vision*. Springer, 2014, pp. 610–613.

[45] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for an autonomous mobile robot,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Cincinnati, OH, USA, 1990, pp. 384–389.

[46] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, vol. 3. Hoboken, NJ, USA: Wiley, 2006.

[47] G. Oriolo, A. De Luca, and M. Vendittelli, “WMR control via dynamic feedback linearization: Design, implementation, and experimental validation,” *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 6, pp. 835–852, Nov. 2002.

[48] A. Ess, B. Leibe, and L. V. Gool, “Depth and appearance for mobile scene analysis,” in *Proc. Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, 2007, pp. 1–8.

[49] M. Gupta. (Jul. 2014). *Demonstration Video of SURF-Based Human Tracking Algorithm*. [Online]. Available: https://www.youtube.com/watch?v=wUxhAQeWXGg&feature=youtu.be

[50] M. Gupta. (Sep. 2015). *SURF-Based Human Tracking From a Mobile Robot Platform*. [Online]. Available: https://www.youtube.com/watch?v=J2I2E38cB-g

[51] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[52] S. Garg and S. Kumar, “Mean-shift based object tracking algorithm using SURF features,” in *Proc. Int. Conf. Signal Process. Robot. Autom.*, 2013, pp. 187–194.

[53] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, “An experimental comparison of online object-tracking algorithms,” in *Proc. SPIE Opt. Eng. Appl.*, San Diego, CA, USA, 2011, Art. no. 81381A.

[54] A. Mobilerobots. (1995). *Mobile Robots*. [Online]. Available: http://www.mobilerobots.com/Mobile_Robots.asp

Meenakshi Gupta received the B.E. degree in electronics and instrumentation engineering from the University of Rajasthan, Jaipur, India, in 2004, and the M.Tech. degree in mobile robotics and the Ph.D. degree in computer vision and mobile robotics from the Indian Institute of Technology Kanpur, Kanpur, India, in 2006 and 2014, respectively.

Her current research interests include computer vision with applications in machine vision and mobile robotics.

Swagat Kumar (S’08–M’13) received the master’s and Ph.D. degree in electrical engineering from the Indian Institute of Technology Kanpur, Kanpur, India, in 2004 and 2009, respectively.

He was a Post-Doctoral Fellow with Kyushu University, Fukuoka, Japan, for one year. Then he was an Assistant Professor with the Indian Institute of Technology Jodhpur, Jodhpur, India, for about two years. In 2012, he joined the R&D division of Tata Consultancy Services at New Delhi, India where he currently heads the robotics research group. His current research interests include machine learning, robotics, and computer vision.

Laxmidhar Behera (S’92–M’03–SM’03) received the B.Sc. and M.Sc. degrees in engineering from NIT Rourkela, Rourkela, India, in 1988 and 1990, respectively, the Ph.D. degree from the Indian Institute of Technology at Delhi, New Delhi, India, and the Post-Doctoral degree from the German National Research Center for Information Technology, Sank Augustin, Germany, in 2001.

He was an Assistant Professor with the BITS Pilani, Pilani, India, from 1995 to 1999. He is currently a Professor with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur, India. His research interests include intelligent control, robotics, neural networks, and cognitive modeling.

Venkatesh K. Subramanian received the B.E. degree in electronics from Bangalore University, Bengaluru, India, in 1987, and the M.Tech. and Ph.D. degrees from the Indian Institute of Technology Kanpur, India, in 1989 and 1995, respectively.

He was an Assistant Professor with the Electronics and Communication Department, Indian Institute of Technology Guwahati, Guwahati, India, from 1995 to 1999. Since 1999, he has been with the Faculty of the Electrical Engineering Department, Indian Institute of Technology Kanpur. His current research interests include generalizations of system theory, image/video processing, computer vision with applications in machine vision, visual robot navigation, human–computer interfaces, light fields, computational photography, and allied areas.