www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 4 Issue 11 Nov 2015, Page No. 15071-15082

Hybrid visual servoing for tracking multiple targets with a swarm of mobile robots

Gossaye Mekonnen Alemu

Indian Institute of Technology Roorkee,
Roorkee-247667, India
gossayemekonnen@gmail.com

Abstract: A new hybrid visual servoing based tracking of multiple targets using a swarm of mobile robots is proposed. This distributed algorithm has position based visual servoing (PBVS) and image based visual servoing (IBVS). In addition, the proposed method consists of two approaches: interaction locally among robots and target tracking. Furthermore, neural network extended Kalman filter (NEKF) is used for reducing noises which is existed during tracking targets. When the targets are slower than the robots, Lyapunov function can be used for showing that the robots asymptotically converge to each vertex of the desired configurations meanwhile tracking the targets. Towards the algorithm practical execution, it is necessary to identify the observation ability of each robot in an efficient and inexpensive way. Infrared proximity sensors and monocular camera are applied to fulfill these requirements. Our simulation results describe the proposed algorithm confirms that the considered distributed tracking multi-targets method applying robots swarm is effective and straightforward to implement.

Keywords: A swarm of robots, Local interactions, Triangle pattern, Target tracking, infrared sensor.

1. Introduction

The problem of object tracking using a single robot has been used in many real applications, such as surveillance or exploration with the necessary high-degree of capabilities. Recently, a lot of interests have been shown to swarms of robots with low computational and sensing abilities. This can bring many benefits over tracking with a single robot for instance in terms of effectiveness, error tolerance, adaptableness, and so on [1, 2]. Developing features that robots swarms can exhibit, new applications have come out, such as odor source localization or tracing of toxic plume [3, 4, 5, 6], and have extended to support tracking of multiple targets [7, 8, 9, 10, 11].
By combining controller method for local tracking and a high level behavioral based structure in a topological map of the environment is specified, the distributed robots navigate in the regions based on target density was proposed in [8]. Generally, these works are termed as cooperative target tracking. Additional associated works were primarily dedicated to implementing decentralized tracking approaches of mobile sensor networks or robot swarms for multiple xed targets were discussed in [8, 10, 11]. A problem of multiple odor sources localization by means of swarms of mobile robots was addressed in [8]. A decentralized approach for multiple targets tracking was implemented with a mobile sensor network according to the triangulation principle given in [10].
Local configurations accomplished by the above mentioned local interactions may result in a net type. These configurations present numerous redundant connections making certain utmost reliability and flexibility from the topology point of view. Based on the extent of the robots interaction each other, the network topologies should be grouped into entirely or partially connected [18]. The entirely connected topologies have each robot interact with all of other robots at the same time within a definite range. Therefore, it has too taut constraints on the motion robot, and more complexity of computations will be developed. Particularly, deadlocks may occur where some of the robots have become trapped in narrow place. These conditions occur in the majority of the prior works [12, 15, 16, 17].
This paper contribution is developing a new distributed algorithm that makes possible robots a swarm for tracking multiple moving targets and/or capturing them with limited sensing ability while achieving dynamic configuration. The multi target tracking is coordinated problem for autonomous a swarm mobile robots, while the targets are visible for a limited number of mobile robots. To solve this problem, we use the combination of IBVS and PBVS to improve the performance of the method. A neural network extended kalman filter (NEKF) is used for reducing noises existed during the motion of the object [19, 20]. The IBVS is implemented in a two-step process. We consider the universal law of gravitation [13] for the relative degree of attraction among individual targets and interactions among locally existed swarms of robots [14]. Specially, the proposed targets tracking approach is achieved without using any identifiers, leader, explicit communication or common coordinate frame. To allow robots swarm in order to attain the objective of a collaborative task, the individual robots motion should be coordinated, preferable to have a decentralized form.
The rest of the paper is arranged as follows: The computational model and system description is presented in section 2. In section 3 the algorithm of observation function is discussed. In section 4 the tracking function is presented. In section 5 the hybrid visual servoing algorithm for tracking moving multiple targets is presented. In section 6 presents the SLAM algorithm using NEKF. Simulation results are presented in Section 7. The conclusion of this paper is presented in section 8.

2. Problem Formulation

2.1. Model overview and definition

We consider a swarms of n mobile robots as $\{m_1, \ldots, m_n\}$. Consider that an initial allocation of each robot is random and have distinct positions [21]. All robots move autonomously on a plane of 2D and they do not have identifiers and leader. It also assumed, they do not occupy the same coordinate system, and do not maintain any past actions memory that gives the property of self-stabilization inherently [22, 23]. A limited range observation leads each robot to identify other robots positions within its sight range only. Furthermore, there is no explicit communication among robots. Let the position of a mobile robot m_i is $\mathbf{r}_i(t)$ at time t be a state vector denoted as $\mathbf{r}_i(t) = [r_{ix}, r_{iy}]^T$. Let us define kinematics of m'$_i$s by $\dot{r}_{ix} = u_i \cos\theta_i$, $\dot{r}_{iy} = u_i \sin\theta_i$, where $\mathbf{u}_i$ is the translational velocity and θ_i is the angular velocity of m_i. Using the model illustrated above, all the robots execute the same algorithm and executed for individual robot, and act asynchronously and autonomously to each other.

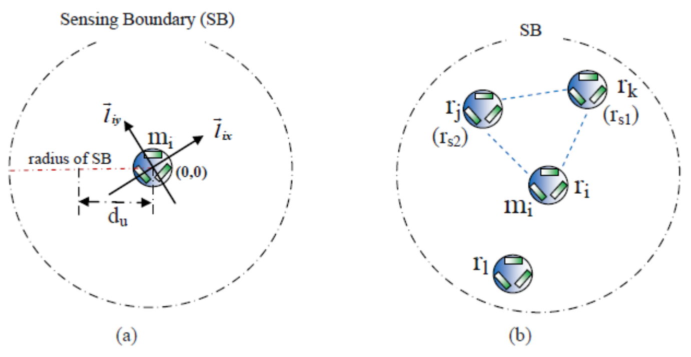


Figure 1: Representations and definitions. (a) local coordinates (r'$_i$s) and sensing boundary (SB), (b) set of observations O_i , set of neighbors N_i, and triangular configuration T_i.

We assume the axes of local coordinates frame of a robot m_i is denoted by $\vec{l}_{ix}$ and $\vec{l}_{iy}$ as shown in figure 1a, $\vec{l}_{ix}$ is the vertical axis of m_i's local coordinate and its moving direction, and $\vec{l}_{iy}$ is the horizontal axis, after rotating counterclockwise by 90^o from the vertical axis. The origin position of mi is when $\mathbf{r}_i(t) = (0, 0)$, we use $\mathbf{r}_i$ for simplicity afterwards. The distance from the robot m_is at position $\mathbf{r}_i$ to the robot m_js at position $\mathbf{r}_j$ is defined as $dist(r_i, r_j)$. We denote du as a desired distance from m_i and m_j. Subsequently, m_i observes inside its sensing boundary(SB) located other robots. As shown in figure 1b, it estimates the observed robots distance up to their central positions, results in a set which contains the positions $\mathbf{O}_i(= r_j, r_k)$ relative to local coordinates of itself. Therefore, m_i can choose two robots m_{s1} and m_{s2} inside its SB. They are the neighbors of mi and the set of their respective positions is denoted by $N_i(= r_{s1}, r_{s2})$. If r_i and N_i are given, the three individual positions in a set $\{r_i, r_{s1}, r_{s2}\}$ is known as *Triangular Configuration*, symbolized as T_i, where the angle $< r_{s1}r_ir_{s2}$ is symbolized as α_i. Following the definition for *the Equilateral Configuration*, denote it by E$_i$, as all the distance configuration change of T_i are equivalent to d_u. We require a measure signifying to what extent T_i configuration is the same as E_i. If T_i is given , the distance changes (D_i) relative to r_i can be expressed as follows

$$D_i = \begin{cases} (dist(r_m, r_n) - d_u)^2 & \text{if } m \neq n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Where $\{\{r_m, r_n\} \mid r_m, r_n \in T_i = \{r_i, r_{s1}, r_{s2}\}\}$. For simplicity $\mathbf{D}_i$ is denoted as $(\mathbf{d}_k - \mathbf{d}_u)^2$. By means of T_i and E_i , the *Local Interaction* formally can be defined as follows: for the given T_i, the local interaction permits r_i of m_i to persist d_u with N_i at every time in the direction of forming E_i. A number of extra assumptions can be taken for robot model construction: (1) the moving targets are slower than m_i, (2) if the target is inside its SB, the estimation of distance and bearing of the target separately is possible from m_i, and (3) at each time m$_i$ is allocated for a single target. Then we can properly explain the Targets Tracking problem depend on the local interaction as expressed below:

For the given mobile robots m_1 ,..., m_n situated at randomly separate positions and targets under moving, how to allow robots to follow the targets through their positions shaped into E_i.

Figure 2 illustrates the overall system control architecture. The controller inputs incorporate the measurement data acquired by the infrared sensors and monocular cameras for the predefined distance d_u among neighboring robots.

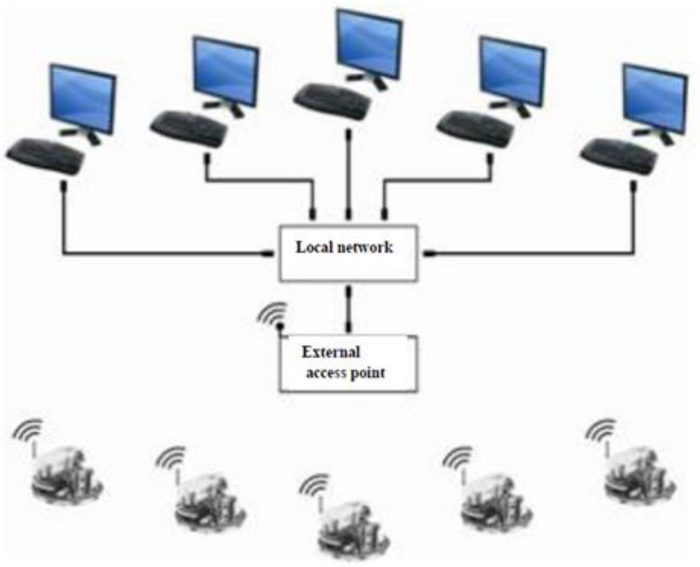


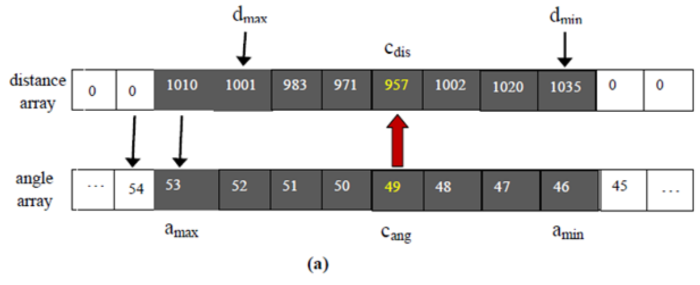
Figure 2: Mobile robots system integration overview.

3. The Observation function

3.1. Algorithm of observation function

Observation function [21] offers consistent estimation of the neighboring robots surface, which can be acquired with the steps listed below. Two one dimensional arrays in the memory of each robot are constructed by the measurement step as shown in figure 3a. Now, each array size can be automatically changed based on the angular interval of servo motor. When m$_i$ scans the surroundings at regular intervals using its infrared sensors and monocular camera, the distance to the surface of the neighboring robots is recorded in the first array of the corresponding cell. Meanwhile, angle of the servo motor is stored in the second array in such a way that the distance array keep up a correspondence to the angle array of the motor. Subsequently, m_i ensures cells of their distance array that hold a value of non-zero (from d_{min} lower bound limit to d_{max} upper bound limit) and reads cells of the corresponding array of angle. The measurement data is corrected in the update step according to a reference value. At the same time as data is recording in the arrays of distance and angle, simultaneously the estimated distance is recorded in the subsequent cell of the array as an integer value of intensity. After scanning 360 degree, the algorithm Sobel edge detection [24] enhances the surface detection data to the original value.

The robots positions are identified in the recognition step. m_i selects the non-zero value cells starting from d_{min} to d_{max} in the array of updated distance. Next, r_{min}, r_{max}, and r_s are the three feature points specified using d_{min}, d_{max}, c_{dis}, and in the angle array there is their subsequent angle value cells, respectively.



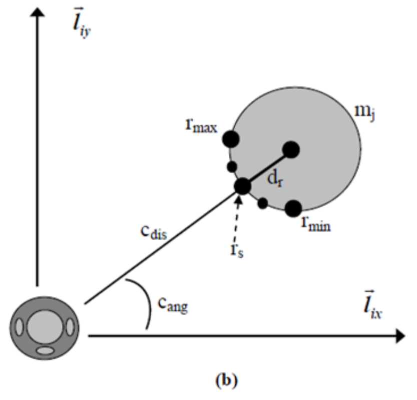


Figure 3: A mobile robot m_j surface scanning observation. (a) arrays of distances and angles for m_i ; (b) m_i observes m_j , the neighbor mobile robot.

Through the average computation of a series of numeric values in the angle array of the motor, m_i choose the cell that stores the value equal or nearest to the average, and sets c_{ang} central angle for this value. The central distance cdis is the distance cell corresponds to c_{ang}. As illustrated in figure 3b, computation of r_s is done using c_{dis} with the shortest distance value among the cells and c_{ang}. Then, r_i is computed by $dist(r_{min}, r_{max})$ and varifies whether $dist(r_{min}, r_{max})$ is smaller than the controller reference diameter.

If this distance greater than the controller reference diameter, the data stored in the cells can be taken as the border of an arena. If not, these cells data are taken as a robot. During the process mentioned above, if mobile robots are sensed the central point r_j of them can be computed by adding c_{dis} to the radius dr as shown in figure 3b. Therefore, the observation function outputs are O_i of the nearest robots.

4. Tracking function

4.1. Local interaction

The role of infrared in this section is to detect the nearby mobile robots and by avoiding collision leads to motion. Now, we give details of the local interaction method [21] among mobile robots that capable to generate E_i of side length d_u from an arbitrary T_i using three neighboring mobile robots. This method consists of the interaction function ϕ with parameters r_i and N_i at every time. Assume m_i and its two neighbors mobile robots are m_{s1} and m_{s2} are placed inside its SB. As illustrated in figure 4a, these three robots have configuration of the form T_i with vertices are r_i , r_{s1}, and r_{s2}, respectively. Primarily, m_i computes the centroid of the triangle $<r_i r_{s1} r_{s2}$, indicated by r_{tc}, relative to its local frame, and it measures the angle ϕ which is formed by the line linking the 2 neighbor mobile robots $\overline{r_{s1}r_{s2}}$ and, $\vec{l}_{i,y}$. By means of r_{tc} and ϕ, m_i computes the point r_{it} for next movement by its present observation of neighbor mobile robots.

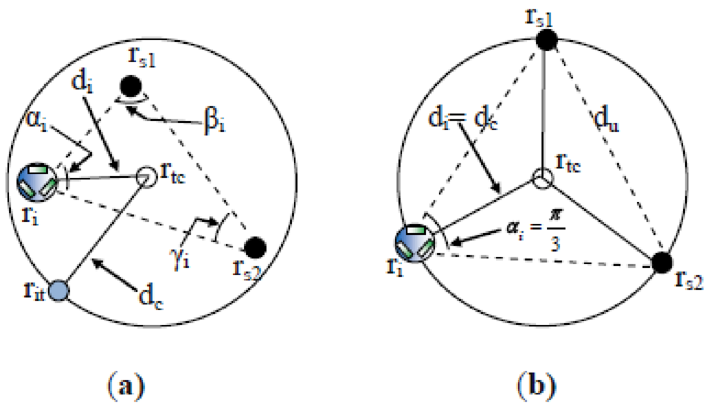


Figure 4: Illustrations of Local interaction. (a) the two parameters of control: range d_i and bearing α_i ; (b) desired configuration in equilateral triangular form.

Usually, using this method, m_i can retain a distance d_u among its two neighboring mobile robots at every time. Specifically, each mobile robot tries to structure of an isosceles triangle through N_i at every time, and by repetitively doing this, three mobile robots configuration has the form of E_i by themselves. As shown in figure 4b, we supposed that the circumscribed circle of an equilateral triangle with its centroid is r_{tc} of $<r_i r_{s1} r_{s2}$ and radius d_c is $d_u/\sqrt{3}$. The position of each robot determined by the local interaction through computing the distance d_i from r_{tc} and α_i as shown in figure 4a.

4.2. Target tracking

The target tracking method provides an answer to how to locate m_is moving direction on the way to a desired target, and at the same time how to put together the neighbor robots positions to form E_i based on the direction of desired target.

Here also we used both infrared sensors and monocular camera for simultaneously detecting the distance among the neighbors robots and searching the target object for tracking those targets in the SB. Under this method, it is considered that mi is faster than the targets which are moving. While recognizing multiple mobile targets, defined as $\{g_k|1\leq k\leq n\}$, in figure 5a, m_i chooses its tracks in the direction of a target g_k. According to the law of gravitation, m_i selects its path by means of the degree of attraction relative to the targets, defined the preferred vector f_k, its magnitude is known by $||f_k|| = ||1/d^2_k||$ where d_k is the distance from g_k to m_i.

Therefore, the identified targets set G_i is represented by the favorite vectors set $\mathbf{f}_k|1 \leq k \leq n$. Next m_i chooses the highest magnitude of $\mathbf{f}_k$, with the value of $||\boldsymbol{f}_k||_{max}$. For selected $\boldsymbol{f}_k$, m_i is allocated to a single target inside its SB at every time. As illustrated in figure 5b, m_i identifies a favorite maximum area of target $\mathbf{A}(\mathbf{f}_{\max})$ inside its SB unified with the plane upper half part in the direction of $||\mathbf{f}_k||_{\max}$. m_i confirms neighbors exist or not in any $\mathbf{A}(\mathbf{f}_{\max})$. If neighbors are detected, m_i chooses the first neighbor m_{s1} situated the minimum distance away from r_i to describe r_{s1}. If not, m_i selects a virtual point r_v placed at certain distance d_v far from r_i along $||\mathrm{f}_k||_{\max}$ to define the value of r_{s1}. While the required target is not observed, m_i searches inside its SB its first neighbor. As shown in figure 5c, m_i identifies its $\mathbf{h}$ heading relative to its local frame. Consider $\mathbf{A}(\mathbf{h})$ represents the heading direction area inside SB and has intersection with plane from the upper half through $\mathbf{h}$. m_i ensures whether there present any neighbors within $\mathbf{A}(\mathbf{h})$. If the existence of neighbors within $\mathbf{A}(\mathbf{h})$ are detected, m_i chooses m_{s1} with the smallest distance far from m_i. If not, m_i searches m_{s1} inside SB as method described above. The next neighbor m_{s2} is chosen by the entire distance from r_{s1} to r_i along r_{s2} to be minimum. Therefore, by means of r_i and N_i, r_{ti} can be computed by ϕ using local interaction method.

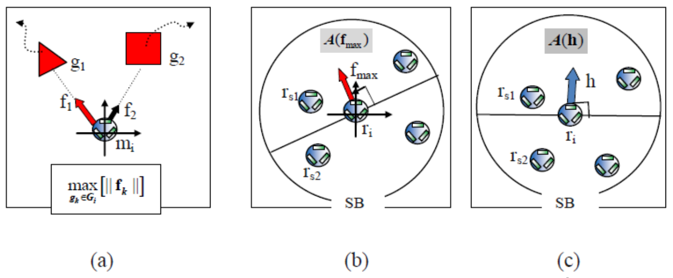


Figure 5: Trajectory tracking method illustration. (a) favorite vectors computation ; (b) neighbor selection computation inside $\mathbf{A}(\mathbf{f}_{\max})$; (c) neighbor selection computation inside $\mathbf{A}(\mathbf{h})$.

5. Hybrid visual servoing algorithm

In case of moving multiple targets, the existed algorithms should be customized to guarantee the stability and error-free tracking. This can be done by addition of moving target velocity in the control algorithm. Consider both PBVS for determining the pose of the targets in Cartesian coordinate and the IBVS controller of a point feature related to a 3D point P (expansion to numerous point features). The point feature is denoted by $\mathbf{u} = (u_x, u_y)^T$ in the image plane. These coordinates expresses the centered pixel coordinates with respect to the principal point of the camera. If the point P is fixed, the rate of change of $\mathbf{u}$ depend on the cameras linear velocity (v_l) and angular velocity (ω_l) [25].

In order to have vision based control method, we use only image data for solving this estimator. Therefore, the control law to be implemented is modified as, detail explanation is given in [28],

$$\begin{bmatrix} v_l \\ \omega_l \end{bmatrix} = \boldsymbol{J}_{\boldsymbol{u}}^{\dagger}(\boldsymbol{u}, \hat{\boldsymbol{Z}})\boldsymbol{K}_{\boldsymbol{g}}\boldsymbol{e} + \begin{bmatrix} \hat{v}_p \\ 0 \end{bmatrix}, \qquad (2)$$

where $\boldsymbol{J}_{\boldsymbol{u}}^{\dagger}$ is the pseudo inverse of $\mathbf{J}_u$, $\hat{\boldsymbol{z}}$ is the estimated depth obtained from the depth observer, $\boldsymbol{K}_{\boldsymbol{g}}$ is a positive definite gain matrix, and $\boldsymbol{e} = \boldsymbol{u}_a - \boldsymbol{u}_d$ is error of the image feature with respect to the desired point and $\hat{\boldsymbol{v}}_{\boldsymbol{p}}$ is the estimated object feature velocity produced by the estimator of the target velocity. The figurative description of this method is shown in [28]. The proposed hybrid visual servoing algorithm for a swarm of mobile robots tracking multiple targets is described in Algorithm 1.

Algorithm 1: Details of online Hybrid Visual Servoing algorithm for tracking moving targets with a swarm of mobile robots.

```
Input: A list of input data:
-The measurements of image features $M_f(t)$, where $M_f = \{M_{fi} : i \in \{1,..., m\}\}$, and
-The pose of image features X at time t.
Output: A Hybrid algorithm $U = H(x)$, that contains outputs $(v, \omega)$ of the two IBVS and PBVS.
  • for $i=1$ to $n$ do
  • for till the convergence do
  • if $j \in$ ( IBVS, PBVS ) then
  • if $X(actual) \neq X(desired)$ then
  • for $j \in$ IBVS do
  • -Acquire images intensity I from the corresponding
      o features in the IBVS algorithm, $U_i = h_i(m_f)$.
  • -Compute optical flow $(v_l, \omega_l)$ as in equation (2).
  • -The robot translate and rotate from X(actual)
      o towards X(desired) by applying both the local
      o interaction and tracking methods simultaneously.
  • return;
  • if $j \in$ PBVS then
  • for $X(actual)(t) \neq X(desired)$ do
  • -Compute the pose using PBVS algorithm.
  • return X(actual)
  • -Compute the outputs $(v_l, \omega_l)$ of the hybrid algorithm, $U_j$
  • $= H_j(x)$.
  • if IBVS or PBVS then
  • -Compute linearization through switching between the
  • output signals of the algorithms IBVS and PBVS by
  • applying NEKF controller.
  • return;
  • return;
```

6. SLAM algorithm using Neural Network Extended Kalman filter

The hybrid filter of an artificial neural network (ANN) performs as an observer for learning the uncertainty of the system on-line along with an EKF. An adaptive state estimation method using an EKF and a neural network is implemented for reducing noise. The omnidirectional wheeled mobile robot with values of encoder (v_l, ω_l) learns values (x'_t, y'_t, θ'_t) which are information values driven from working environment (x_t, y_t, θ_t) using algorithms of multilayer perceptron (MLP). Figure 6 gives a schematic diagram of this process, where μ_t is the mean value of the feedback and Σ_t is the covariance [19, 20, 26, 27].

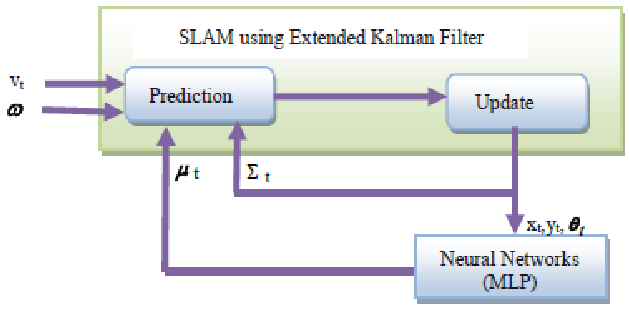


Figure 6: Architecture of Hybrid filter for SLAM.

6.1. Prediction step

Prediction step is the first phase of NEKF, the standard state propagation equations are used for propagating the joint state vector and the state. The equations are:

$$\hat{x}_{t|t-1} = \begin{bmatrix} \hat{x}_{u(t|t-1)} \\ \hat{x}_{\omega(t|t-1)} \\ \hat{x}_{f(t|t-1)} \end{bmatrix} = \begin{bmatrix} f_{true}(.)+x_u Net \\ \hat{x}_{\omega(t|t-1)} \\ \hat{x}_{f(t|t-1)} \end{bmatrix} \qquad (3)$$

where, $\mathbf{f}_{true}(.)$ is theoretical motion model, $x_{uNet} = NN(\hat{x}_{u(t-1|t-1)}, \hat{x}_{\omega(t-1|t-1)}, u_{t|t-1})$ is the output of the neural network, $\hat{x}_{u(t-1|t-1)}$ is the robot state estimate at $t-1$, and $u_{t|t-1}$ is the control vector of the robot motion that initiates its motion from state $t-1$ to state t. The network input is formed from the previous robot state vector $\hat{x}_{u(t-1|t-1)}$ and controls $u_{t|t-1}$:

$$x_{Net} = [(\hat{x}_{u(t-1|t-1)})^T \quad (\hat{u}_{t|t-1})^T] \qquad (4)$$

6.2. Update step

The update step of NEKF implemented by applying the following equations in the same way as in the standard EKF:

$$\begin{aligned} S_t &= H_t \Sigma_{t|t-1} H_t^T + R_t \\ K_t &= \Sigma_{t|t-1} H_t^T S_t^{-1} \\ \hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t (z_t - \hat{z}) \\ \Sigma_{t|t} &= (I - K_t H_t) \Sigma_{t|t-1} \end{aligned} \qquad (5)$$

Where, S_t is the state vector innovation covariance, $\hat{x}_{t|t-1}$ is the predicted joint state vector, $\Sigma_{t|t-1}$ is the covariance of the predicted joint state vector, H_t is measurement model $h(.)$ Jacobian, R_t is the covariance matrix of the measurement model, and K_t is the gain in the NKEF filter. z_t is the measurement of positions of the observed features in the image Plane in visual SLAM (VSLAM) and $\hat{z}_t$ is the predicted positions vector of features, $\hat{x}_{t|t}$ the joint state vector updated values, and $\Sigma_{t|t}$ the covariance of the updated values of the joint state vector. The robot can distinguish its pose using these recursive estimation methods.

7. Simulation platform

The simulation platform used in this paper is explained in this section. The objective of the simulation at this level is for tracking multiple of objects which are learned offline. A swarms of mobile robots are moving towards the moving objects by using the features of the moving object under the condition of local interaction method. This is done by considering the omnidirectional wheeled mobile robots that acquires image features from the monocular cameras for vision based controller and state estimation. The simulation platform is based on robotino mobile robots. Figure 7 shows a swarm of robotino mobile robots setup that have USB wireless access points and web cameras. Table 1 shows the specifications omnidirectional mobile robots. The proposed a swarm of mobile robots tracking multiple moving objects algorithm carried out by the mobile robots velocities which are obtained from the features velocities.

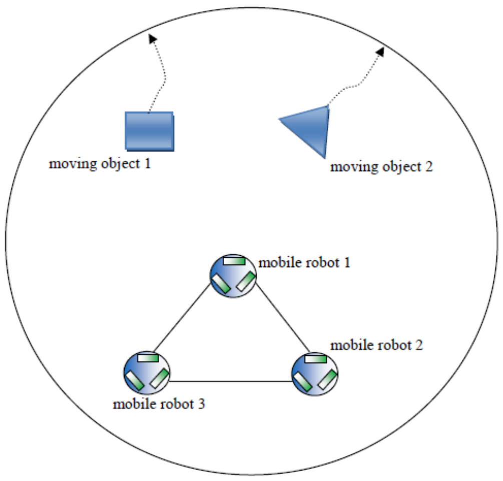


Figure 7: a swarm of mobile robots tracking multiple moving objects platform.

8. Simulation Results

Multi-targets tracking simulations using a swarm of mobile robots are carried out in Matlab-Simulink based on optical flow visual controller, neural extended kalman filter, PID controller, kinematics and dynamics models of the mobile robots as shown in simulink model Figure 8. The parameter values used for simulation are given in Table 1. The mobile robot trajectory is a straight line with inputs $x = V_{xd} * t$ and $y = V_{yd} * t$. Here V_{xd} and V_{yd} are constant velocity inputs along x and y direction respectively, and t is time.

Table 1: Omnidirctional wheeled mobile robot parameters used in simulations.

Parameters	Symbol	Value	Units
Robot mass	m	11	Kg
Common radius of wheels	r	0.04	m
Initial distance between actual and desired pose	ρ	4	m
Robot moment of inertia	I_Q	0.0176	Kgm^2
Proportional gain	K_p	500	
Derivative gain	K_d	100	
Integral gain	K_i	900	
Input velocity along x from trajectory generator	V_{xd}	25	m/s
Input velocity along y from trajectory generator	V_{yd}	25	m/s
Desired angular velocity	ω_d	0	rad/s

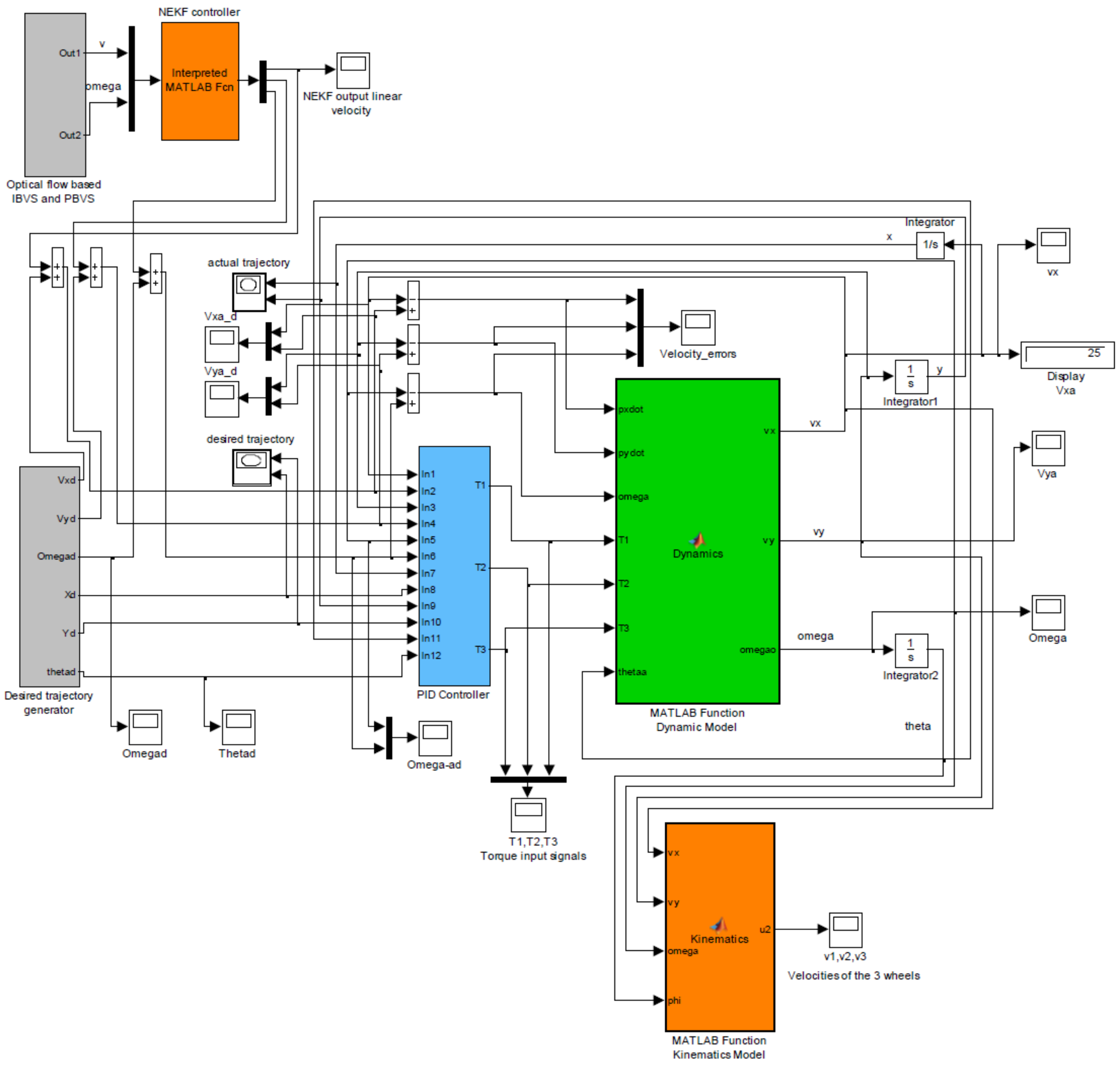

NEKF controller
Interpreted MATLAB Fcn
NEKF output linear velocity
Optical flow based IBVS and PBVS
actual trajectory
desired trajectory
Velocity_errors
Integrator
Display Vxa
Integrator1
Integrator2
Desired trajectory generator
PID Controller
Dynamics
MATLAB Function Dynamic Model
Omegad
Thetad
Omega-ad
T1,T2,T3 Torque input signals
Kinematics
MATLAB Function Kinematics Model
v1,v2,v3 Velocities of the 3 wheels

Figure 8: The simulink model for the mobile robot with controllers.

Figure 9 is the simulation result performed on swarms of mobile robots, where distance interval between individual robot and their Ni are shown with respect to time. This figure illustrates the mobile robot moves as the required given path. Figure 10 shows the three torque inputs for wheels of the mobile robot and will get constant value after few seconds of motion which leads to the desired pose near to the object: (a) for robot 1, (b) for robot 2, (c) for robot 3. Figures 11 and 12 show the desired and actual velocities of the linear velocity components inputs along the x and y axis: (a) for robot 1, (b) for robot 2, (c) for robot 3, respectively. It can be seen from the figures that the motion of the robot is becomes stable after a small time interval. Figure 13 shows the velocities of left, right and middle wheels: (a) for robot 1, (b) for robot 2, (c) for robot 3. The plots show the velocities of the wheels along the straight

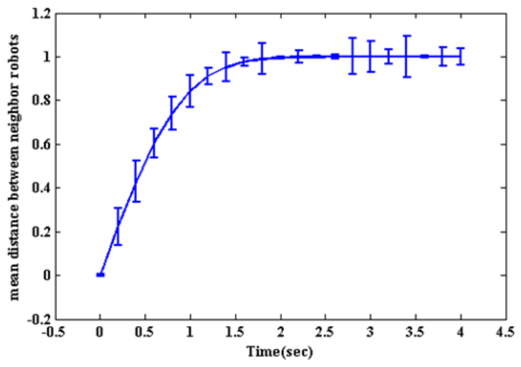
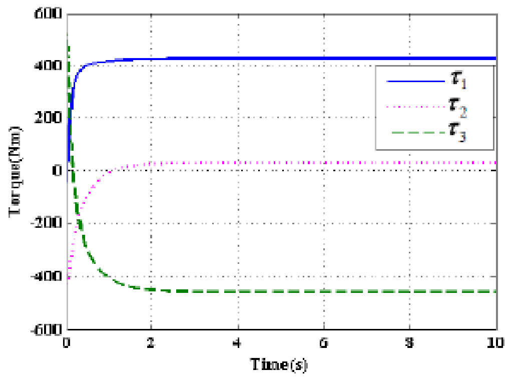
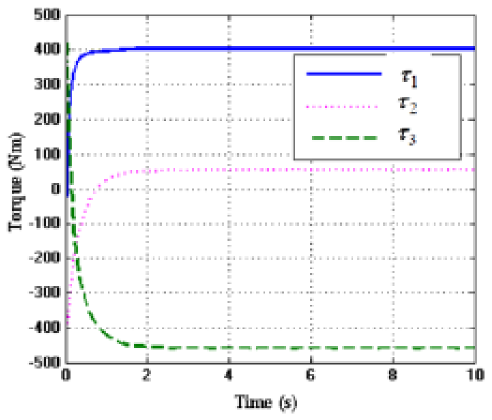


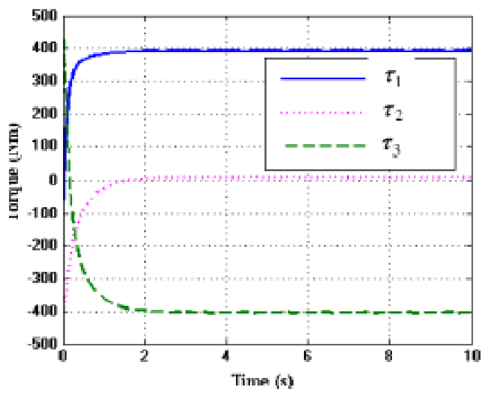
Figure 9: Variations of distance between r_i and r_{s1}.



(a)

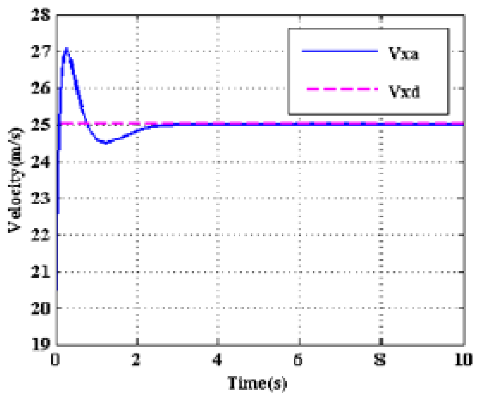


(b)

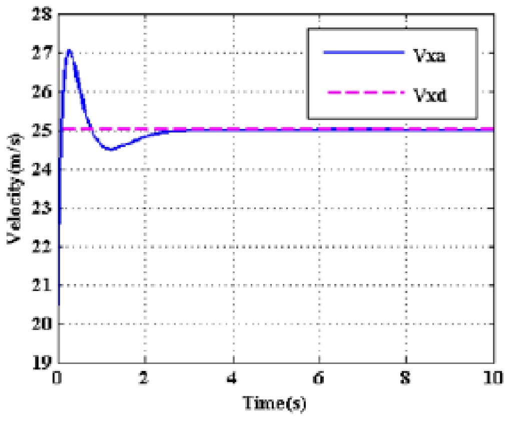


(c)

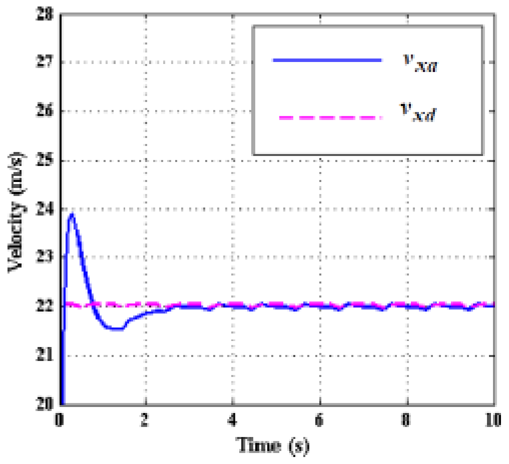
Figure 10: Torque inputs to the dynamics of the mobile robot: (a) for robot 1, (b) for robot 2, (c) for robot 3.



(a)

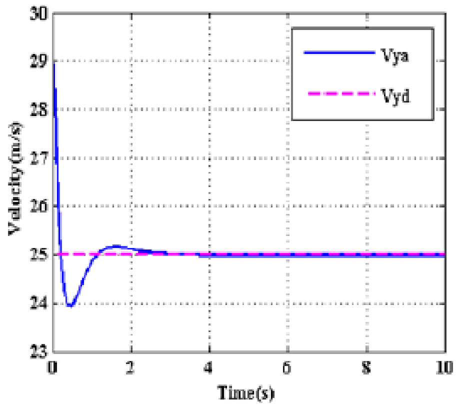


(b)

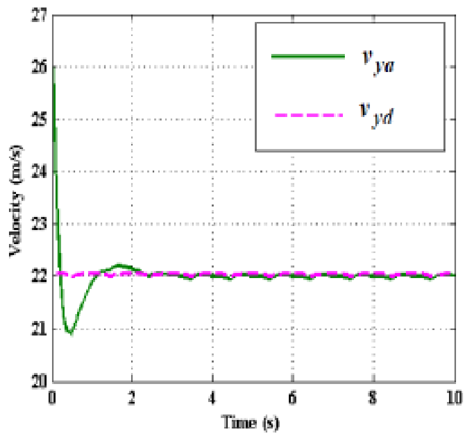


(c)

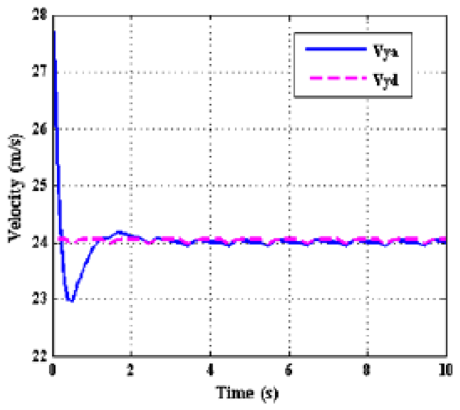
Figure 11: The actual and desired component velocity inputs of wheels along x: (a) for robot 1, (b) for robot 2, (c) for robot 3.



(a)

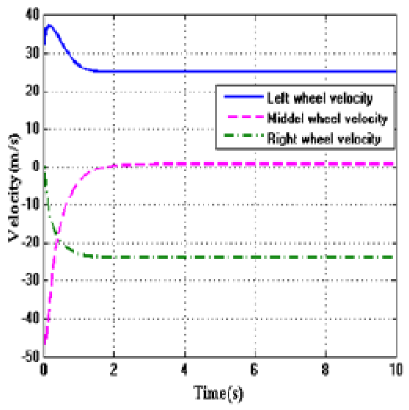


(b)

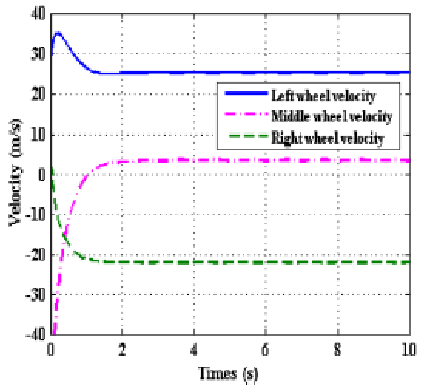


(c)

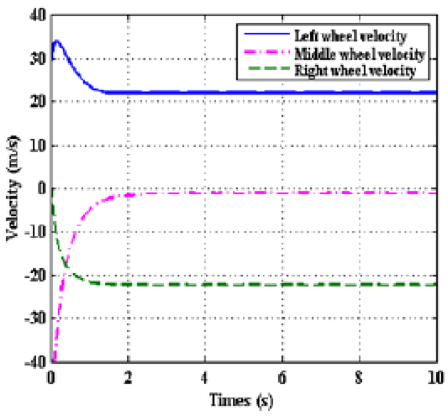
Figure 12: The actual and desired component velocity inputs of wheels along y: (a) for robot 1, (b) for robot 2, (c) for robot 3.



(a)

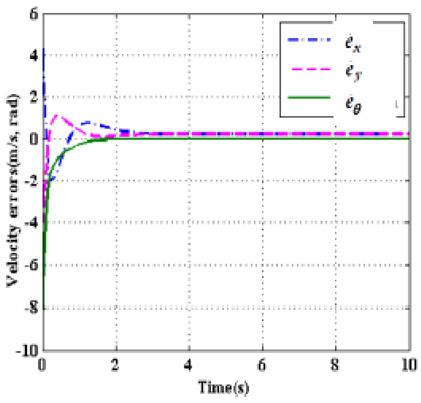


(b)

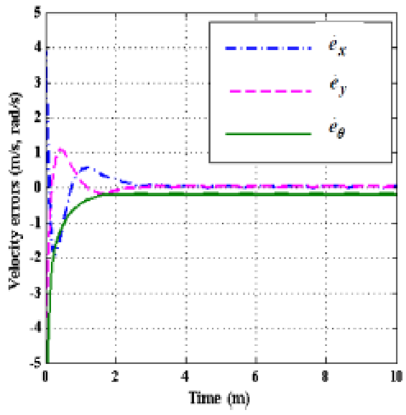


(c)

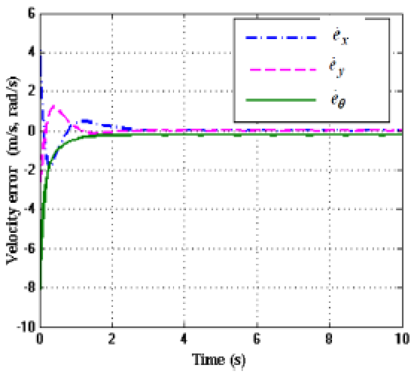
Figure 13: The velocities of the 3 wheels: (a) for robot 1, (b) for robot 2, (c) for robot 3.



(a)



(b)



(c)

Figure 14: The inputs velocity errors: (a) for robot 1, (b) for robot 2, (c) for robot 3 ($\dot{e}_x = v_{xa} - v_{xd}, \dot{e} = v_{ya} - v_{yd}, \dot{e}_\theta = \omega_a - \omega_d$).

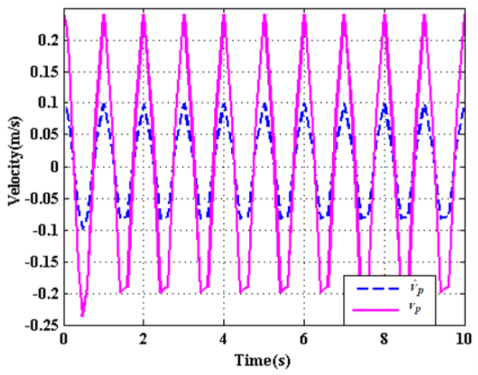
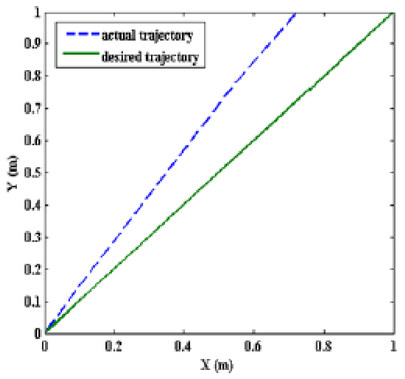
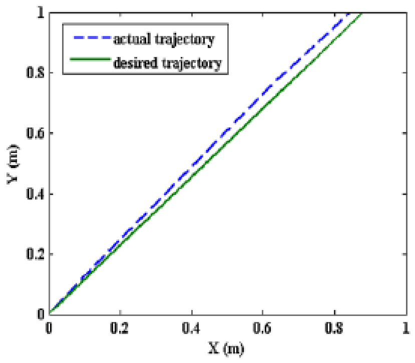


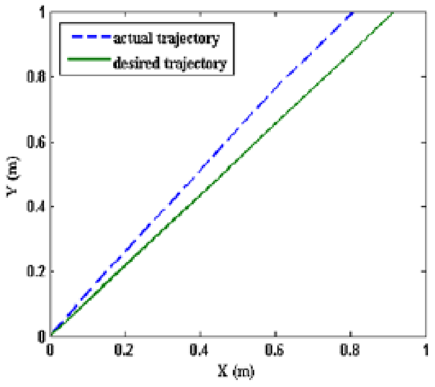
Figure 15: Variation of point feature velocity v_p and the estimated point feature velocity $\hat{v}_p$.



(a)



(b)



(c)

Figure 16: Show the desired and actual straight line trajectory of the mobile robot: (a) for robot 1, (b) for robot 2, (c) for robot 3.

line after making rotational motion in order to search the object. For making translational motion the signs of the left and the right wheels motor functions should be opposite, the reason behind it is that the three wheels of the omnidirectional mobile robots are 120 degrees separated from each other. and target. In [29] presented a robust image interest point detector and descriptor which is called SURF.

10. Conclusion

A real-time tracking method is presented in this chapter, allowing swarms of mobile robots to track multiple targets that are moving as having meshes structure of regular triangle during local interactions. The presented algorithm is distributed and free from deadlock, also it does not require a leader, beacon, universal coordinate system, previous states memory, or explicitly given communication means. By utilizing infrared sensors and monocular camera, every mobile robot could acquire information for relative positioning along with the neighboring robots surface geometry. In general it can be summed up as follows: (1) the proposed tracking algorithm shows the globally asymptotically convergence properties and demonstrated by the entire simulations. (2) Infrared sensors and camera were used. Their features consist of high reliability, low cost and simple to integrate into mobile robots. (3) The proposed algorithm can be successfully functional to sensor networks of mobile robots for surveillance operations or holding toxic materials.

References

[1] H. Choset, “Coverage for robotics-a survey of recent results,” Ann Math Artif Intell, vol. 31, no. 1-5 (4), pp. 113-126, 2001.

[2] E. Sahin, “Swarm robotics: from sources of inspiration to domains of application,” In Proceedings of 8th international conference on the simulation of adaptive behavior, LNCS, vol. 3342, pp. 10-20, 2005.

[3] D. Zarzhitsky, D. Spears, and W. Spears, “Coverage Distributed robotics approach of to chemical plume tracingapplication,” In Proceedings of IEEE/RSJ international conference on intelligent robots and systems, pp. 4034-4039, 2005.

[4] W. Jatmiko, K. Sekiyama, and T. Fukuda, “A particle swarm-based mobile sensor network for odor source localization in a dynamic environment,” Gini M, Voyles R (eds) Distributed autonomous robotic systems Springer, Japan, vol. 7, pp. 71-80, 2007.

[5] AT. Hayes, A. Martinoli, and RM. Goodman, “Swarm robotic odor localization: offline optimization and validation with real robots,” Robotica, vol. 21, no. 4, pp. 427-441, 2003.

[6] A. Dhariwal, GS. Sukhatme, and AAG. Requicha, “Bacterium inspired robots for environmental monitoring,” In Proceedings of IEEE international conference on robotics and automation, pp 1436-1443, 2004.

[7] JR. Spletzer, and CJ. Taylor, “Dynamic sensor planning and control for optimally tracking targets,” Int J Robot Res, vol 22, no. 1, pp. 7-20, 2003.

Figure 14 shows the velocity errors occurs during the motion of the mobile robot, which tends to zero while the robot moves towards the desired pose: (a) for robot 1, (b) for robot 2, (c) for robot 3. Figure 15 shows n moving object feature velocity and the estimated velocity which estimates the velocity of the point features. Figure 16 shows the actual and the desired trajectories of the omnidirectional wheeled mobile robots: (a) for robot 1, (b) for robot 2, (c) for robot 3. The speeded up robust feature (SURF) method is used to obtain feature matches on current

[8] B. Jung, and GS. Sukhatme, “Tracking targets using multiple robots: the effect of environment occlusion,” Autonom Robots, vol. 13, no. 3, pp. 191-205, 2002.

[9] KN. Krishnanand, P. Amruth, and MH. Guruprasad, “Glow worm inspired robot swarm for simultaneous taxis towards multiple radiation sources,” In Proceedings of IEEE international conference on robotics and automation, pp. 958-963, 2006.

[10] S. Kamath, E. Meisner, and V. Isler, “Triangulation based multi target tracking with mobile sensor networks,” In Proceedings of IEEE international conference on robotics and automation, pp. 3283-3288, 2007.

[11] X. Cui, CT. Hardin, RK. Ragade, and AS. Elmaghraby, “A swarm approach for emission sources localization,” In Proceedings of 16th IEEE international conference on tools with artifficial intelligence, pp. 424-430, 2004.

[12] W. Spears, D. Spears, J. Hamann, and R. Heil, “Distributed, physics-based control of swarms of vehicles,” Autonom Robots, vol. 17, no. 2-3, pp. 137-162, 2004.

[13] D. Halliday, R. Resnick, and J. Walker, Fundamentals of physics, 5th Ed.Wiley, New York, 1997.

[14] G. Lee, and NY. Chong, “A geometric approach to deploying robot swarms,” Ann Math, vol. 52, no. 2-4, pp. 257-280, 2009.

[15] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, “Self-organizing formation algorithm for active elements,” In Proceedings of the IEEE symposium on reliable, distributed systems, pp 416-421, 2002.

[16] N. Heo, and PK. Varshney, “A distributed self spreading algorithm for mobile wireless sensor networks,” In Proceedings of the IEEE wireless communication and networking conference, pp 1597-1602, 2003.

[17] G-L. Wang, G. Cao, and TL. Porta, “Movement-assisted sensor deployment,” In Proceedings of the IEEE infocom conference, pp 2469-2479, 2004.

[18] S. Ghosh, K. Basu, and SK. Das, “What a Mesh! An architecture for next-generation radio access networks,” IEEE Netw, vol. 19, no. 5, pp. 35-42, 2005.

[19] K. Kramer, and S. Stubberud, “Tracking of multiple target types with a single neural extended kalman filter,” Int J Intell Syst vol. 25, pp. 440-459, 2010.

[20] S. Stubberud, and K. Kramer, “Analysis of system identification using the neural extended Kalman filter,” In Proceedings of 19th International Conference on System Engineering, Las Vegas, paper 978-0-7695-3331-5, vol. 08, pp. 153-158, 2008.

[21] G. Lee, NY. Chong, and H. Christensen, “Tracking multiple moving targets with swarms of mobile robots,” Intel Serv Robotics, Springer, 3:61-71, 2010.

[22] I. Suzuki, and M. Yamashita, “Distributed anonymous mobile robots: formation of geometric patterns,” SIAM J Comput, vol. 28, no. 4, pp. 1347-1363, 1999.

[23] S. Dolev, Self-stabilization, MIT Press, Cambridge, 2000.

[24] RC. Gonzalez, RE. Woods, Digital image processing, 2nd Ed. Prentice Hall, Engle-wood Cliffs, 2002.

[25] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control, Advanced Textbooks in Control and Signal Processing, Springer, 2011.

[26] KA. Kramer, and SC. Stubberud, Analysis and implementation of a neural extended Kalman iter for target tracking, Int J Neural Syst 16(1):1-13, 2006.

[27] AR. Stubberud, "A validation of the neural extended Kalman filter," In Proceedings of 18th International Conference on System Engineering, Coventry, University, London, pp. 3-8, 2006.

[28] N. Shahriari, S. Fantasia, F. Flacco, and G. Oriolo, "Robotic Visual Servoing of Moving Targets," In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokoyo, Japan, pp. 77-82, 2013.

[29] Bay H, Ess A, Tuytelaars T, L. Van Gool, "SURF: Speeded Up Robust Features," Comput Vis Image Underst 110(3):346-359 (2008).