Robotica (2013) volume 31, pp. 267–284. © Cambridge University Press 2012
doi:10.1017/S0263574712000239

# Experiments on multi-agent capture of a stochastically moving object using modified projective path planning

Vijaysingh Shinde, Ashish Dutta and Anupam Saxena*

*Indian Institute of Technology, Kanpur 208016, India*

(Accepted April 13, 2012. First published online: May 24, 2012)

**SUMMARY**
Most of the past research in swarm robotics has considered object capture and transport using a specified and very large number of agents. The objects therein were either stationary or moving deterministically (i.e., along a known path). In most previous efforts, the obstacles were also considered stationary. Here we present a modified projective path planning algorithm and illustrate via laboratory experiments that an object exhibiting stochastic (unplanned) but low-speed motion can be restrained by a limited number of agents guided in real-time across randomly moving obstacles. Relaxation of certain restrictions in the grasping objective allows for the determination of a minimum number and placement of agents around the perimeter of any generically shaped prismatic object. A closed loop experiment is designed using a single overhead camera that provides the visual feedback and helps determine the instantaneous positions of all entities in the workspace. Control signals are sent to the robots via wireless modules by a central processing unit to navigate and guide them to their respective new positions in the subsequent time-step. Agents continue to receive signals until they restrict the moving object in form closure.

KEYWORDS: Multi-agent system; Form closure; Optimal capture; Object tracking; Online motion planning.

## 1. Introduction and Prior Work

Cooperative robotics is a research area wherein multiple robots perform a task that cannot be accomplished using a single robot. Some of these include planetary exploration, reconnaissance, rescue, and cleaning.[1] Of particular interest is the task of capture and transportation that can have applications in, say, defense and outer space systems. In defense, strategic enemy motion can be checked. Trapping of a ball in robot soccer can be performed by a number of robots. In outer space systems, a nonoperational satellite/shuttle undergoing a misbehaved/random motion can be restricted and brought to the station for repair. Space debris can be restrained and destroyed. Previous works on multi-agent systems (e.g., in refs. [1–5]) mainly focus on planar cases and use a large number of robots to perform tasks such as capture and transportation. In most of these, the object is assumed stationary. After capture, the object is transported by robots moving in formation. In few cases (e.g., in refs. [6, 16, 17]), where a limited number of robots are used, the object is either gripped rigidly or pushed to the desired goal point. Experimental studies wherein a moving object is captured optimally in form closure using the least number of robots in the presence of randomly moving obstacles are rare. Such a study is the main focus of this paper.

* Corresponding author. E-mail: anupams@iitk.ac.in

### *1.1. Object capture and transportation*

The problem of object capture is closely related to that of object grasp. The analysis of mechanical fixtures dates back to the work of Reuleaux.[7] Hanafusa[8] and Salisbury[9] introduced robotic grasping with a three-finger hand. Yoshikawa[10] proposed requisite conditions for form and force closure. By force closure it is meant that an object is in equilibrium with respect to the forces but it is still free to rotate. In form closure, both forces and moments are balanced so that all motion modes of the object are restricted. Force closure is attainable with a three-point grasp, while for form closure, a four-point grasp is mandatory. Numerous algorithms to attain form and force closures for an object and finger systems are presented in refs. [11–14]. A detailed review on object grasping was given by Bicchi and Kumar.[15] Ahmadabadi and Nakano[16] proposed an approach for four mobile robots to manipulate an object at rest. The task is divided into two subtasks, "constrain" and "move." Based on this notion, the object can be gripped rigidly and carried along a straight line and/or it can be moved about a fixed point. Sugar and Kumar[17] described a framework and control algorithm to coordinate multiple mobile robots with manipulators focusing on tasks that require grasping, manipulation, and transporting large and possibly flexible objects without special purpose fixtures. In simulation, Wang and Kumar[18] used the potential field method to obtain object closure conditions using a large group of robots. The object was then moved to a desired location by controlled robot formation or caging. Potential field method has been also applied for multi-robot manipulation, decentralized control, and abstraction (e.g., in refs. [19, 20, 21]). In some, collision between multi-agents is also permitted. Grob *et al.*[24] used the concept of autonomous self-assembly of swarm robots and performed several experiments to rigidly connect to a prey using grippers. Thereafter, the prey was transferred to a desired location.

The following works are related to object manipulation mainly by pushing. Mason[22,23] investigated the mechanics of pushing and discovered a decision procedure to determine the sense of motion of the object. Peshkin and Sanderson[24,25] extended Mason's pushing analysis by defining the relation between the pushing distance and the rotation rate. They developed a procedure to orient parts that moved along a conveyor belt through a series of fences.[26] Yamada and Saito[27] proposed to push a box by multiple robots without explicit communication for a dynamic environment. A two-point pushing operation[28] can eliminate positioning uncertainty in all degrees of freedom (DOF) in a plane. However, a single-point pushing operation is more challenging. Three-point pushing was investigated by Sudsang *et al.*[6] Their method did not require the robots to always be in contact with the object. Sudsang *et al.*[6] decomposed the construction of manipulation plans into the computation of a collision-free path for an object. This was followed by the construction of elementary robot motion sequences to execute this path. They performed manipulation planning in presence of stationary obstacles by exploiting previously available exact or approximate motion planners.

The following are the novel features and main assumptions in this paper that are presented in comparison with our previous effort in ref. [29] and the work by Sudsang *et al.*[6] In this work, prismatic objects and obstacles are all regarded to be moving stochastically (i.e., in an unplanned manner) and with low speeds within the 2D workspace. In ref. [29], the object moved along a known trajectory, while the obstacles were assumed stationary. The projective path-planning algorithm herein is modified significantly compared to that in ref. [29] to facilitate hardware implementation and experimental validation in real-time. At any time-step, an agent's reaction within the workspace is categorized into three explicit functions: (i) *free motion*, (ii) *collision avoidance*, and (iii) *capture*. These functions were implemented only implicitly in ref. [29], which verified the planning algorithm in simulation. In the free-motion function, the agent's rotation is not bounded by an upper limit unlike in ref. [29]. The collision avoidance function herein is more direct and less involved. The sector search method for collision detection in ref. [29] is reckoned to be temporally expensive and hence is discarded in this work. Also, a significantly simpler object–agent interaction, in contrast to a three-case execution in ref. [29], is modeled herein. The capture function implemented in this work is more thorough compared to the one in ref. [29], as it not only places an agent near its respective goal point but it also orients the agent such that the latter exerts a normal force on the respective edge.

In addition, there exist salient differences between the proposed approach and that by Sudsang *et al.*[6] They aimed to transport an initially stationary object across a group of immobile obstacles. In this work, however, the focus is on capturing a randomly moving object with dynamic obstacles present in the workspace. The two works are similar in terms of the minimum number of robots employed to perform respective tasks. Sudsang *et al.*[6] showed that for most prismatic objects, use of three robots is adequate for pushing. Here, however, an optimization approach[29–31] is used to systematically determine the number and final capture positions for the agents. Section 2 briefs this optimization scheme and shows that an adapted objective can yield a minimum three agents in many but not all cases. In Section 3, the modified projective path-planning algorithm employed online to guide the agents toward their respective goal points on the object's perimeter is presented. Differences between the modified algorithm and its previous implementation in ref. [29] are also detailed. Sections 4–6 describe the anatomy of the agent, camera calibration procedure, and the experimental setup. In Section 7, a template-matching algorithm is briefed using which the agents, object, and obstacles, all moving within the workspace, are identified using the central processing unit (CPU). Metrics for error analysis are described in Section 8, and experimental results are presented in Section 9. The results are discussed in Section 10 and conclusions are drawn.

## 2. Optimal Capture of a Prismatic Object

Sudsang *et al.*[6] note that three agents are sufficient to push a prismatic object. Their concept is based on the second-order immobility that incorporates curvature effects. That is, certain equilibrium grasp configurations around an object that do not achieve form closure can still effectively prevent any finite motion of the object. Sudsang *et al.*[6] employ a more general notion of *inescapable configuration space* (ICS) to demonstrate the possibility of pushing an object using three disc-shaped agents in absence of friction. They describe a method to construct maximum-radius workspace discs, where the robots can move independently and simultaneously yet keeping the object constrained within the ICS. Their algorithm requires three segments of a polygonal object to "enclose a triangle for all possible orientations of the object."

In comparison with refs. [29–31], we show via simulation that four robots that apply only normal forces at the respective contact points can optimally capture any prismatic object in form closure. We achieve optimality by ensuring complete object immobilization (i.e., in both translation and rotation) and aiming at the most possible reduction in (i) contact forces, (ii) difference between the magnitudes of clockwise and counter clockwise moments, and (iii) the number of capture points or agents. The object perimeter is discretized into a number of points. The problem of finding the best locations on the object is formulated as follows: "Given an object with '*n*' finite points on its boundary where mobile robots can apply only frictionless single contact forces, it is required to find the optimal grasp points for object capture." In ref. [29], the solution is sought by maximizing the following objective:

$$\text{Max} : f = \left\{ \frac{|M_{\text{cw}}| + |M_{\text{ccw}}|}{N^k} \right\} \left\{ \frac{1}{|N_{\text{cw}} - N_{\text{ccw}}| + s} \right\}, \quad (1)$$

Here, $M_{\text{cw}}$ and $M_{\text{ccw}}$ are the sums of clockwise and counter clockwise moments, respectively. Each moment due to an agent at an edge of the object is computed about the geometric center of the latter by assuming that the agent applies a unit normal force. Maximizing $|M_{\text{cw}}|$ and $|M_{\text{ccw}}|$ therefore maximizes the individual moment arms, which is equivalent to minimizing the contact forces. $N$ is the total number of robots engaged in capture and $k$ is a user-specified parameter. Minimization of $N^k$ (maximization of $1/N^k$) is equivalent to

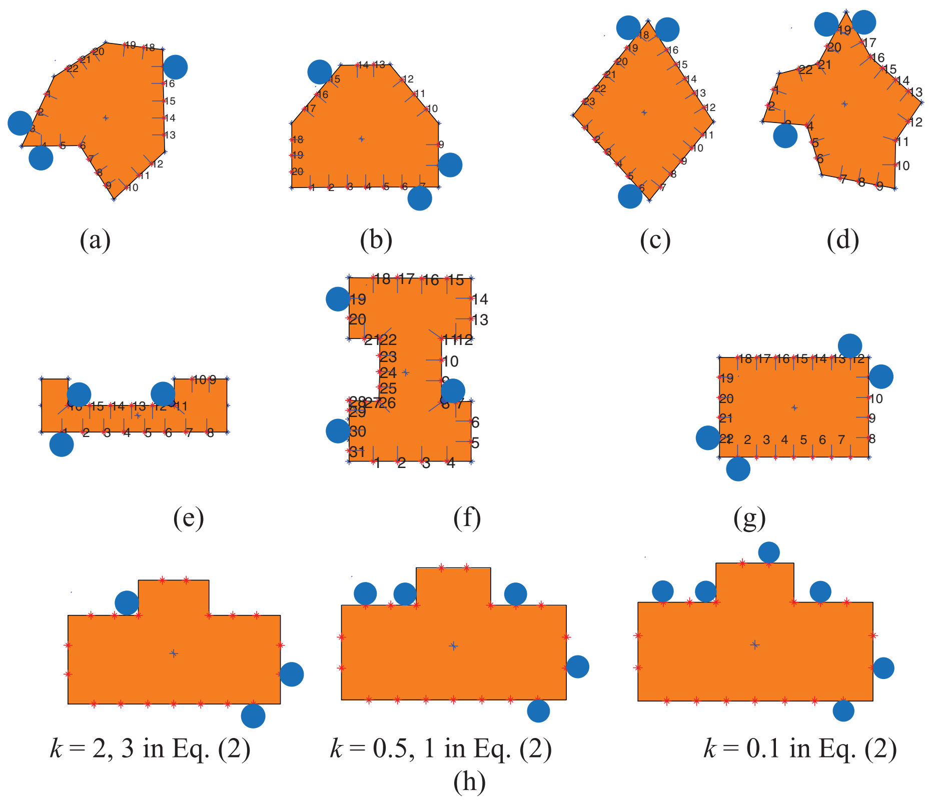


Fig. 1. (Colour online) (a–g) Optimal capture points/agents (circles) around objects of different shapes showing the minimum number of robots required.

minimizing the total number of capture points or robots. Further, if $k$ is reduced, the influence of minimization of the number of robots is lowered. If $k$ is set as zero, the number of capture points is not minimized. $N_{\text{cw}}$ is the number of robots resisting clockwise object rotation whereas $N_{\text{ccw}}$ are those that resist the anticlockwise moments. Minimization of $|N_{\text{cw}} - N_{\text{ccw}}|$ aims at making the number of agents resisting both moment types the same. $\varepsilon$, a small positive number, ensures a result in the case $N_{\text{cw}} = N_{\text{ccw}}$. In addition to the three aforementioned capture goals (i–iii) combined into a single objective in Eq. (1), a set of constraints is also used. A constraint based on local accessibility angles[29] for polygonal-shaped objects ensures that the object cannot translate or rotate even slightly when in catch. Other constraints used ensure that the robots are not clustered in one region and their positions do not overlap with each other. To allow for better distribution of agents, no two robots are permitted to be on the same edge of the object.

The discrete $n$ points are modeled as a binary string that is used as a design variable vector (e.g., 0000100001000100001000000...) in optimization. $n$ is user-specified and is indirectly governed by the size of the object and the agent. The optimal capture points can vary with $n$. A position in the string represents a discrete point on the object's perimeter. If a robot is present at a point, the corresponding entry gets the value "1." Otherwise, it remains "0." A large penalty is imposed on the objective in Eq. (1) if the accessibility angle constraint[29] is violated. Capture points are determined by maximizing the objective in Eq. (1) using a stochastic search (e.g., Genetic Algorithm[32] or a Random Mutation Hill Climber search). Since three independent objectives constitute Eq. (1), multiple solutions exist.

It is usually observed that maximization of Eq. (1) yields an even number of robots, often four and not less. This is because with four robots, the form closure grasp is guaranteed and further, $N_{cw} = N_{ccw} = 2$. A slight modification in Eq. (1) can allow for an odd number of capture points/robots with the same set of constraints used. If the second term in Eq. (1) is ignored, the new objective

$$\text{Max} : f = \left( \frac{|M_{\text{cw}}| + |M_{\text{ccw}}|}{(N)^k} \right) \tag{2}$$

can yield three minimum robots required for most prismatic objects. Figure 1 shows some examples of prismatic shapes (for $k = 3$ in Eq. (2)) that can be captured by three robots and also cases that need four agents. One can observe that in most cases, three robots can constrain the object's displacement. Even though a configuration may not be in force closure, the accessibility angle constraint[29] is satisfied. However, in some cases (e.g., Figs. 1(e) and (f)), one or more robots violate the single point contact restriction. That is, they are positioned at the corners to maintain contact with two edges. If this constraint is strictly adhered to, a fourth robot will be required to restrict the object's motion. In Fig. 1(g), as expected, capture is predicted by four robots. Thus, for prismatic shapes of which the edges are all at right angles, no three edges extend to form a triangle, and those that do not have concave corners, using the fourth robot or grasp point is essential. Fig. 1(h) illustrates the consequence of the reduction in parameter $k$ in Eq. (2). As expected, there

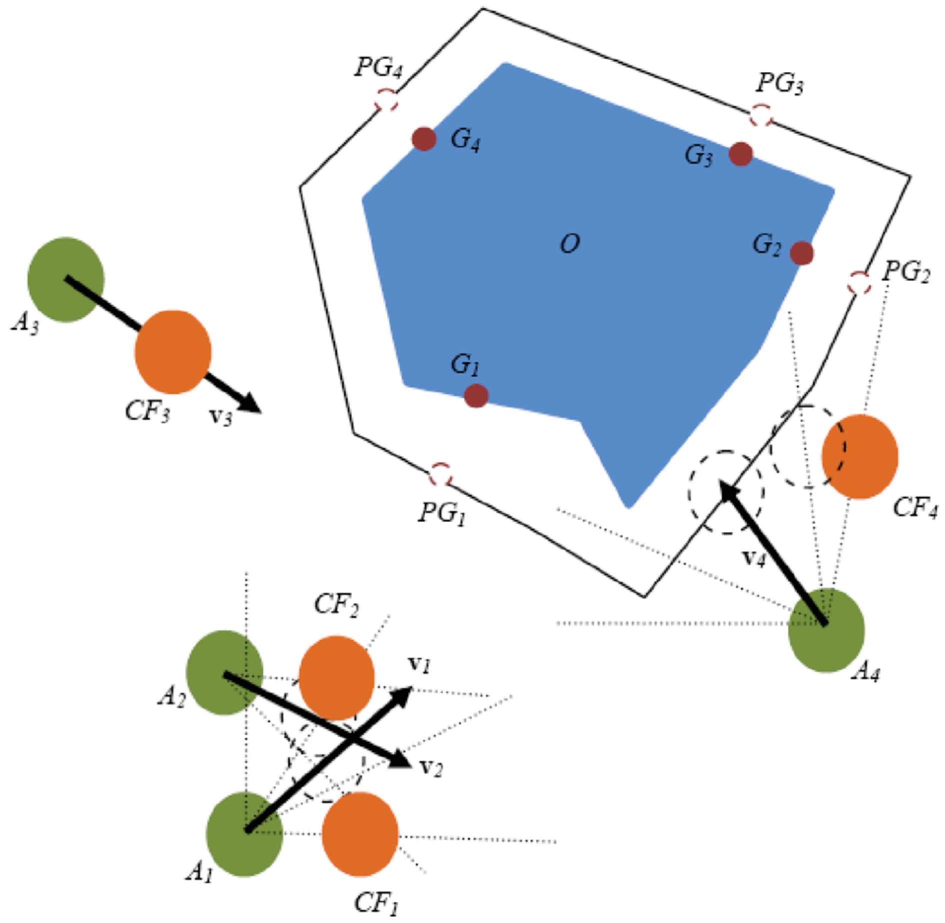


Fig. 2. (Colour online) The schematic of the projective path planning algorithm[29] for object capture. Key: O represents the object, $G_i$ ($i = 1, \ldots, 4$) are the goal points on the object for form closure, $PG_i$ ($i = 1, \ldots, 4$) are the pseudo goal points projected onto the $C$ space of the object that the agents $A_i$ first approach, $v_i$ are the current motion directions of the agents, $CF_i$ are the new, collision-free positions of the agents in the subsequent time-step. The dashed circles represent the projected positions of the agents depicting the search procedure for collision-free directions. $A_4$ shows collision avoidance with the object, while $A_2$ and $A_1$ depict collision avoidance between themselves.

is increase in the number of capture points. The object in Fig. 1(h) has vertical symmetry. Thus, mirror images of shown capture configurations are also the grasp solutions.

Since three robots may not be sufficient to restrain all modes of planar motion of any generic object, here we validate our experimental results with four robots. The reasons are as follows: (i) Form closure is guaranteed implying that both displacements and rotations of any arbitrarily shaped polygonal object can be restricted in a captured configuration, (ii) four robots help balance the clockwise and counter-clockwise moments better, (iii) contact forces are less with four robots than with three agents, and (iv) no significant ease in control or energy reduction is expected if the number of robots are reduced from four to three.

## 3. Modified Projective Path Planning

In earlier research, static objects are either surrounded by a large number of agents (e.g., in refs. [1–5]) or lifted by constraining them between robots or grippers (e.g., in refs. [6, 16, 17]). In this work, it is assumed that an object and obstacles move at low speeds along unplanned paths. The accessibility angle constraint[29] enables the determination of fail proof grasps. The best capture points on the object boundary where the agents should respectively approach and exert least possible forces are determined systematically. Using a large number of robots leads to increased energy consumption, traffic routing problems, and those in communication with host controller. With projective path planning, resources are utilized efficiently. Many previous efforts employ the potential field approaches to plan the paths offline. Herein, through a centralized vision-based online planner, collision checks are performed at each time-step. In case potential collision cases are forecasted, robots are made to change their current approach directions so that their subsequent directions are collision-free. The projective path-planning algorithm, as implemented in ref. [29], is summarized below. Details of the modified algorithm are described thereafter. Essential differences between the current and previous implementations are highlighted.

The primal goal of the projective path-planning algorithm is to determine robot positions at each time-step so that all possible collisions (agent–agent, agent–object, and agent–obstacle) are avoided. As implemented in ref. [29], the position of each agent is projected along its current direction of motion to simulate work environment in the subsequent time-step. Figure 2 summarizes the algorithm for capture.

If agent–agent collision is predicted, respective alternate collision-free directions are determined. For an agent, a sector search for a collision-free direction is initiated along candidate vectors to the left and right of the current motion vector. The sector angle between the left and right vectors is widened in case collision is foreseen. Once a direction is determined along which the agent is free to move, the search is ceased and the agent is moved forward. In case both directions are collision-free, one of them is chosen randomly for the agent. The object–agent collision if foreseen is modeled in three separate cases. Depending on the severity of the impending collision, the agent is either aligned with an object's edge or moved away from the object along the angular bisector of the two edges, or the normal to the potentially colliding edge. The agent–obstacle collision is treated as an agent–agent collision with one agent being stationary.

The motion-planning algorithm implemented herein is modified significantly from its previous version in ref. [29]. Modifications are performed for easier hardware implementation and relatively faster agent reaction during online capture. We assume that both, the object and obstacle(s) undergo unplanned but low-speed motions. Hence, their projected positions are assumed the same as their respective current positions when determining the subsequent motion directions for agents. Also, because the object and obstacles move randomly, their projected/subsequent positions are not known. In ref. [29], the projected positions for the deterministically moving object were considered, and since the obstacles were stationary, their projected and current positions were unchanged. All entities, e.g., agents, obstacles, and the object, within the work environment are identified via circular templates (see Sections 5–7). These templates are placed on the vertices of the object, and not on the form closure goal points. The designated capture points are stored parametrically. A parameter value is assigned to each edge. If an edge does not contain a capture point, the corresponding parameter value is zero. Else, the capture point on the $j$th edge is identified as $\mathbf{G} = \mathbf{v}_1(1 - t_j) + \mathbf{v}_2\, t_j$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the two vertices of this edge.

The CPU controls the motion of each agent using the following functions: (i) robot orientation, (ii) free motion, (iii) collision avoidance, and (iv) capture. In ref. [29], these functions were not implemented explicitly.

### *3.1. Robot orientation*

This function involves finding the orientation of a robot with respect to its goal point. In Fig. 3(a), let P1 be the center of the circular template on the right of the robot's center R. Let P2 be the center of the template on the left. This order is determined before the capture experiment is initiated when templates for the object, obstacles, and agents are identified sequentially. The template-matching algorithm (Section 7) gives the coordinates of these points at the rate of five frames per second. Let G be the point on the object that the robot is to approach. To determine whether the robot is oriented toward its goal, the cross product $\mathbf{RP2} \times \mathbf{RG}$ is used. If $\mathbf{RP2} \times \mathbf{RG} < 0$, P2 lies to the left of $\mathbf{RG}$, which implies that the robot is oriented toward its goal point. If the cross product is positive, the robot is oriented away from its capture point. The included angle between $\mathbf{RP2}$ and $\mathbf{RG}$ is termed as the *orientation angle* (Fig. 3(c)). The agent's forward direction of motion is defined such that P2 is always to the left of the robot's center. Compared to ref. [29], the orientation angle is newly defined and employed mainly by the capture function described later.

### *3.2. Free motion*

This motion is executed by the agent in cases where its involvement in collision is not foreseen. The agent is free to move toward its goal point. At every time-step, distances $|\mathbf{P1G}|$ and $|\mathbf{P2G}|$ are computed. The agent's motion is executed within five cases: (a) If $\mathbf{RP2} \times \mathbf{RG} < 0$ and $|\mathbf{P1G}| > |\mathbf{P2G}|$, the robot is rotated anticlockwise and moved forward toward the goal $\mathbf{G}$ by a small distance. (b) If $|(|\mathbf{P1G}| - |\mathbf{P2G}|)| < tol$ (a specified tolerance, here 8 pixels), the robot is only moved forward. (c) If $\mathbf{RP2} \times \mathbf{RG} < 0$ and $|\mathbf{P1G}| < |\mathbf{P2G}|$, the robot is rotated clockwise and moved forward. (d) If $\mathbf{RP2} \times \mathbf{RG} > 0$ and $|\mathbf{P1G}| > |\mathbf{P2G}|$ or $\mathbf{P1G}| < |\mathbf{P2G}|$, the robot is rotated clockwise. These cases are depicted in Fig. 3(a) (bottom). Implementation of the free agent motion in ref. [29] was rather less involved. Therein, rotations and translations were performed simultaneously. Also, rotation was limited by a permissible bound ($15^\circ$). Here translation is performed after the agent is rotated to align with the goal point, i.e., when $\mathbf{RP2} \times \mathbf{RG} = 0$. As the robots are assumed non-holonomic, allowing free and unbounded rotation provides additional flexibility in motion planning. If an agent is unable to move forward, it can always turn about its axis and retract. To ensure no template loss (see discussion), agent translations and rotations are performed at low speeds.

### *3.3. Collision avoidance*

This feature ensures that an agent does not collide with any other agent, obstacles, or the object. The feature is invoked if a part of an entity (say, point $A$ in Fig. 3(b)) is sighted within the projected position (or an envelope) of the agent. The function is executed with two cases. Let $\mathbf{d}$ be the current direction of motion of the agent. (a) If $|\mathbf{P1A}| < |\mathbf{P2A}|$, and if $\mathbf{d} \cdot \mathbf{RA} > 0$, the robot is rotated anticlockwise until $\mathbf{d}_{\text{new}} \cdot \mathbf{RA} < 0$. (b) If $|\mathbf{P1A}| > |\mathbf{P2A}|$, and if $\mathbf{d} \cdot \mathbf{RA} > 0$, the robot is rotated clockwise until $\mathbf{d}_{\text{new}} \cdot \mathbf{RA} < 0$. $\mathbf{d}_{\text{new}}$ is the new direction of motion for the agent. The robot then moves forward along the new direction until point $A$ is out of its projected position or the envelope. Point $A$ is taken as the nearest of the intersection points between the projected position of the agent and (a) the projected position of another agent or (b) the colliding edge(s) of an object in its current position. Collision between robot–robot, robot–obstacle, and robot–object is considered in three separate cases. Robot–robot collision between two agents is foreseen in cases where their projected positions intersect. In such cases, the collision avoidance function is executed for both robots. An obstacle is modeled by its center and a virtual circular envelope, the radius of which depends on the size of the obstacle. If the envelopes of the obstacle in its current state and the robot in its projected state intersect, collision avoidance is performed for the robot.

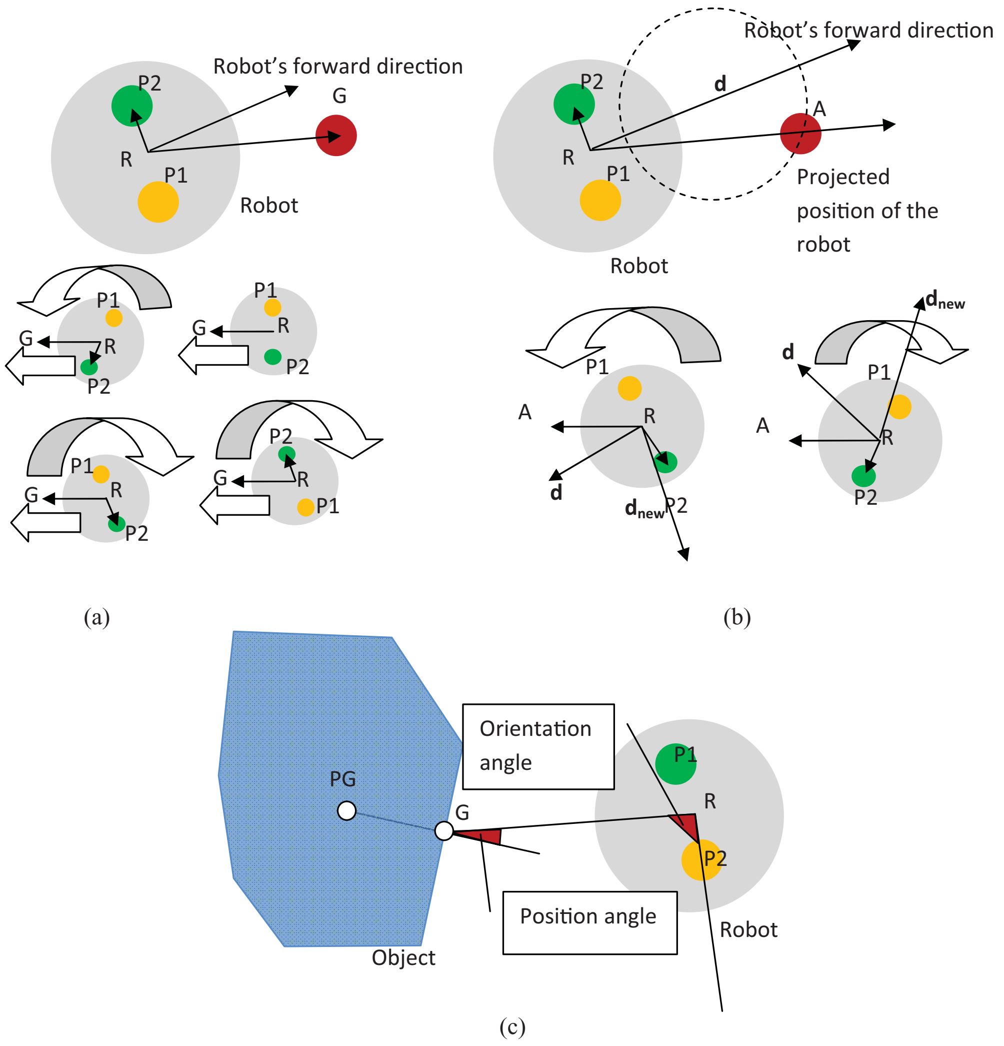


Fig. 3. (Colour online) The robot is shown in gray with two circular templates shown in green and yellow. The red circular template represents the goal point G. Angle between RP2 and RG is the orientation angle. (a) Cases of free motion of the agents. The thick arrows depict potential robot rotation and then translation directions. (b) Agent in collision avoidance with point A. (c) Orientation and position angles.

An object is identified by its vertices. In each frame, the minimum distance between a robot and the object is computed. First, min(**R** − **CP**$_i$) is registered, where **R** is the robot's center and CP$_i$, $i = 1, 2, \ldots, n$ are the object's vertices. Let $\mathbf{CP}_{min}$ be the vertex of the object nearest to the robot. The edges of the object passing through $\mathbf{CP}_{min}$ are also registered. $D_{min}$, the distance between the robot and the object is the minimum distance between the end points of the two registered edges and the center of the robot. The function also determines the point on the object's perimeter, and it is nearest to the robot. If this point does not correspond to the robot's goal point, the object is treated as an obstacle and collision avoidance function is executed for the robot. Otherwise, the capture function discussed below is implemented.

The collision avoidance function is relatively simpler, more conservative, and faster to execute than that used in ref. [29]. The sector search method in ref. [29] to prevent two agents from colliding is not used here, since the time required in determining the free directions for the agents is significantly more. Unlike in ref. [29], no convex envelope is considered for the object. Here the agents approach the respective goal points on its perimeter directly. The object–agent collision in ref. [29] was considered in three separate cases. Therein, a robot was either aligned to one of the object's edges, or was pushed back. The latter occurred when the agent was very close and was forecasted to hit one of the vertices or edges of the object. Accordingly, the agent was pushed away either along the angular bisector or normal to the colliding edge(s). Here collision avoidance is much simpler and conservative. Whenever collision is predicted, the agents turn back and retract. More precisely, the dot product between their current and new directions of motion becomes negative.

Whenever the distance between a robot and its specified goal is within some tolerance (here 100 pixels), the capture function is executed. The collision avoidance function between the robot and the object is switched off. In the capture function, robot movements are based on the *position angle* (Fig. 3(c)) subtended near its goal point. To compute the position angle, a pseudo goal point PG is identified. PG is the goal point projected into the object along the normal to the edge at a distance of 20 pixels from the goal point G

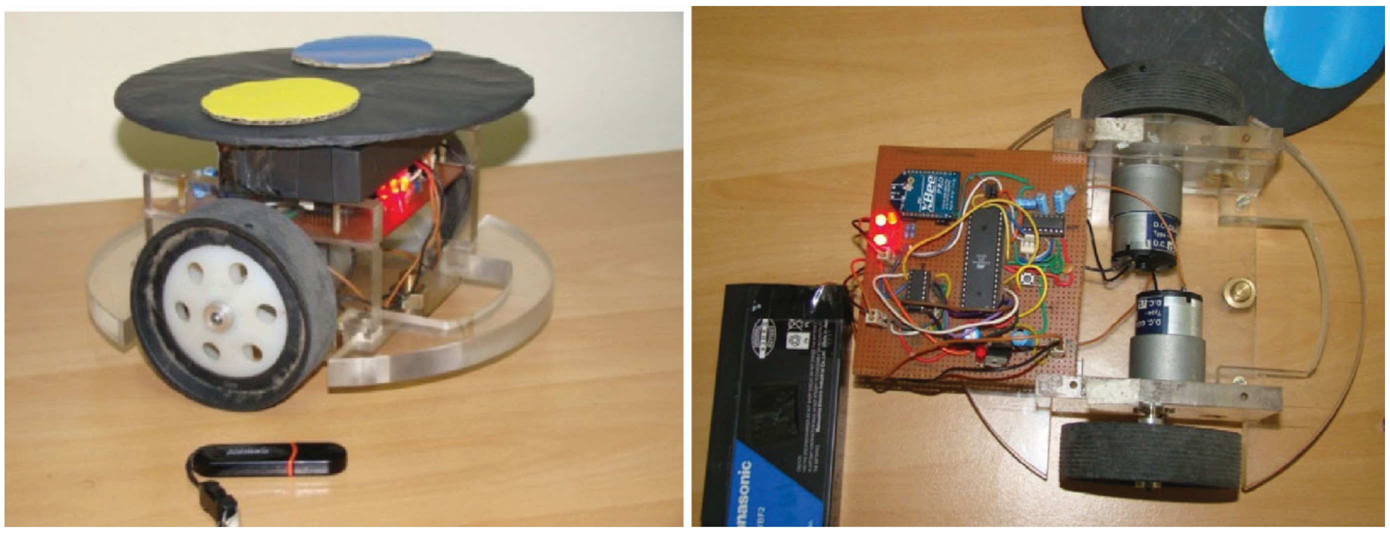


Fig. 4. (Colour online) The mobile robot designed and used for object capture.

(Fig. 3(c)). The angle between **PGG** and **GR** is the position angle. If **R** is to the left of **PGG** when viewing from the robot's center toward its capture point (i.e., $\mathbf{PGG} \times \mathbf{GR} < 0$), the position angle is considered as negative and *vice versa*.

The capture function in ref. [29] is such that after all agents reach the projected goal points on the convex envelope of the object, they converge together toward their respective true goal positions on the object's boundary. Therein the position and orientation of the agents at their capture positions were not used. Capture was considered accomplished if the distances of the robots from their respective goals were within the specified threshold. In the present work, the capture function is implemented to ensure that the robot center is almost along the normal to the edge at the respective goal points. Also, the robot is oriented such that its subsequent motion direction is nearly normal to the contact edge, and toward the object. In other words, capture herein is performed more rigorously. Thus, for a robot, the tolerance range for the position angle is (between –20° and 20°) and that for the orientation angle is (−15°, 15°) from the ideal angle which is 90°. These ranges may vary with different laboratory settings (size of the agents and object, camera position, or other parameters) and capture requirements, and accordingly, the capture function can be made to be stricter (by decreasing the thresholds) or relaxed. The function is executed with three cases.

### *3.4. Capture*

*Case 1*: Position angle between −20° and 20°. The robot is almost in front of its goal point but its orientation angle may not be within the specified limits. In that case all robots are allowed to approach their angle tolerance limits. Thereafter all robots approach the object for capture. If the position angle of a robot is between −20° and 20° and the robot is oriented toward its goal point but the orientation angle is not within the specified tolerance, i.e., if $|\mathbf{P1G}| > |\mathbf{P2G}|$, the robot is rotated anticlockwise, otherwise in clockwise direction. If the robot is oriented away from its goal point, i.e., if $|\mathbf{P1G}| > |\mathbf{P2G}|$, the robot is rotated clockwise, otherwise in the counterclockwise direction.

*Case2*: Position angle less than −20°. Here the position angle needs to be increased. If the robot is oriented toward its goal point, it is rotated clockwise. Otherwise, if the motion direction of the robot is opposite to where its goal is, and if the orientation angle is less than 20°, the robot is moved forward. By moving forward, the position angle of the robot is increased to an extent that case 1 can get invoked. It may happen that the position angle becomes more than 20° and the robot is oriented away from its goal point. In that case, the robot is rotated anticlockwise.

*Case 3*: Position angle is greater than 20°. The position angle needs to be reduced. If the robot is oriented toward its goal point, it is rotated anticlockwise. Otherwise, if the robot is oriented away and if the orientation angle is less than 20°, robot is moved forward so that the position angle is reduced. The capture function switches to case 1 if the position angle is reduced to less than 20°.

When these cases are explored, the robot approaches its respective goal point. The capture process is stopped when all robots are close to their goal points within the permissible tolerance. The orientation angle criterion is not used in this decision-making. However, the exact goal point may not be reached and/or the subsequent agent motions may not be directed normal to the respective edge of the object. These errors are quantified in Section 8.

## 4. Mobile Robots

Four mobile robots (Fig. 4) were fabricated to experimentally perform and evaluate object capture for numerous cases with different object shapes. Each robot has two large side wheels and a castor wheel for frontal support. With the two side wheels actuated independently, the robot can move forward, backward, sideways, or turn at any angle about its axis. A robot is driven by two DC motors with inbuilt gearbox, and is controlled using an *Atmega16* microcontroller and a power amplifier. It has two colored circular templates on its top, based on which its position and orientation are identified by an overhead camera (see Section 6). Each robot is powered by an onboard lithium-ion rechargeable battery.

The command transfer from the CPU (see Fig. 6) to the agent is accomplished through the X-Bee wireless module.

## 5. Camera Calibration

Before the experiments are performed, the overhead camera is calibrated using the standard Tsai calibration method.[33] The camera used is a 1.3-mega pixel camera with a focal length of 30 mm and a resolution of 640 × 480. Coplanar, fully optimized camera calibration is performed, and the following parameters are calculated: *f*: focal length of the camera, *k*: radial lens distortion coefficient, *Cx* and *Cy*: coordinates of the center of radial lens distortion, *Sx*: scale factor to account for any uncertainty because of imperfections in hardware timing for scanning and digitization, *Rx*, *Ry*, and *Rz*: rotation angles for the transformation between the world and camera coordinates, and *Tx*, *Ty*, and *Tz*: 3-dimensional (3D) translation components for the transformation between the world and camera coordinates.

The transformation from world ($X_w$, $Y_w$, and $Z_w$) to image ($X_i$, $Y_i$, and $Z_i$) coordinates considers the parameters of the camera in the following equation:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{T}, \tag{3}$$

where **R** and **T** are the 3 × 3 rotation and 3 × 1 translation matrices, respectively. **R** is obtained by Tsai's algorithm. The transformation from the 3D position (in the image coordinate frame) to the image plane is computed. The workspace (2m × 1.8 m, Section 6) is divided into blocks. On each block, circular templates of different colors are placed. The center of each template is taken as the corresponding block's center point. The coordinates of each block (world coordinates) are known. The coordinates of the center of each block are calculated in terms of pixels. We take readings for a total of 300 points. For better accuracy, colored templates are positioned at the same height in the workspace as that of each robot's top surface on which the templates are placed during the experiments. The optimized parameters obtained from Tsai's algorithm are obtained as follows:

$$\begin{aligned} f &= 45.52\,\text{mm},\ \kappa 1 = 5.23 \times 10^{-4}\,1/\text{mm}^2, \\ Sx &= 1.00,\ Cx = 371.60,\ Cy = 270.25\,\text{pixels}, \\ Tz/f &= 141.63. \end{aligned}$$

Figure 5 quantifies the calibration error in the world coordinates. In the figure, "X" and "Y" coordinates give the actual dimension of workspace (in mm) and "Z" axis gives the error. The maximum error is about 17.02 mm and the minimum error is about 0.57 mm. The normalized calibration error (as defined in ref. [33]) is computed as 6.74 pixels or 1.91 mm. Some sources for the error can be attributed to the heterogeneous lighting conditions, noise, or uneven flooring. Based on the calibration results, 1 mm in the world coordinates is equivalent to 3.53 pixels.

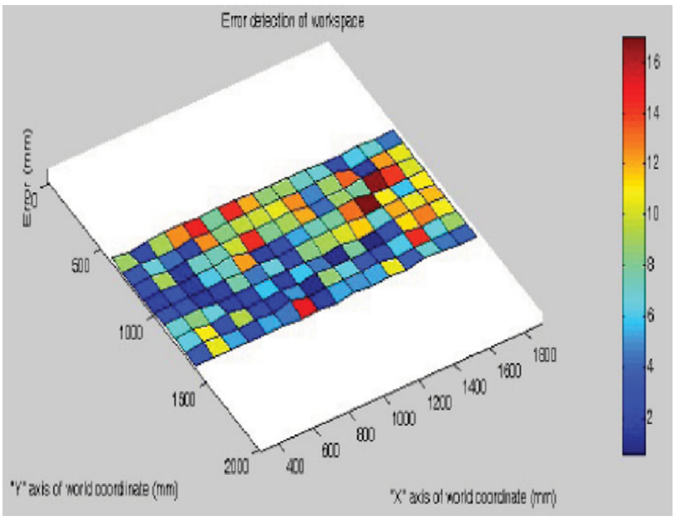


Fig. 5. (Colour online) Workspace error.

## 6. Experimental Setup

The experimental setup consists of a 2 m × 1.8 m workspace, on which the object, obstacles, and the mobile robots move. The objects considered are of various shapes and sizes and are composed of light plastic-based material covered with black paper. Colored templates are placed on the vertices of an object to track its motion via vision. The setup is shown in Fig. 6, in which there are four mobile robots capturing a gradually moving prismatic object while avoiding slowly moving obstacles. The figure at the center of the bottom row shows the overall setup that comprises a work environment, a CPU, and an overhead camera. Figures at bottom-left, bottom-right, and top-middle depict various components of the setup. Images from the work environment are taken at the intervals of five frames per second from the overhead camera and transferred to the processing unit. The processing unit analyzes the images, i.e., colored templates placed atop the agents, object, and the obstacles. The positions of the object and obstacles, and the positions and orientation of the agents are determined. Based on these, the processor computes new positions and orientations of the agents. This information is relayed to the agents via wireless commands.

## 7. Template-Matching Algorithm

The template-matching algorithm tracks colored templates placed on different entities in real-time (Fig. 7). At the start of the motion-planning algorithm, the camera provides the initial configuration of the workspace (i.e., the position and orientation of entities). With the CPU, each circular template on a robot is individually identified by manually clicking near its center. A rectangular template of 20 × 20 pixels is cropped around the center of the circle. If there are *n* circles of different colors to be tracked, then *n* such square templates are generated (Fig. 7) and stored as respective images. In every consequent frame, the algorithm searches for the template in the neighborhood of its previous location. Reducing the height and width of the search window in the image for the next search reduces processing time.

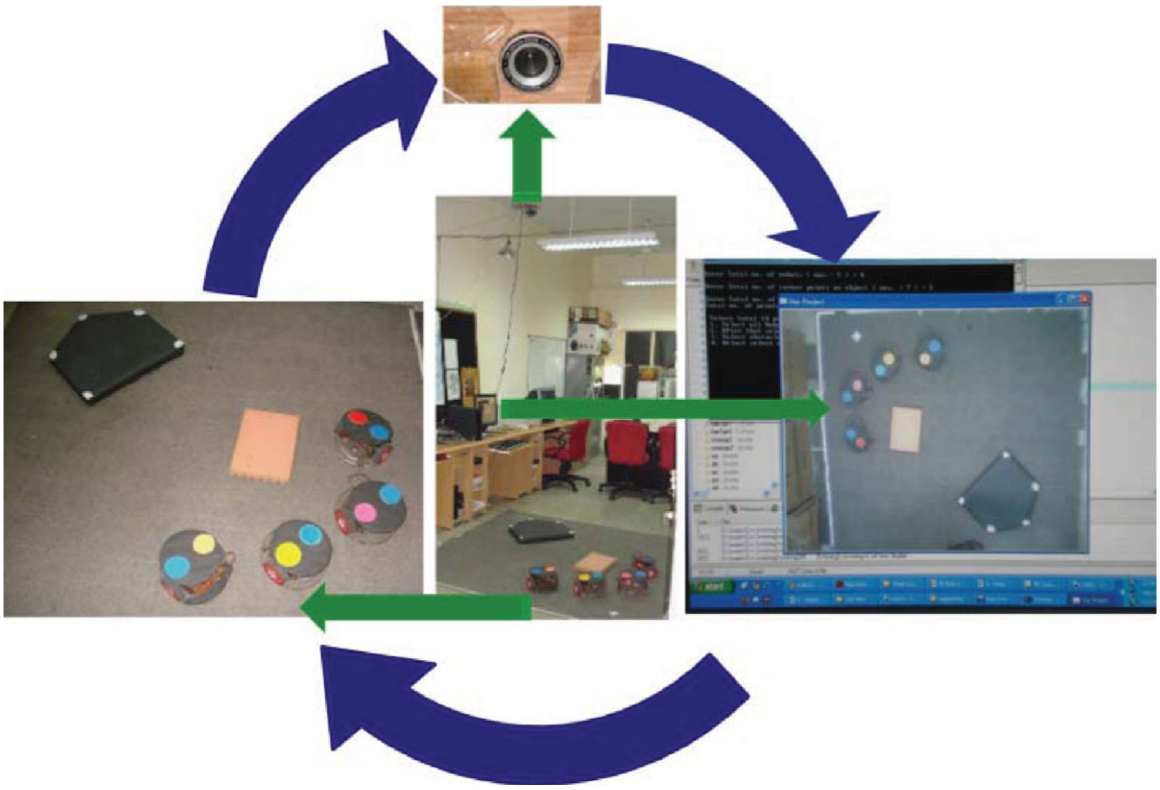


Fig. 6. (Colour online) (Bottom, center) The overall experimental setup; (bottom, left) environment wherein the four mobile robots capture the gradually and randomly moving object while avoiding dynamic obstacles; (top, center) the overhead camera, which sends visual feedback to the CPU; (bottom, right) the CPU, which processes the images and sends motion commands to the agents.

A search area of 60 × 60 pixels$^2$ is used around each template for this purpose. Each search box is cropped around the square template and saved as an image. The center coordinates of the template in the corresponding cropped search boxes are determined. These coordinates in the overall image frame are then computed. Suppose the initial position of the center of the template is (x, y) with respect to the X and Y axes (Fig. 8). The template moves to a new location such that its center coordinates become (new x, new y) in the square search box. The new coordinates of the template become (new x + x – 30, new y + y – 30) with respect to XY. The next search square is a 60 × 60 square with center at (new x + x – 30, new y + y – 30), and this process continues.

## 8. Error Analysis for Capture

In order to quantitatively validate our experimental results, we determine how far are the agents from the respective

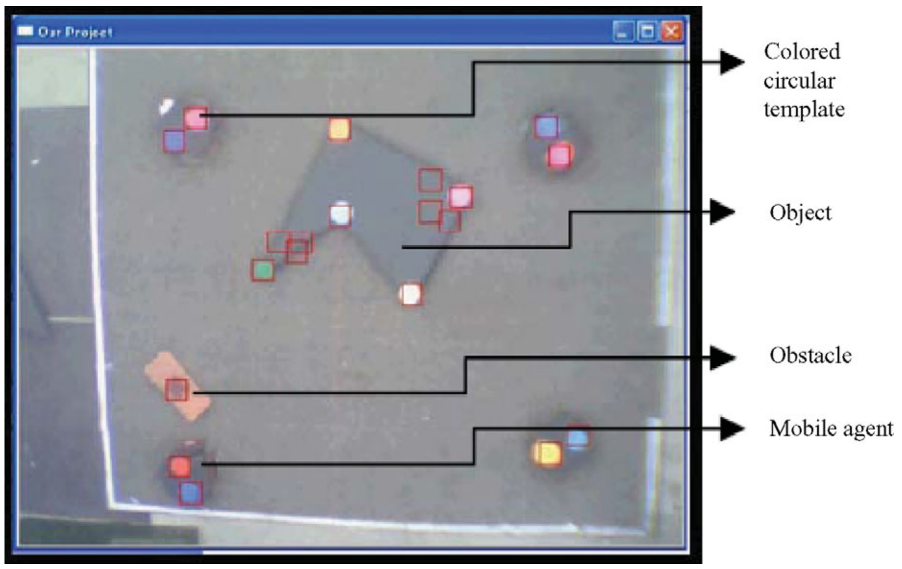


Fig. 7. (Colour online) Tracking of colored templates.

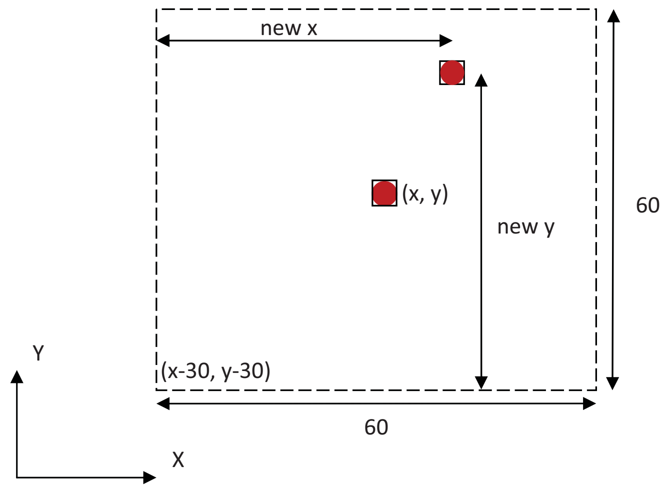


Fig. 8. (Colour online) Relocation of the search box (shown using dashed lines) as the circular template (in red) moves to a new location.

computed goal points during capture. We perform the error analysis as follows. We consider the actual goal point as that calculated via optimization. However, a robot may not approach the actual goal point but a point close to it. We call this point as the captured goal point. If $(X_{\mathrm{ag}}, Y_{\mathrm{ag}})$ are the $X$ and $Y$ coordinates of the actual goal point, and $(X_{\mathrm{cg}}, Y_{\mathrm{cg}})$ are the same for the captured goal point, the variation $\delta dg$ in this distance $dg$ between the two points is given by

$$\delta dg = \frac{\left|X_{\mathrm{cg}} - X_{\mathrm{ag}}\right| \left|\delta(X_{\mathrm{cg}} - X_{\mathrm{ag}})\right| + \left|Y_{\mathrm{cg}} - Y_{\mathrm{ag}}\right| \left|\delta(Y_{\mathrm{cg}} - Y_{\mathrm{ag}})\right|}{dg}. \tag{4}$$

The maximum value of $|\delta(X_{ig} - X_{jg})|$ or $|\delta(Y_{ig} - Y_{jg})|$ where $i$ and $j$ represent subscripts "a" or "c" can be considered as $R$, the radius of a colored template. The maximum possible error between the captured and actual goal points can be

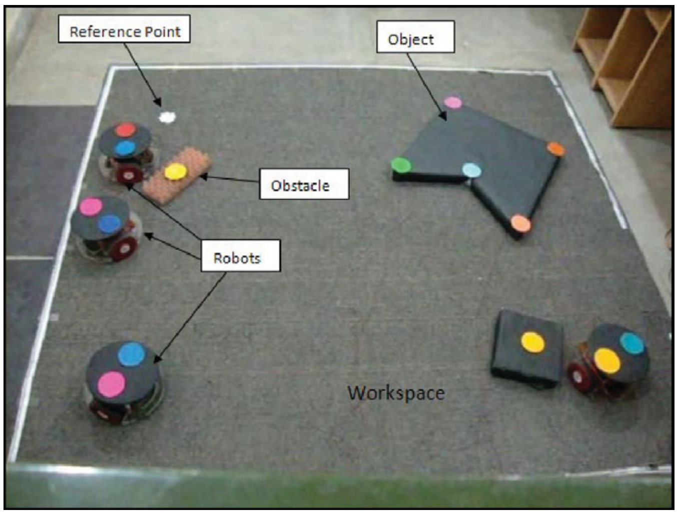


Fig. 10. (Colour online) The work environment for object capture.

found by

$$E\max = |dg| + |\delta dg|. \tag{5}$$

The foregoing analysis suggests that the error is expected to reduce if colored templates of smaller sizes are used at the vertices of the object. This is also verified in Section 9.

## 9. Experiments and Results

We test five different objects for capture with multi-agents using the modified projective path-planning algorithm. Images of the objects are shown in Fig. 9 (top). The optimal capture points for each case are shown in Fig. 9 (bottom). The capture points are determined by maximizing the function in Eq. (1) using Gordy's search algorithm.[32] The workspace, agents, object, and obstacles are all shown generically in Fig. 10.

The capture sequences for the respective five objects are shown in the Figs. 11–15. Each object is identified by the camera via glued colored templates at vertices. The obstacles (pink or black) are rectangular shaped and are identified

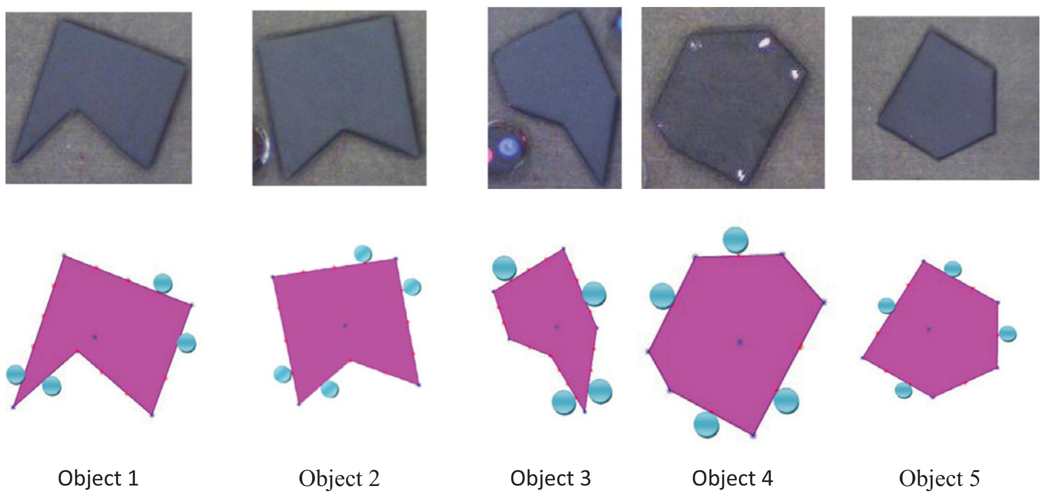


Fig. 9. (Colour online) (Top) Five different prismatic objects, and (bottom) the corresponding form closure points/goal points for agents.

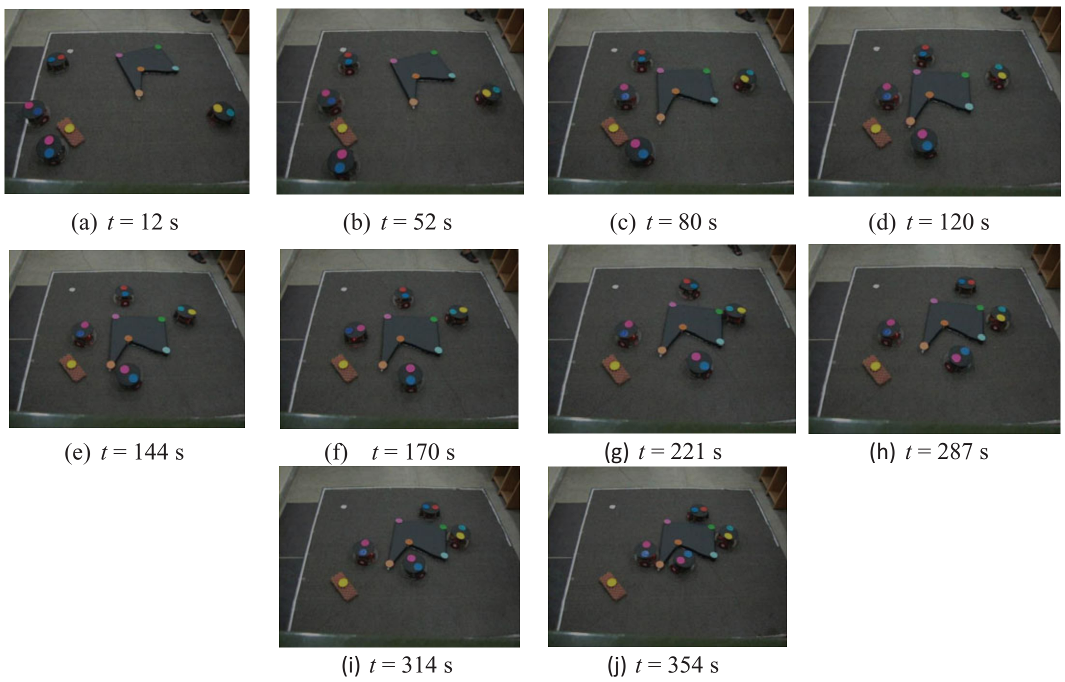


Fig. 11. (Colour online) Sequence illustrating the capture of object 1. The object changes its position/orientation in frames (b–f). The stationary obstacle is placed at bottom-left of the workspace. Three agents are initially placed on the left, while the fourth is placed on the right.

using a single colored template over them. The first image (a) in each figure shows the workspace just before the start of the experiment. The final image (j) in the figures shows the respective objects in form closure, arrested by all four robots. In each figure, the workspace (2 × 1.8 m) or (640 × 480pixel) is shown by a white rectangular boundary. To manipulate an object in an unplanned manner, the latter is pulled manually and arbitrarily through the attached tethers. Care is taken that the displacements imparted to the object are small. The object motion can be observed through the image sequence with respect to the reference cross mark. All figures exhibit successful capture of objects under different conditions. In each case, the number and placement of obstacles, starting positions of agents, and object motion are different.

*Error in capture*: At the final capture configuration, the agents do not reach the designated actual goal points but get placed near the captured goal points. Fig. 16 (left) compares the actual (shown using crosses) and captured (shown using circles) goal points for the five used objects. The tables on the right depict the error values as per Eq. (5). The errors are as high as 16%.

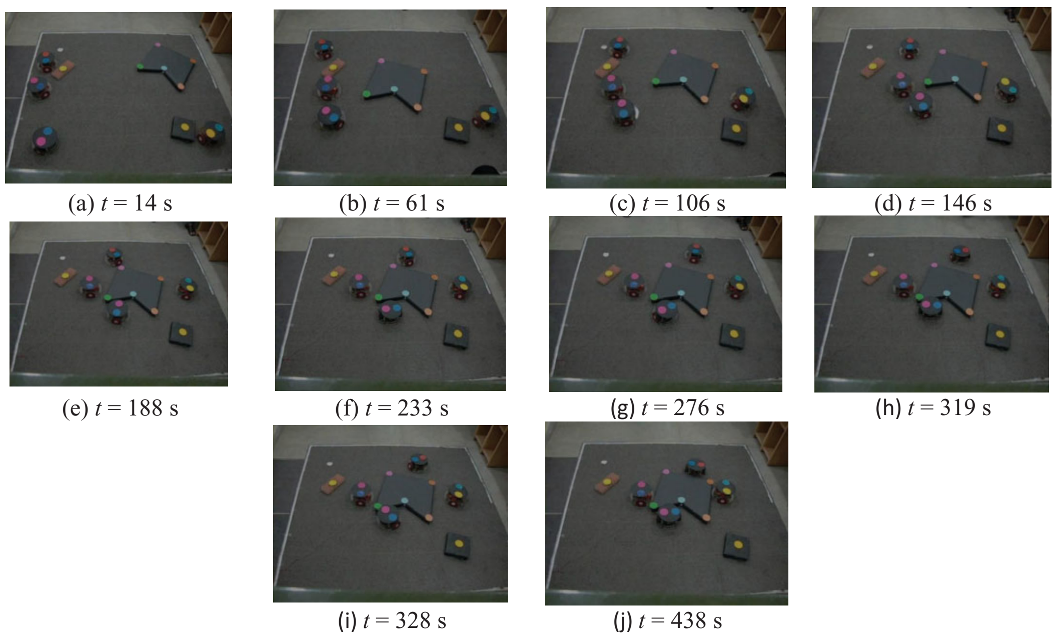


Fig. 12. (Colour online) Capture of object 2. The object changes position in the first four frames. A stationary obstacle is placed near the reference "+" point (top left), and another at the bottom right.

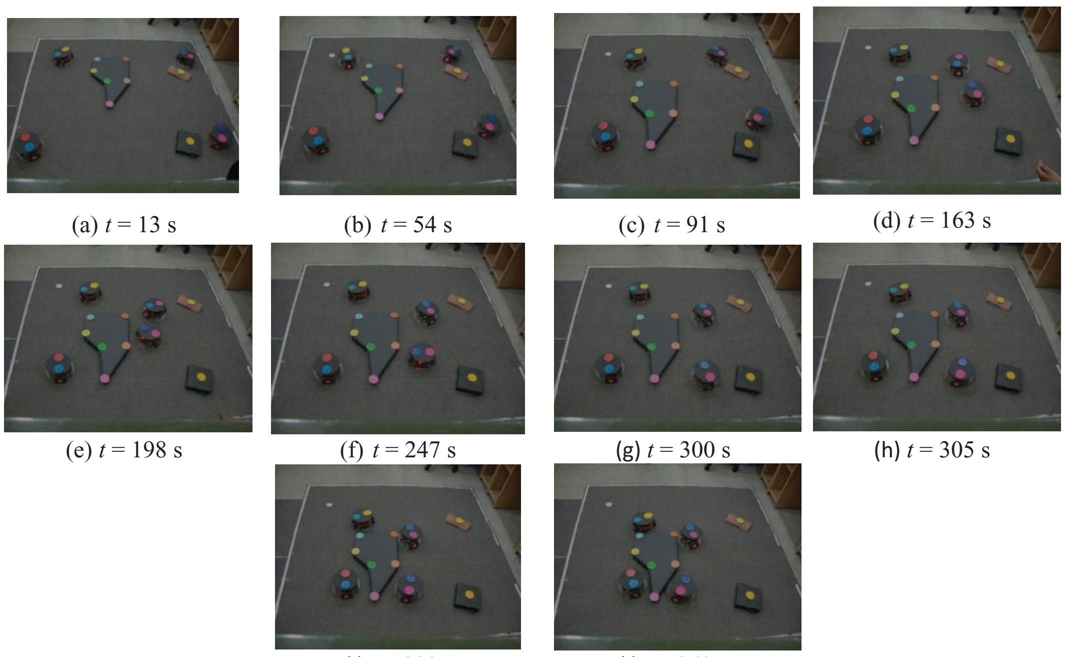

(a) $t = 13$ s (b) $t = 54$ s (c) $t = 91$ s (d) $t = 163$ s (e) $t = 198$ s (f) $t = 247$ s (g) $t = 300$ s (h) $t = 305$ s (i) $t = 323$ s (j) $t = 360$ s

Fig. 13. (Colour online) Capturing object 3. The object is gradually moved from top to center of the workspace. Two obstacles, each of rectangular shape (pink and black with one template each over them), are present in the workspace (top and bottom right). The agents initially commence from four different corners of the workspace.

Slightly better results are achieved if reduced template sizes are used at the vertices of the objects. Experiments performed suggest that for template size of less than 30 mm in diameter, it is more likely that the overhead camera will fail to detect some circular templates. This is referred to as frame/template loss. For the experiments performed, the template size of around 40 mm in diameter was found reasonable while capturing the objects. Thus, this value is the best tradeoff between accuracy of goal computation and likelihood of frame loss. Experiments are performed again with objects 2 and 5 with this reduced size of circular templates, which are used on the perimeter of these objects. Figure 17 depicts errors between the assigned and captured goal positions. Figures 18 and 19 show the capture process in different time-steps.

In addition to approaching the specified goal points accurately, it was also desired via the new capture function (Section 3.4) that the orientation angles for all the agents at

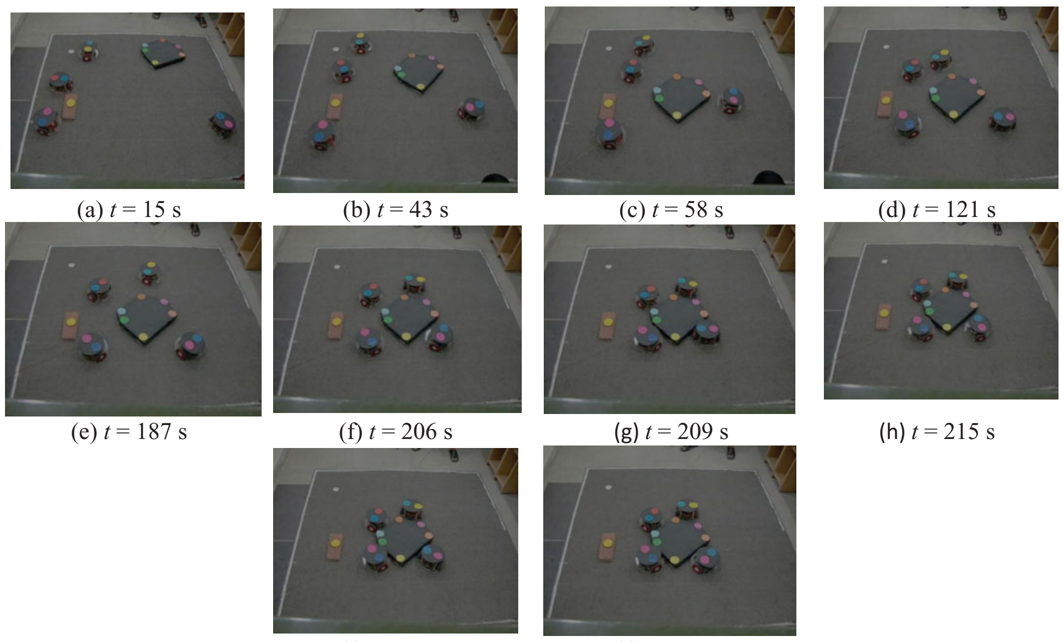

(a) $t = 15$ s (b) $t = 43$ s (c) $t = 58$ s (d) $t = 121$ s (e) $t = 187$ s (f) $t = 206$ s (g) $t = 209$ s (h) $t = 215$ s (i) $t = 221$ s (j) $t = 306$ s

Fig. 14. (Colour online) Object 4 in capture. The object is gradually moved from the top right corner of the workspace to the center.

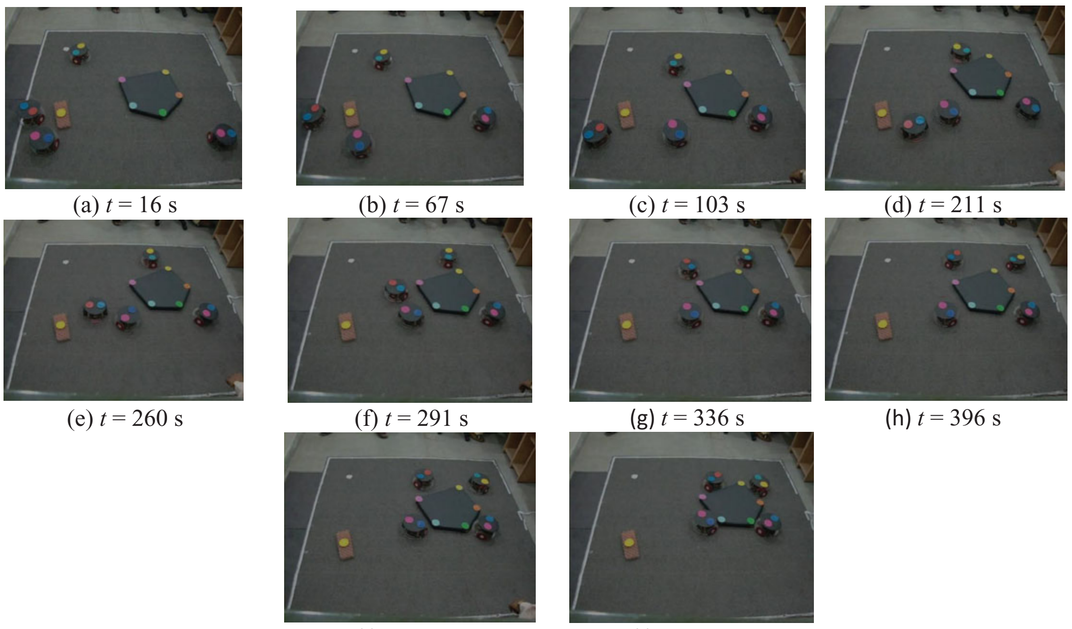


Fig. 15. (Colour online) Capturing object 5. This convex object is captured near the right edge of the workspace.

their respective capture positions were close to 90° . Such an effort was not made in ref. [29]. The orientation angles and their deviations from the desired values are given in Table I. The orientation angle is taken as $90 - |\cos^{(-1)} (\mathbf{e} \cdot \mathbf{P1P2})|$, where vector **e** corresponds to the edge, and **P1P2** is the vector joining the centers of two circular templates placed over the respective agent. The worst deviation is close to 23° compared with the permissible 15° variation.

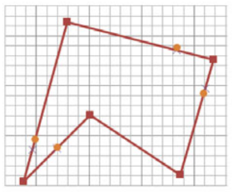
## 10. Discussion and Closure

This paper validates the capture of a randomly but gradually moving object experimentally with four robots using the modified projective path-planning method. Experiments are performed with stationary and/or randomly moving obstacles. Both object and obstacles are assumed to exhibit low-speed motion. The capture points around the prismatic object are determined so that the clockwise and counterclockwise moments are maximized (i.e., contact forces are minimized), and the object is seized in form closure. A standard algorithm is used to calibrate the overhead camera, which captures the images from the workspace and passes them to the CPU. The CPU analyzes the workspace and relays the motion commands via wireless modules to the agents so that they approach their respective goal points while avoiding collision between themselves, the obstacles, and the object. Form closure points are determined prior to performing the experiments, but path planning for the agents is performed online.

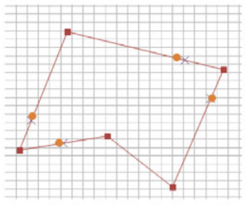
Although four robots are used for form closure capture in this paper, less or more number of robots may be required for capture in general. The modified projective path-planning algorithm presented herein is generic and can accommodate any number of robots. Sudsang *et al.*[6] discussed the possibility of surrounding an object with three instead of four (or more) robots. They used the concept of ICS, within which an object can be restrained. This work differs from the one by Sudsang *et al.*[6] in that the latter focuses on pushing an object through the workspace having stationary obstacles. Here the aim is to capture a moving object with dynamic obstacles within the workspace. Nevertheless, with appropriate modification in the objective in Eq. (1), it is possible to determine three capture points for most prismatic objects (Fig. 1). For rectangular objects, four agents are needed. This is because such objects will always have at least 1 free DOF irrespective of how the three

Table I. Orientation angles and errors for different agents R1–R4.

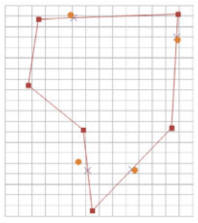
| | Orientation angle (*OA*) | | | | Error = \|90° − *OA*\| | | | |
|---|---|---|---|---|---|---|---|---|
| Capture case | R1 | R2 | R3 | R4 | R1 | R2 | R3 | R4 |
| Object 1 (Fig. 11) | 84 | 74 | 77 | 79 | 6 | 16 | 13 | 11 |
| Object 2 (Fig. 12) | 80 | 77 | 87 | 76 | 10 | 13 | 3 | 14 |
| Object 3 (Fig. 13) | 81 | 67 | 74 | 79 | 9 | 23 | 16 | 11 |
| Object 4 (Fig. 14) | 79 | 78 | 76 | 65 | 11 | 12 | 14 | 15 |
| Object 5 (Fig. 15) | 75 | 87 | 87 | 72 | 15 | 3 | 3 | 18 |
| Object 2 (Fig. 18) | 76 | 79 | 84 | 73 | 14 | 11 | 6 | 17 |
| Object 5 (Fig. 19) | 78 | 87 | 88 | 74 | 12 | 3 | 2 | 16 |



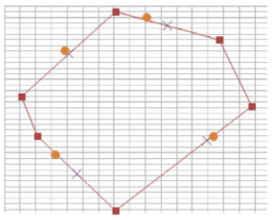
| Goal point | $E_{\max} = (\lvert\delta dg\rvert+\lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 9.31 | 4.01 |
| 2 | 12.80 | 8.30 |
| 3 | 0.70 | 1.08 |
| 4 | 17.31 | 15.97 |
| Max | 17.31 | 15.97 |



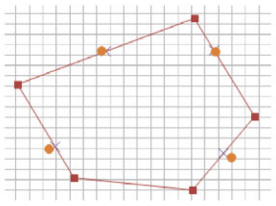
| Goal point | $E_{\max} = (\lvert\delta dg\rvert+\lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 20.48 | 11.57 |
| 2 | 9.74 | 2.33 |
| 3 | 12.02 | 5.43 |
| 4 | 11.62 | 7.18 |
| Max | 20.48 | 11.57 |



| Goal point | $E_{\max} = (\lvert\delta dg\rvert+\lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 24.03 | 15.27 |
| 2 | 3.40 | 5.23 |
| 3 | 15.95 | 5.45 |
| 4 | 2.91 | 4.48 |
| Max | 24.03 | 15.27 |



| Goal point | $E_{\max} = (\lvert\delta dg\rvert+\lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 16.97 | 4.35 |
| 2 | 27.58 | 22.02 |
| 3 | 9.61 | 8.14 |
| 4 | 32.17 | 27.74 |
| Max | 32.17 | 22.02 |

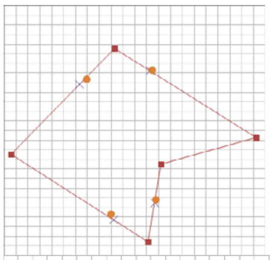


| Goal point | $E_{\max} = (\lvert\delta dg\rvert+\lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 12.5 | 3.84 |
| 2 | 15.93 | 3.09 |
| 3 | 21.5 | 11.53 |
| 4 | 7.85 | 7.60 |
| Max | 21.5 | 11.53 |

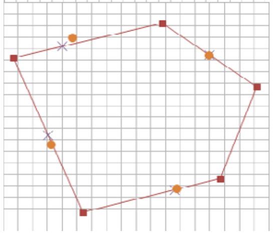
Fig. 16. (Colour online) The agents do not approach the designated goal points but capture the object at points near to them. Tables on the right show the error in the agent positions per goal point, and also the maximum errors. KEY: Squares represent the vertices of the objects, crosses depict the designated goal points determined using the optimization procedure, and circles show the positions where the agents reach to restrain the object.

agents are placed around it. In general, the problem of finding a minimum number of agents required for object capture depends on the following factors: (i) All external forces and moments (if any) exerted by a dynamic object should be considered; (ii) the size and density of the object, and friction should not be ignored; and (iii) the maximum contact force an agent can apply without being pushed back should be taken into account. Factors (ii) and (iii) are particularly important for pushing applications. To determine the capture points, the objective in Eq. (1) can be generalized accordingly to accommodate any number of agents, and force balance on each agent (the resultant force applied on it should not be greater than the frictional force) while not allowing any free passage to a moving object. The minimum number of agents is therefore neither three nor four for a generic problem.

Figure 16 depicts the errors (discrepancy between specified and actual goal points) in capture for five objects. For object one, the maximum error between the designated and captured goal points is recorded as 17.3 mm. For objects 2–5, they are, respectively, 11.6, 15.3, 22, and 11.5 mm. These errors correspond to the cases where a template size of 50 mm is employed. With reduced template size, i.e., 40 mm, the maximum errors between the assigned and captured goal points for objects 2 and 5 are 12.5 mm and 15 mm, respectively. The errors are lowered with reduced template size but not significantly. With regard to the error in orientation angles, the maximum deviation recorded is close to 23°, which is marginally higher than the permitted limit of 15° . A major source is the calibration error depicted in Fig. 5, which might have been caused due to non-homogenous lighting, improper camera orientation (i.e., not perpendicular



| Goal No. | $E_{\max} = (\lvert\delta dg\rvert + \lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 13.04 | 10.22 |
| 2 | 9.24 | 12.47 |
| 3 | 6.68 | 3.48 |
| 4 | 6.43 | 5.85 |
| Max | 13.04 | 12.47 |



| Goal No. | $E_{\max} = (\lvert\delta dg\rvert + \lvert dg\rvert)$ | % Error in actual and captured goal points w.r.t. robot diameter |
|---|---|---|
| 1 | 15.06 | 13.63 |
| 2 | 9.96 | 14.98 |
| 3 | 5.5 | 0.76 |
| 4 | 4.92 | 2.74 |
| Max | 15.06 | 14.98 |

Fig. 17. (Colour online) Error between the designated goal points and capture points if circular templates of smaller size are used at the vertices of the object.

to the workspace), and other factors. The second source, as mentioned earlier, is the size of the used template. Templates of smaller sizes lower the error. However, their size cannot be reduced below a certain size. Otherwise, there is a likelihood that the camera loses track of templates and hence the entities within the workspace. A better vision system can help improve the capture procedure using smaller templates. Other sources of error are related to the robot motion, as the two wheels may not rotate at the same speed, slip at the wheels and uneven surface. Robots used herein are non-holonomic in that they cannot turn or move concurrently. They can either turn or move.

All experiments performed here are with low-speed motions of the object, obstacles, and robots to demonstrate the proof of concept. This is done to allow enough time for the overhead camera to capture the circular templates, for the CPU to process the images and analyze the motion sequence for the agents, and relay the commands via wireless modules. Some test experiments performed (Fig. 20) suggest that at high translation or rotation speeds, or in high intensity background, template losses do become more prominent. Frame losses can occur if a robot is rotated with a speed of 0.7 rad/s (40°/s) or more (Fig. 20(a)). Under a moderately lit background, speeds of up to 0.57 rad/s (32.7°/s) may be possible (Fig. 20(b)). Frame losses can also occur if a bright background is used (Fig. 20(c)). Figure 21 suggests that the agents can achieve average translation speeds of as high as 100 mm/s. One way around the template loss is to increase the dimension of the search box. However, care should be taken that the two search boxes do not overlap. With small speed assumption, instantaneous collision (impact) between robot–robot and robot–object is also avoided. Lower rate

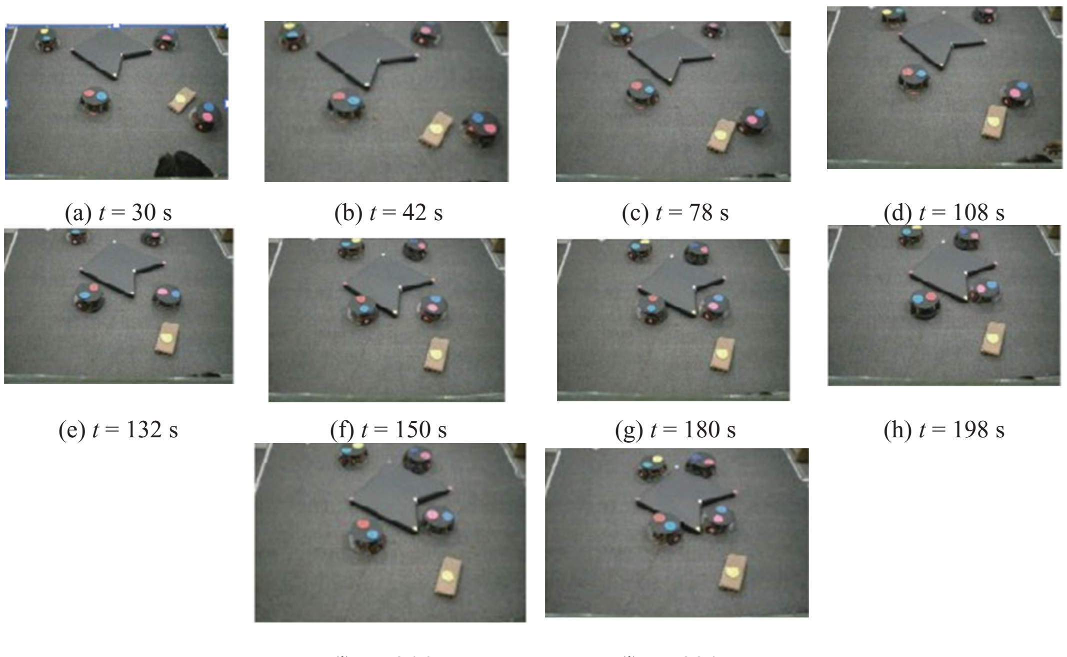

(a) $t = 30$ s (b) $t = 42$ s (c) $t = 78$ s (d) $t = 108$ s
(e) $t = 132$ s (f) $t = 150$ s (g) $t = 180$ s (h) $t = 198$ s
(i) $t = 246$ s (j) $t = 294$ s

Fig. 18. (Colour online) Capture of object 2 with reduced template size. Both obstacle and object move randomly.

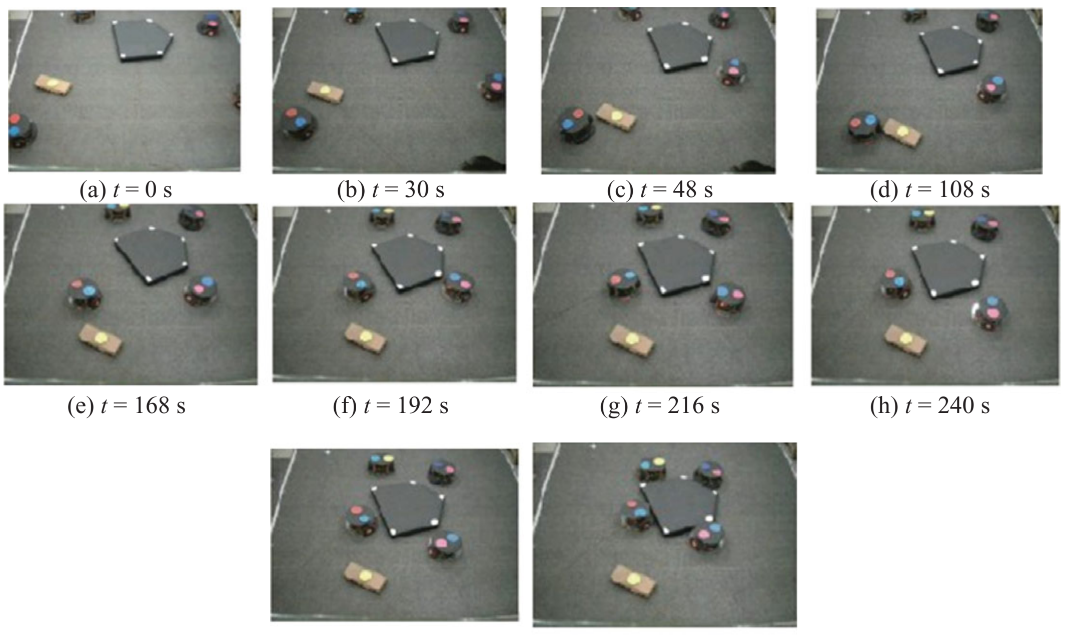


Fig. 19. (Colour online) Capture of object 5 with reduced template size. The object moves from top to the center, while the obstacle moves from left toward the bottom of the workspace.

of frame capture (five frames per second), high processing time because of image capturing/writing, smaller workspace, and light intensities are some factors that impose low-speed restriction on the agents, which is a limitation of the current experimental setup. Because of the aforementioned restriction, experiments to determine actual limiting speeds of the agents and the object for successful capture of the latter could not be performed.

It is expected that during the chase the average speed of the object should be less than the speed of the agents, otherwise the capture will not be accomplished. To illustrate the entire capture process and to quantify the average speeds of object and agents, the example of the capture of object 2 (Fig. 12) is used. Figure 21 depicts the entire process. All unplanned positions of the object are shown using dashed lines. Collision-free robot trajectories are also shown. The final object position is shown using thick lines and those of the agents are shown using the "◇" markers. Figure 22 shows the plot of average speeds of the object and robots. The maximum average speeds attained by agents 1–4 are close to 102, 65, 70 and 104 mm/s, respectively, and that by the object is close to 54 mm/s. The average speed of the object is more or less lower than the speeds of all agents. Only around $t = 100$ s does the average speed of the object exceeds marginally than those of robots 2 and 3. Between $t =$ 200 s and 400 s, all agents surround the object (Fig. 12(f–i)),

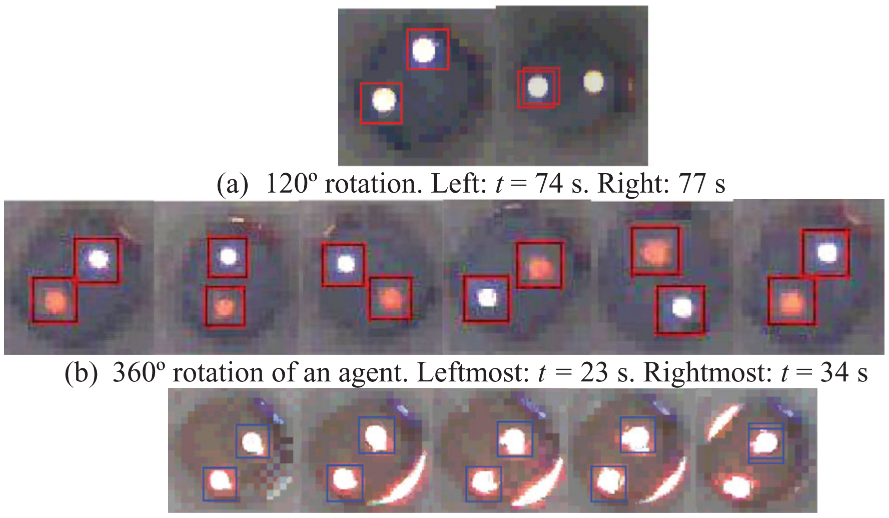

(a) 120º rotation. Left: $t = 74$ s. Right: 77 s
(b) 360º rotation of an agent. Leftmost: $t = 23$ s. Rightmost: $t = 34$ s
(c) Left–right: Increased light intensity

Fig. 20. (Colour online) Instances of template loss. (a) Due to high angular speed, the square box over a circular template is lost, implying that the orientation of the agent can no longer be determined by the CPU. (b) Square boxes remain over their respective circular templates when the agent undergoes low-speed rotation. (c) Observation similar to that in (a) as the intensity of the background light is increased.

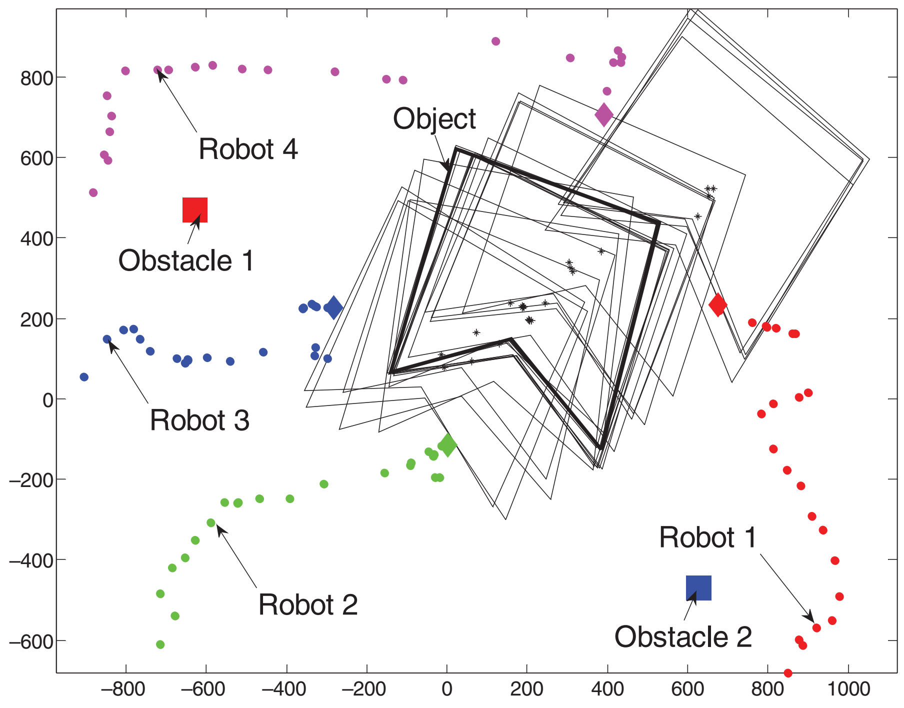


Fig. 21. (Colour online) Capture of object 2 using four agents (Fig. 12). Both obstacles are stationary. Dashed lines show all intermediate positions of the object. Solid lines depict its final position at capture. The "◇" markers show the respective final agent positions.

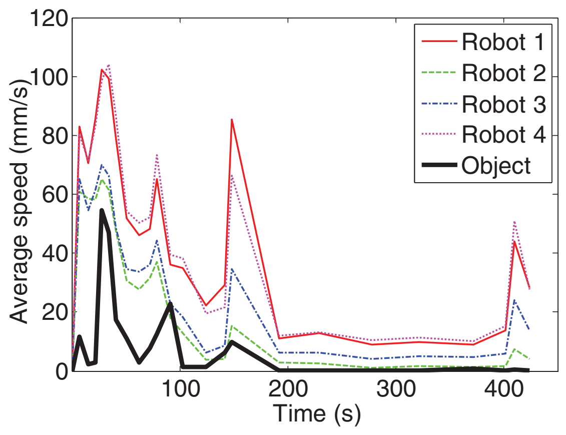


Fig. 22. (Colour online) Average speed of the robots and object during the capture process of object 2 (Fig. 12).

thereby restricting its motion. Hence, the average speed of the object drops to values close to zero. Within this interval, the agents whose average speeds are lower than 20 mm/s, prepare for capture by adjusting their position and orientation angles (Section 3.4). Finally at $t = 424$ s, capture is accomplished.

For a centralized system as the one used here, a better image capturing system, a faster processing unit, and efficient wireless modules will allow for increased agent speeds and help reduce time. A set of overhead cameras may capture the workspace and sequentially send the images to the CPU. A parallel processing system where each CPU individually controls a group of agents (if larger in number) is definitely preferred. Powerful wireless modules will also help reduce time. A significantly more efficient way to reduce the task time is to decentralize the system, and render autonomy and distributed functionality to the agents. The current implementation of the modified projective path-planning algorithm does not use and control the speed information. Controlling the latter will be additionally beneficial in the capture algorithm.

While it was suggested earlier that two robots could transport an object via a sequence of broken serpentine motion, it is felt that better algorithms can be used to push an object across the workspace having dynamic obstacles. One such algorithm is proposed by Sudsang *et al.*[6] for stationary obstacles. It is felt that at least three instead of two robots will be better for transportation applications. Three of the many overall challenges, however, would be to (i) handle dynamic obstacles, (ii) plan transportation in real-time, and (iii) perform capture and transportation on an uneven terrain with speed control on the agents.

**Acknowledgments**

The authors acknowledge Shlok Kumar, Paras Arora, Bishwanath Panda, and Tapas Maji for their help in building the robots and planning the experiments. The third author also acknowledges the support of the Alexander von Humboldt Foundation, and Neelam Jauhari for help in the proof reading of this manuscript.

## References

1. W. Wolfram Burgard, M. Moors, C. Stachniss and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.* **21**(3), 367–378 (2005).
2. C. Belta and V. Kumar, "Abstraction and control for group of robots," *IEEE Trans. Robot.* **20**(5), 865–875 (2004).
3. R. Fierro and A. K. Das, "Hybrid Control of Reconfigurable Robot Formations," **In**: *Proceedings of the American Control Conference*, Denver, Colorado (Jun. 4–6, 2003) pp. 4607–4612.

4. Z. Wang and V. Kumar, "Object closure and Manipulation by Multiple Cooperating Mobile Robots," **In**: *Proceedings of the 2002 IEEE, International Conference on Robotics and Automation*, Washington, DC (2002) pp. 394–399.
5. M. Hashimoto and A. F. Uminoroib, "Dynamic Control Approach for Motion Coordination of Multiple Wheeled Mobile Robots Transporting a Single Object," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*,Yokohama, Japan (Jul. 26–30, 1993).
6. A. Sudsang, F. Rothganger and J. Ponce,"Motion planning for disc-shaped robots pushing a polygonal object in the plane," *IEEE Trans. Robot. Autom.* **18**(4), 550–562 (2002).
7. F. Reulaux, *The Kinematics of Machinery* (Macmillan, Dover, NY, 1876, 1963).
8. H. Hanfusa and H. Asada, "A robot Hand with Elastic Fingers and Its Application to the Assembly Process," **In**: *Proceedings of the Symposium on Informatics and Control Problems in Manufacturing Technology*, Moscow, Russia (June 3–5, 1977) pp. 127–138.
9. J. Salisbury and M. Mason, *Robot Hands and Mechanics of Manipulation*. (MIT Press, Cambridge, MA, 1985).
10. T. Yoshikawa, "Passive and Active Closures by Constraining Mechanisms," **In**: *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, MN, vol. 2. (IEEE Computer Society Press, Washington, DC, 1996) pp. 1477–1484.
11. D. Ding, Y. Liu, J. Zhang and A. Knoll, "Computation of Fingertip Positions for a Form-Closure Grasp," **In**: *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, Seoul, Korea (May 2001), pp. 21–26.
12. E. Rimon and A. Blake, "Caging 2D Bodies by 1-Parameter Two Fingered Gripping Systems," **In**: *Proceedings of the IEEE, International Conference on Robotics and Automation*, Minneapolis, MN, vol. 2, (IEEE Computer Society Press, Washington, DC, 1996) pp. 1458–1464.
13. J. Ponce and B. Faverjon, "On computing three-finger force closure grasp of polygonal objects," *IEEE Trans. Robot. Autom.* **11**(6), 868–881 (1995).
14. H. Asada, "Kinematic analysis of work-part fixturing for flexible assembly with automatic reconfigurable fixtures," *IEEE J. Robot. Autom.* **2**, 86–93 (1985).
15. A. Bicchi and V. Kumar, "Robot Grasping and Contact: A Review," **In**: *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA (Apr. 24–28, 2000) pp. 348–353.
16. M. N. Ahmadabadi and E. Nakano "A 'Constrain and Move' approach to distributed object manipulation," *IEEE Trans. Robot. Autom.* **17**(2), 157–172 (2001).
17. T. G. Sugar and V. Kumar, "Control of cooperating mobile manipulators," *IEEE Trans. Robot. Autom.* **18**(11), 94–103 (2002).
18. Z. Wang and V. Kumar, "Object Closure and Manipulation by Multiple Cooperating Mobile Robots," **In**: *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC (2002) pp. 394–399.
19. P. Song and V. Kumar, "A Potential Field-Based Approach to Multirobot Manipulation," **In**: *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC (May 2002) pp 1217–1222.
20. T. G. Sugar and V. Kumar "Decentralized Control of Cooperating Mobile Manipulators," **In**: *Proceedings of the IEEE, International Conference on Robotics & Automation*, Leuven, Belgium (May 16–20 1998) vol. 4, pp. 2916–2921.
21. R. Grob, M. Bonani, F. Mondada and M. Borigo "Autonomous self-assembly in swarm bots," *IEEE Trans. Robot.* **22**(6), 1115–1130 (2006).
22. M. T. Mason, "Mechanics and planning of manipulator pushing operations," *Int. J. Robot. Res.* **5**(3), 53–71 (1986).
23. M. T. Mason and R. C. Brost "Automatic Grasp Planning: An Operation Space Approach," **In**: *Proceedings of Sixth Symposium on Theory and Practice of Robots and Manipulators*, Cracow, Poland. (: Alma-Press, Berlin, Germany, Sep. 1986), pp. 321–328.
24. M. A. Peshkin and A. C. Sanderson "The motion of a pushed sliding workpiece," *IEEE J. Robot. Autom.* **4**(6), 569–598 (1988).
25. M. A. Peshkin, *Robotic Manipulation Strategies* (Prentice Hall, New Jersey, 1990).
26. M. A. Peshkin and A. C. Sanderson "Planning robotic strategies for work pieces that slide," *IEEE J. Robot. Autom.* **4**(5), 524–531 (1988).
27. S. Yamada and J. Saito, "Adaptive action selection without explicit communication for multi-robot box-pushing," *IEEE Trans. Syst. Man. Cybern. C: Appl. Rev.* **31**(3), 398–404 (2001).
28. Z. Balorda, "Reducing Uncertainty of Objects by Robot Pushing," **In**: *Proceedings of the IEEE Conference Robotics and Automation*, Cincinnati, OH (1990) pp. 1051–1056.
29. P. Sharma, A. Saxena and A. Dutta, "Optimal arrest and guidance of a moving prismatic object using multi-agents," *Robotica* **26**(1), 41–53 (2008).
30. P. Sharma, A. Saxena and A. Dutta, "Obstacle Avoidance, Arrest and Guidance of a Prismatic 2D Object Using Multi Agents," **In**: *Proceedings of the SICE International Conference on Instrumentation, Control and Information Technology*, Kagawa, Japan (2007) pp. 1053–1057.
31. P. Sharma, A. Saxena and A. Dutta, "Multi-Agent Form Closure Capture of a Generic 2D Polygonal Object Based on Projective Path Planning," **In**: *Proceedings of the ASME 2006 International Design Engineering Technical Conferences*, Philadelphia, USA (Sep. 2006) pp. 1–8.
32. M. Gordy, *Matlab Codes, GA: A Matlab Routine for Function Maximization Using Genetic Algorithm*, available at: ftp://all.repec.org/RePEc/cod/html/Matlab/gordy.si (1996).
33. Y. Roger and R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE J. Robot. Autom.* **3**(4), 323–344 (1987).