SINGLE VIEW DEPTH ESTIMATION BASED FORMATION CONTROL OF ROBOTIC SWARMS: IMPLEMENTATION USING REALISTIC ROBOT SIMULATOR

V. GAZI

TOBB University of Economics and Technology, Dept. Electrical and Electronics Engineering, Söğütözü Cad., No: 43, 06560 Ankara, TURKEY.

B. FIDAN

National ICT Australia and the Australian National University Locked Bag 8001, Canberra ACT 2601, Australia.

S. ZHAI

Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, the Netherlands.

In earlier articles we had developed a formation control method based on single view depth estimation. In this paper, we implement that strategy on a robotic swarm composed of non-holonomic agents using the physics based Webots robot simulator. First, we review the single view depth estimation based scheme and the distributed control laws for the agents. Then, we discuss the related modifications due to the non-holonomic constraints of the agents and some related implementation issues and present some simulation results obtained using the proposed control scheme. The scheme is based completely on local information implying that neither global position information nor communication between robots are needed for the implementation of the algorithm.

Keywords: Vision Based Formation Control; Robotic Swarms; Single View Depth Estimation; Non-Sophisticated Camera; Non-Holonomic Constraints.

1. Introduction

In earlier articles,[1,2] we developed a practical scheme based on single view depth estimation for formation motion control of robotic swarms. In this scheme, using a single non-sophisticated camera on the robots (agents) and a priori information about the heights of these robots and objects, the distances (and therefore the relative positions) between the robots are estimated using a single view. By a non-sophisticated camera we mean one

that has limited resolution and limited field of view (FOV), e.g. between $60^o - 90^o$. (Such cameras are very common in the camera market.) The scheme is based completely on local passive 2-dimensional vision information and global position information or communication between robots are not needed for the implementation of the algorithm. In other words, in order to maintain the formation only passive 2-dimensional vision information and local coordinate frames are used within the swarm of robots.

The formation control procedure is a leader based approach based on earlier results in[3,4] in which the estimated inter-individual distances are being used. We consider the problem of moving the robot group from any arbitrary initial position to any arbitrary (but marked with colors) final position without deforming the formation shape during motion. As an important difference from the previous articles here we consider non-holonomic agents and present related modifications of the control algorithm (which was originally developed for agents without velocity constraints). Moreover, in addition to the implementation of the algorithm using a physics based realistic robot simulator (Webots in this case), we also discuss some implementation issues that can be encountered in real problems. We also present simulation results obtained using the above mentioned simulator.

2. Problem Formulation

2.1. *Agent Model*

As a difference from the articles in[1,2] here we consider a swarm consisting of non-holonomic agents with dynamics of agent A_i given by

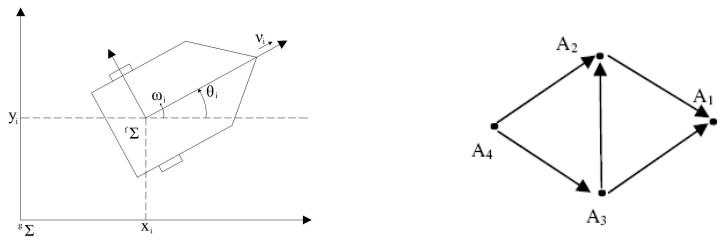
$$\begin{aligned} \dot{x}_i(t) &= \bar{v}_i(t)\cos(\theta_i(t)), \\ \dot{y}_i(t) &= \bar{v}_i(t)\sin(\theta_i(t)), \\ \dot{\theta}_i(t) &= w_i(t) \end{aligned} \tag{1}$$

where $p_i(t) = [x_i(t), y_i(t)]$ represents the Cartesian coordinates with respect to some global coordinate system - see Figure 1(a) - and $\theta_i(t)$ is the steering angle at time t. The control inputs of agent A_i are the linear speed $\bar{v}_i(t)$ and the angular speed $w_i(t)$.

2.2. *Formation Control Problem*

We consider a leader-follower based approach for formation control of a group of robots. It is assumed that in the swarm there is one leader, one first follower (sometimes called co-leader) and many normal followers. Only the leader and the co-leader are assumed to know (or to detect/determine)

the desired final destination of the swarm. The job of the leader is to move towards that final destination. The control algorithm of the co-leader is designed so that it keeps a predefined distance from the leader during motion, and once they are at the final destination it moves around to leader in order to properly re-orient the swarm. The normal followers are required to keep predefined distance from two agents in front all the time in order to preserve the geometric shape of the formation. Figure 1(b) shows four agents which have formed a diamond formation. In the figure agent A_1 is the leader, agent A_2 (which takes as a reference, i.e., keeps a predefined distance to, only agent A_1) is the first follower (the co-leader), whereas agents A_3 (which keeps predefined distances to agents A_1 and A_2) and A_4 (which keeps predefined distances to agents A_2 and A_3) are normal followers.



(a) Unicycle agent model. (b) Diamond formation with four agents.

Fig. 1. Agent model and example formation.

3. Single View Depth Estimation Based Controller

3.1. *Single View Depth Estimation Scheme*

Consider the vision geometry of a convex camera shown in Figure 2(a). The optical center is denoted with point C, the Z-axis denotes the axis parallel to the principal axis, C_f is the focus point on the CCD image plane (which is parallel to the CXY plane), and f is the focal length.[5] Since the actual image on the CCD image plane will be reversed as shown in Figure 2(a), for convenience we consider the image on a fictitious image plane (FIP) as shown in Figure 2(b). The FIP is 180°-rotation of the CCD coordinate frame around the origin and is on the same focal distance f from the camera center C.

Consider a 3-dimensional point $\mathbf{X} = (X, Y, Z) \in \mathbb{R}^3$ which is imaged on

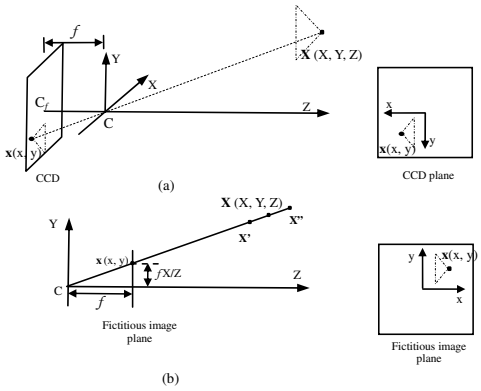


Fig. 2. Illustration of the single view depth estimation scheme.

the FIP at coordinates $\mathbf{x} = (x, y)$. Since digital images are represented as pixels let $\bar{\mathbf{x}} = (\bar{x}, \bar{y})$ denote the coordinates in pixels of the point $\mathbf{x} = (x, y)$. Ignoring the quantization effects, it can be shown that[1,5] the value of $\bar{\mathbf{x}} = (\bar{x}, \bar{y})$ is given by

$$\bar{x} = \frac{1}{Z}\Big(a_x X + x_0 Z\Big), \quad \bar{y} = \frac{1}{Z}\Big(sX + a_y Y + y_0 Z\Big), \tag{2}$$

where $a_x,\ a_y,\ x_0,\ y_0$, and s are camera parameters. Assuming that one pixel has a size (m_x, m_y) and the origin of the image plane is displaced (p_x, p_y) pixels from the point at which the Z-axis intersects the plane, the values of the parameters $a_x,\ a_y,\ x_0$, and y_0 are given by

$$a_x = f m_x, \quad a_y = f m_y, \quad x_0 = p_x m_x, \quad y_0 = p_y m_y.$$

The skew parameter s is nonzero only for non-rectangle pixels.

Assume that all the robots move on (and the objects lie in) the same plane and the robots know the hight of each other (and that of the objects). In particular, assume that the robots have two horizontal feature lines along their upper border and bottom border, respectively (and the same is true for the objects in the environment, if any). Consider a robot with a CCD camera has detected another robot with its camera and denote with $P_1(X, Y_1, Z)$ and $P_2(X, Y_2, Z)$ the coordinates of two points placed on the same vertical line and on the upper and lower feature lines of the observed robot. From equation (2) we have

$$\bar{y}_1 = \frac{1}{Z}\Big(a_y Y_1 + y_0 Z + sX\Big), \quad \bar{y}_2 = \frac{1}{Z}\Big(a_y Y_2 + y_0 Z + sX\Big) \tag{3}$$

where $(\bar{x}, \bar{y}_1), (\bar{x}, \bar{y}_2)$, such that $\bar{y}_1 > \bar{y}_2$, are the pixel coordinates of the images of P_1, and P_2, respectively, on the FIP.

Denoting $\bar{h} = \bar{y}_1 - \bar{y}_2$ and $H = Y_1 - Y_2$ one obtains[1]

$$Z = \frac{a_y H}{\bar{h}},\ X = \frac{a_y H(\bar{x} - x_0)}{a_x \bar{h}},\ d = \sqrt{Z^2 + X^2} = \frac{a_y H}{\bar{h}} \sqrt{1 + \left(\frac{\bar{x} - x_0}{a_x}\right)^2}, \tag{4}$$

where d is the distance between the camera and the target (robot).

3.2. *Control Laws*

In this section we present the expressions of the agent controllers that guarantee that the agents will keep the formation.[1]

Let us define the function $\beta_\epsilon : \mathbb{R}^+ \to \mathbb{R}^+$ as

$$\beta_\epsilon(\delta(t)) = \begin{cases} 0, & \delta(t) < \epsilon, \\ \frac{\delta(t) - \epsilon}{\epsilon}, & \epsilon \leq \delta(t) < 2\epsilon, \\ 1, & \delta(t) \geq 2\epsilon. \end{cases} \tag{5}$$

Then, the control laws for the robots are defined as described below.

Control Law of the Leader: The control law of the leader is defined as

$$v_1(t) = \sigma_1 \bar{v} \beta_{\varepsilon_f} \left(\left\| p_{1f}^{(1)}(t) \right\| \right) p_{1f}^{(1)}(t) / \left\| p_{1f}^{(1)}(t) \right\|, \tag{6}$$

where $p_{1f}^{(1)}(t)$ is the position of the desired final destination point expressed in the frame of the leader, $\bar{v}$ is abound on the agent velocities, $0 < \sigma_1 < 1$ is a small parameter, and ε_f is a small bound on the allowable error between the agent final position and the desired destination.

Control Law of the Co-Leader: We define the control law of the first follower as

$$v_2(t) = \beta_{\varepsilon_k}\left(\left| \bar{\delta}_{12}(t) \right| \right) v_{21}(t) + \left(\sqrt{1 - \beta_{\varepsilon_k}\left(\left| \bar{\delta}_{12}(t) \right| \right)^2} \right) v_{22}(t) + v_1^{(2)}(t) \tag{7}$$

where ε_k denotes a small bound on the allowable tracking error, $v_1^{(2)}(t)$ is the velocity of the leader expressed in the co-leader reference frame, and

$$v_{21}(t) = \bar{v} \operatorname{sgn}\left(\bar{\delta}_{12}(t)\right) p_1^{(2)}(t) / \left\| p_1^{(2)}(t) \right\|,$$

$$v_{22}(t) = \begin{cases} \sigma_2 \bar{v} \delta_{12}^{\perp}(t), & \left| \varphi_{1f}^2 \right| > FOV, \\ \sigma_2 \bar{v} \beta_{\varepsilon_f} \left(\left\| p_{2f}^{(2)}(t) \right\| \right) \operatorname{sgn}\left(p_{2f}^{(2)}(t)^\top \delta_{12}^{\perp}(t) \right) \delta_{12}^{\perp}(t), & \left| \varphi_{1f}^2 \right| \leq FOV. \end{cases}$$

Here $p_1^{(2)}(t)$ and $p_{2f}^{(2)}(t)$ represent the leader position vector and the final desired destination vector (for the co-leader) φ_{1f}^2 is the angle between them all expressed in the local coordinate frame of the co-leader, and

$$\bar{\delta}_{12}(t) = \left\| p_1^{(2)}(t) \right\| - d_{21}, \quad \delta_{12}^{\perp}(t) = \left(-p_{1y}^{(2)}(t), p_{1x}^{(2)}(t) \right) / \left\| p_1^{(2)}(t) \right\|.$$

Control Law of the Normal Followers: The control law for the normal follower A_i $(i = 1, ..., N)$ is defined as

$$v_i(t) = \bar{v}\beta_{\varepsilon_k}\left(\left\| p_{id}^{(i)}(t) \right\| \right) p_{id}^{(i)}(t) / \left| p_{id}^{(i)}(t) \right| + \dot{p}_{id}^{(i)}(t) \tag{8}$$

where $p_{id}^{(i)}(t)$ is the desired point for agent A_i (i.e., the point located at the desired distance to both of its preceding reference agents) expressed in the agent local reference frame.

Note that the above controllers can be directly applied to holonomic point agents of the form considered in.[1] However, for non-holonomic agents with dynamics of the form (1) further development is needed.

3.3. *Modification for Non-Holonomic Agents*

Since the agents obey the non-holonomic unicycle dynamics in (1) and the control inputs are the scalar $\bar{v}_i(t)$ and $w_i(t)$, we cannot directly apply the velocity vectors presented in the above section. In fact, one can see that the velocity of agent A_i satisfies

$$v_i = \begin{bmatrix} v_{xi} \\ v_{yi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \bar{v}_i(t) \begin{bmatrix} \cos(\theta_i(t)) \\ \sin(\theta_i(t)) \end{bmatrix}.$$

Therefore, we set the value of the linear speed control of the agent as

$$\bar{v}_i(t) = \|v_i(t)\| \tag{9}$$

where $v_i(t)$ is the control input calculated using the procedure discussed in the preceding section, i.e., depending on the agent it is calculated using either (6), (7), or (8). Moreover, in order to orient the agent in the direction of $v_i(t)$ we set the angular control as

$$w_i(t) = -\alpha\Big(\theta_i(t) - \theta_{id}(t)\Big) + \dot{\theta}_{id}(t) \tag{10}$$

where $\alpha > 0$ is a proportional gain, and the desired direction of motion $\theta_{id}(t)$ and its time derivative $\dot{\theta}_{id}(t)$ are given by

$$\theta_{id}(t) = \tan^{-1}\left(\frac{v_{yi}(t)}{v_{xi}(t)} \right), \quad \dot{\theta}_{id}(t) = \begin{cases} \frac{\frac{d}{dt}(v_{yi}) \cdot v_{xi} - \frac{d}{dt}(v_{xi}) \cdot v_{yi}}{(v_{xi})^2 + (v_{yi})^2}, & v_i(t) \neq 0, \\ 0, & v_i(t) = 0. \end{cases}$$

Note that here we use the $\tan^{-1}(\cdot)$ function as 4-quadrant tangent inverse function. Using techniques from non-linear control theory (such as for example backstepping, feedback linearization, sliding mode) one might be able to develop more sophisticated (and perhaps more effective) controllers as well. However, developing such controllers is outside the scope of this article.

4. Implementation

In order to implement the formation control procedure discussed in the preceding sections we use the Webots simulator.[a] The software contains a physics based simulation engine which can be used to simulate different type of robots. Moreover, it contains predefined models of several commonly used research robots such as Khepera and e-puck. In this article we chose the model for Khepera III robots[b] since they are also available in our lab (although currently they do not have a properly working camera system). Note that these robots obey the unicycle agent dynamics in equation (1) (also depicted in Figure 1(a)).

As an example implementation we consider the formation shown in Figure 1(b). In order to be able to implement the algorithm, we equipped all the robots with cameras which have 640×480 pixels resolution and 90^o FOV. Moreover, we mounted the cameras on servo motors on top of the robots to allow them to perform a panning motion, i.e., to rotate around the vertical axis of the robot. The reason for such a configuration is that otherwise it is extremely difficult (or even impossible) for the followers to keep the preceding reference robots (in the formation) in the line of sight while simultaneously performing the control objective. To see this note that first of all, for the formation in Figure 1(b) agent A_3 has to look at 60^o to the left in order to be able to have both agents A_1 and A_2 within its field of view. This is not specific to the considered formation and similar situations will arise for follower agents in other (possibly more complex) formations as well. Second and more important issue is that since the robots have velocity constraints (they cannot move in the direction of the axel which connects the two wheels) they need to turn towards their desired direction of motion. Then, when the desired direction of motion is perpendicular to (or at least makes a sufficiently large angle with) the current direction of motion of the robot, while the robot is reorienting itself towards the desired direction

[a]Webots is a product of the Cyberbotics company http://www.cyberbotics.com.
[b]Khepera III robots are product of the K-Team company http://www.k-team.com.

of motion, if its camera is fixed, it looses the line of sight of the preceding (reference) agents. In other words, while trying to satisfy the control objective and to turn to the desired direction of motion, the agent looses line of sight of the agents it is required to follow and therefore becomes unable to measure its distance to them. In order to avoid such situations and to satisfy the control objective while keeping the reference agents in line of sight of the robot we implement a rotating (around the vertical axis) camera (in simulation). Implementing such a system on real robots using a servo motor is straight forward. Then, we control the camera orientation of agent A_i $(i = 1, ..., N)$ using control of the form

$$\dot{\theta}_{ci} = -w_i - \alpha_c \varphi^i_{jk}$$

where w_i is the angular speed for the agent given in equation (10) and $\alpha_c > 0$ is a proportional gain. For the follower agents the value of φ^i_{jk} is given by $\varphi^i_{jk} = \left(\frac{\phi^i_j + \phi^i_k}{2}\right)$ where ϕ^i_j and ϕ^i_k are the relative angles or bearings (with respect to the normal axis of the camera) of the preceding reference agents A_j and A_k expressed in the coordinate system of agent A_i. (For example, in the configuration in Figure 1(b) for $i = 3$ we have $j = 1$ and $k = 2$, whereas for $i = 4$ we have $j = 3$ and $k = 2$.) For the leader $\varphi^1_{jk} = \phi^1_f$ is the angle of the destination point and for the co-leader $\varphi^2_{jk} = \varphi^2_{1f}$ is the angle between the leader and the final destination point for that agent. With such a controller the agents try to keep (both of) their references in line of sight.

In order to make easier the image processing task (and to illustrate only the concept) we place cylinders with predefined hight and having different colors on top of the robots. (Note that such accessories can easily be placed on the actual robots as well.) Moreover, we mark the desired destination points for the leader and the first follower with similar colored cylinders.

Since usually the pixels in a digital image are counted from the upper left corner of the image, the camera parameters x_0 and y_0 here are given by $x_0 = -320$ and $y_0 = +240$ and the y axis is reversed. In other words, while $\bar{x} - 320$ shows the pixel number in the x-direction, $-\bar{y} + 240$ determines the pixel number in the y-direction (of the image frame resulting in negative $\bar{h}$ and negative a_y). Since the pixels in the camera driver of the simulator are square we set $s = 0$ and the values of the other camera parameters as $a_y = -237.5$ and $a_x = 237.5$ (determined determined more or less experimentally by trial and error).

Note also that we set the agent reference frame (for the control objective) so that the origin is located at the position of the robot and the

positive x-axis is parallel to the robot motion plane and normal to the camera image plane and pointing outwards (i.e., it is parallel to the z-axis of the camera frame in Figure 2). Therefore, one should not confuse the $(\bar{x}, \bar{y})$ coordinates corresponding to the pixel coordinates of an image of an agent in the image frame and the (x, y) corresponding to the relative position of that agent in the plane.

Given a robot A_i observing robot A_j, the bearing of A_j in the control frame of A_i is determined by the agent as

$$\phi_j^i = \tan^{-1}\left(\frac{-X}{Z}\right)$$

where X and Z are values calculated in (4).

5. Simulation Results

Figure 3 shows screenshots of the initial and final positions of the agents. The agent configuration and numbering is as in Figure 1(b) (and the leader is marked with red, the co-leader with yellow, agent A_3 with green and no marker is applied to agent A_4). The destination points are the two points away from the initial position of the formation (and the destination of the leader is marked with pink while that for the first follower with blue). The hight of the colored marking cylinders both on the robot and

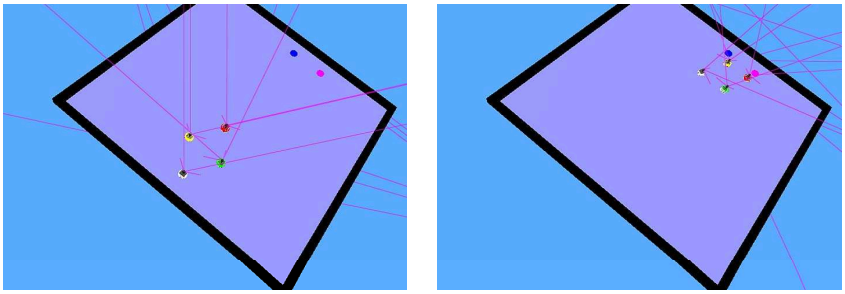


Fig. 3. Screen snapshots from the simulation.

destination points are chosen as $H = 0.03m$. The controller parameters for the simulations were set as $\bar{v} = 0.2$, $\sigma_1 = \sigma_2 = 0.5$, $\alpha = 10$ for the leader and co-leader and $\alpha = 2$ for the normal followers, and $\alpha_c = 1$ for all the robots. The desired distance between neighboring robots is chosen as $d = 0.5m$, while the small bounds on the allowable errors in the formation were set

as $\epsilon_f = 0.01$ and $\epsilon_f = 0.04$. Once the control inputs v_i and w_i for a given agent A_i are computed with a simple transformation they are converted to the right wheel speed and left wheel speed (which are the actual inputs to the robot) using $w_{Ri} = (v_i + Lw_i)/R$ and $w_{Li} = (v_i - Lw_i)/R$, where $R = 0.02215m$ is the radius of the robot wheels and $L = 0.09156/2m$ is the half of the distance between the two wheels.

Another issue to note here is that the controllers in equations (7) and (8) it is assumed that the agents know the velocity of their leader or the reference point. However, since the agents are not moving very fast and in the simulations we perform an update every $64ms$ (leading to insignificant change or no change in the hight of the viewed image) this term is small (although some component can be computed using the own velocity of the robot) and was neglected in the implementations. This might have introduced small error, however the procedure still performed satisfactory enough.

Figure 4 shows the inter-agent distances and the average (approximate) formation error with respect to time. As one can see from the plot of the

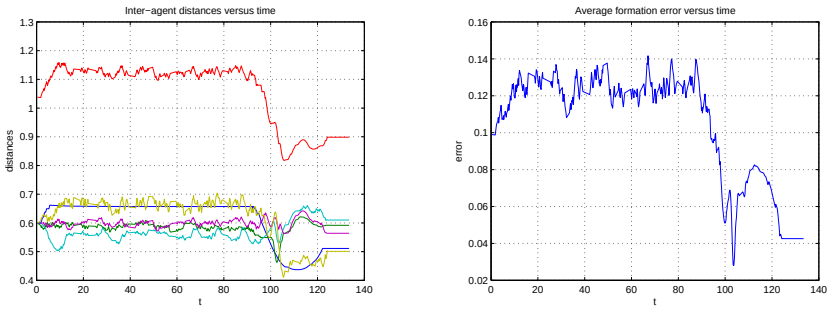


Fig. 4. Inter-agent distances and average formation error versus time.

distances, they are steady during the simulation run. Around $t = 90$'th time step there is a little deformation in the formation which occurs during the rotational motion of the formation around the leader which occurs short after it (the leader) arrives at the final destination (but the co-leader has not arrived at its desired destination yet). One reason for the small deformation is that we implemented the rule in equation (7) in a little different manner using "if-then" type statement and therefore a switching controller. The larger inter-agent distance during motion of the formation

(compared the these at the final destination) are, as mentioned before, due to the fact that we ignored the velocity terms in equations (7) and (8) and also set the distance between the desired final destinations for the leader and the co-leader little smaller than the desired distance between them in the formation.

6. Concluding Remarks

In this article we presented an implementation of a single view depth estimation based formation control scheme for a swarm of non-holonomic robots. We performed the implementation using a physics based simulator and discussed related implementation issues. Implementation of the method on real robots and its extension to other contexts can be a future direction of research.

Acknowledgments

The work of V. Gazi was supported in part by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under grant No. 104E170 and by the European Commission under the GUARDIANS project (FP6 contract No. 045269).

The work of B. Fidan was supported by NICTA. NICTA is funded by the Australian Government as represented by the Dept. of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. S. Zhai and B. Fidan, Single view depth estimation based formation control of robotic swarms: Fundamental design and analysis, in *Proc. 16th Mediterranean Conf. Control and Automation*, (Corsica, France, 2008).
2. S. Zhai, B. Fidan, S. C. Ozturk and V. Gazi, Single view depth estimation based formation control for robotic swarms: Obstacle avoidance, simulation and practical issues, in *Proc. 16th Mediterranean Conf. Control and Automation*, (Corsica, France, 2008).
3. B. Fidan, B. D. O.Anderson, C. Yu and J. M. Hendrickx, *Modeling and Control of Complex Systems* (Taylor and Francis, 2007), ch. Persistent autonomous formations and cohesive motion control, pp. 247–275.
4. S. Sandeep, B. Fidan and C. Yu, Decentralized cohesive motion control of multi-agent formations, in *Proc. 14th Mediterranean Conference on Control and Automation*, June 2006.
5. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2'nd edn. (Cambridge University Press, 2003). pp. 153-157, 312-320.