Magnetic-field-inspired Navigation for Quadcopter Robot in Unknown Environments*

A. Ataka[1,2], H.K. Lam[1], and K. Althoefer[2]

Abstract— In this paper, a magnetic-field-inspired robot navigation is used to navigate an under-actuated quad-copter towards the desired position amidst previously-unknown arbitrary-shaped convex obstacles. Taking inspiration from the phenomena of magnetic field interaction with charged particles observed in nature, the algorithm outperforms previous reactive navigation algorithms for flying robots found in the literature as it is able to reactively generate motion commands relying only on a local sensory information without prior knowledge of the obstacles' shape or location and without getting trapped in local minima configurations. The application of the algorithm in a dynamic model of quadcopter system and in the realistic model of the commercial AscTec Pelican micro-aerial vehicle confirm the superior performance of the algorithm.

I. INTRODUCTION

In the last decade, the advancing capability of micropro-cessors and various sensory devices have equipped robots with powerful tools to detect and perceive previously-unknown environments in real-time. One of the most popular robotic solution in today's world is a micro-aerial vehicle (MAV) due to its hardware simplicity yet powerful software capability. Equipped with onboard sensors, the micro-aerial vehicle has been used for new applications which were previously unfeasible, ranging from parcel delivery, aerial photography, to remote sensing [1]. These new applications require the robot to be equipped with a navigation technique to navigate the robot towards the goal and avoid obstacle without prior information of the environment.

One possible solution to this problem is for the MAV to use simultaneous localization and mapping (SLAM) tech-nique to build the map and infer its position in the map while moving in the environment, exploiting the onboard sensors data [2]. The produced map can then be exploited in the robot navigation stage, such as reported in [3]-[5]. This approach, however, is only favourable for a certain type of environment in which rapid, on-line map building is possible, such as an indoor environment consisting of many unique landmarks.

The onboard sensors can also be exploited to help navigating the robot without specifically building the en-vironmental map. In [6], the authors developed an on-line re-planning technique to help navigating the MAV in previously-unknown environments. This method, however, requires the newly-explored region of the environment to be always taken into consideration in the re-planning process. A recent work reported in [7] used the optical flow and stereo vision technique to control the MAV's velocity and perform obstacle avoidance, however, being only suitable for a simple environment scenario.

Several works exploited only the most-updated sensor information to generate the motion command, an approach known as reactive navigation, such as the use of a potential vector field in [8]. This class of method, however, is mostly unable to guarantee that the robot will successfully reach the goal. Some of these techniques can only be used for an environment consisting of obstacles with a specific geomet-rical shape, such as cubes [9] or circles [10]. A gyroscopic vector field, with the ability to steer the robot's movement from collision, was combined with a back-stepping technique to perform obstacle avoidance for quadcopters [11]. Unlike the standard potential field, this type of field is reported to guarantee goal convergence since it does not affect the robot's energy function. This method, however, is limited to cylindrical and spherical obstacles with perfectly-known geometry. Similar methods were employed to perform multi-robot formation [12]-[14]. However, none of these methods tackle environments with arbitrary-shaped obstacles.

In this paper, a reactive robot navigation algorithm inspired by the natural magnetic field phenomena (as has been reported in the authors' previous works [15]-[16]) is used to navigate a micro-aerial vehicle, specifically an under-actuated quadcopter system, towards the desired position in 3D environments while avoiding convex obstacles along the way without getting trapped in local minima configurations. The reactive nature of the algorithm enables the flying robots to exploit the most-updated environmental information gathered by a local sensor to produce a motion command in real-time. No knowledge of the obstacles' shape or position is needed prior to the robot's movement. The obstacle in the environment is not limited to a specific geometry, but rather it can be any convex obstacle with arbitrary shape. To the best of the authors' knowledge, this is the first work which applies the reactive magnetic-field-inspired navigation to navigate an under-actuated flying robot in an unknown environment.

II. INSPIRATION FROM NATURE

The physical phenomenon which motivated the creation of this algorithm is the magnetic field observed in nature.

[1]A. Ataka and H.K. Lam are with The Centre for Robotics Research (CoRe), Department of Informatics, Kings College London, London WC2R 2LS, United Kingdom.

[2]A. Ataka and K. Althoefer are with the Centre for Advanced Robotics @ Queen Mary (ARQ), Faculty of Science and Engineering, Queen Mary University of London, Mile End Road, London E1 4NS, United Kingdom.
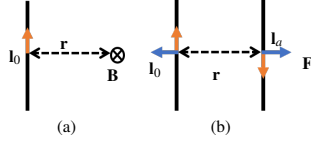
Corresponding author e-mail: ahmad.ataka@kcl.ac.uk

Fig. 1. (a) A magnetic field **B** (pointing inside the paper) is induced by a current-carrying wire. (b) Two parallel wires with currents flow opposite to each other, $\mathbf{l}_a$ and $\mathbf{l}_o$, produce repulsion force **F**.

Fig. 2. (a) The frame definition of the quadcopter system. (b) Tilting the quadcopter along the y-axis produces the x-component of the thrust T which will lead to a horizontal movement.

A magnetic field is observed in the space around a wire in which an electric current flows, as depicted in Fig. 1a. For a very long wire, this magnetic field is given by [17]

$$\mathbf{B} = \frac{\mu_0 i_o}{2\pi r} \frac{\mathbf{l}_o \times \mathbf{r}}{r}, \tag{1}$$

where vector $\mathbf{l}_o$ specifies a unit vector with a direction parallel to the direction of electrical current i_o, μ_0 specifies a permeability constant, $\mathbf{r}$ specifies a position vector of a point in space with respect to the wire, while $\times$ specifies the operation of the vector cross product. When another current-carrying wire, with a length L and current i_a flowing in the direction of unit vector $\mathbf{l}_a$, is present in the close proximity of the first wire, the magnetic field induced by the first wire will affect the flowing electric charge in the second wire. The produced force is described as [17]

$$\mathbf{F} = i_a L \mathbf{l}_a \times \mathbf{B}. \tag{2}$$

From the cross product operation, it can be inferred that the force **F** acting upon the second wire will be directed towards a direction away from the first wire, as illustrated in Fig. 1b. Similarly, a moving charged particle with an electric charge q and velocity **v** will also be influenced by the magnetic field with a similar force described as [17]

$$\mathbf{F} = q\ \mathbf{v} \times \mathbf{B}. \tag{3}$$

Substituting the expression for **B** from (1), and assuming $L = 1$, the force **F** which is acting upon the second wire or moving charged particle can be expressed as

$$\mathbf{F} = \frac{\mu_0 i_a i_o}{2\pi} \frac{\mathbf{l}_a \times (\mathbf{l}_o \times \mathbf{r})}{r^2}, \tag{4}$$

where the term $i_a \mathbf{l}_a$ for the case of wire is equal to the term $q\mathbf{v}$ for the case of moving particle.

Inspired by this concept, a flying robot can be seen as a moving charged particle or a current-carrying wire with velocity direction $\mathbf{l}_a$. To avoid obstacles, this robot will induce an artificial current $\mathbf{l}_o$ on the closest obstacle surface at position $\mathbf{r}_o$ with respect to the robot. This artificial current will then be responsible to induce an artificial magnetic field **B** which affects the movement of the charged particle by a force **F** to steer the robot away from colliding with the obstacle. Recalling that $\mathbf{r}_o = -\mathbf{r}$, the force equation in (4) can be reformulated as follows

$$\mathbf{F} = c\ \mathbf{l}_a \times (\mathbf{r}_o \times \mathbf{l}_o)\ f(|\mathbf{r}_o|). \tag{5}$$

Here, $c > 0$ stands for a constant while $f(|\mathbf{r}_o|) \geq 0$ stands for a scalar function. A skew-symmetric matri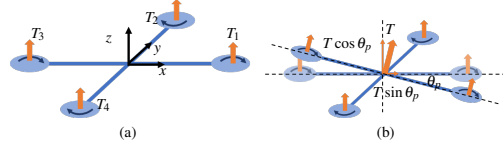x $\hat{\mathbf{l}}$ is introduced as a replacement of the vector cross product operation $\mathbf{l} \times$ for any vector $\mathbf{l} = \begin{bmatrix} l_x & l_y & l_z \end{bmatrix}^T$ as follows

$$\hat{\mathbf{l}} = \begin{bmatrix} 0 & -l_z & l_y \\ l_z & 0 & -l_x \\ -l_y & l_x & 0 \end{bmatrix}. \tag{6}$$

III. Problem Formulation

The robot considered in this paper is an under-actuated micro-aerial vehicle with four symmetrically-placed rotors, famously known as a quadcopter, as depicted in Fig. 2a. Each rotating rotor will generate a thrust T_i which depends on the rotor's speed. The resulting thrust T, produced by the combination of the individual thrusts T_i, is used as a vertical lifting force. To produce the horizontal movement, the quadcopter exploits the pairwise difference of the rotors' rotating speed to produce a torque τ_x or τ_y with respect to its centre of mass which will tilt the robot's roll angle θ_r or pitch angle θ_p respectively, producing a horizontal component of thrust T as illustrated in Fig. 2b. Due to the aerodynamic drag, the rotating rotors produce a reactive torque about the propeller's shaft in the opposite direction of the rotor's rotation. The resulting torques from all rotors τ_z can be used to control the yaw angle θ_y. From this description, it is clear that the system is under-actuated due to the constraint in the thrust direction which can be generated by the robot.

We assume that the robot is located in a bounded workspace $\mathcal{W} \subset \mathbb{R}^3$ whose position is described by a position vector $\mathbf{p} \in \mathcal{W}$ and whose orientation is described by matrix $\mathbf{R} \in SO(3)$. The robot's dynamics is as follows

$$\begin{aligned} \ddot{\mathbf{p}} &= \mathbf{R}\mathbf{e}T + m\mathbf{g}, \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\omega}, \\ \mathbf{J}\dot{\omega} &= \mathbf{J}\hat{\omega}\omega + \tau, \end{aligned} \tag{7}$$

where $\mathbf{e} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is a body-fixed axis which describes the direction of thrust T, m is the robot's mass, $\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.8 \end{bmatrix}^T$ is a gravity vector, ω is a body-fixed angular velocity, $\mathbf{J} \in \mathbb{R}^{3\times3}$ is an inertia matrix, and $\tau = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^T$ is the torque applied to the robot. The thrust T and torque τ are related to the squared of rotors' rotational speed $\mathbf{u}_{sq} = \begin{bmatrix} u_1^2 & u_2^2 & u_3^2 & u_4^2 \end{bmatrix}^T$ as follows

$$\Gamma = \mathbf{D}\mathbf{u}_{sq}, \tag{8}$$

where $\Gamma = \begin{bmatrix} T & \tau \end{bmatrix}^T$ denotes a thrust-torque vector and **D** denotes a constant matrix which depend on the air density and the quadcopter's mechanical property.
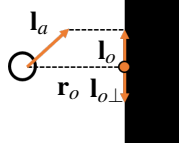
Fig. 3. The artificial current on the obstacle surface $\mathbf{l}_o$ is the projection of the robot's velocity $\mathbf{l}_a$ on to the obstacle surface, while the artificial current $\mathbf{l}_{o\perp}$ is opposite of vector $\mathbf{l}_o$.

In the simulation of the quadcopter's dynamics described in (7), the robot is assumed to be equipped with a local sensor which can detect the surrounding environment inside a sphere of radius r_s centred at the robot's position $\mathbf{p}$. When using the model of the Pelican quadcopter, an RGB-D camera with a limited field of view is placed on the robot. The environment is completely unknown to the robot prior to its movement. Moreover, the robot is also able to measure, in real-time, its position $\mathbf{p}$ with respect to a static global frame, its velocity $\dot{\mathbf{p}}$ (whose direction is defined as $\mathbf{l}_a = \frac{\dot{\mathbf{p}}}{|\dot{\mathbf{p}}|}$), its orientation $\mathbf{R}$ and its body-fixed angular velocity ω.

The environment of the robot consists of $m \in \mathbb{N}$ number of fixed obstacles $\mathcal{O}_i$. Each obstacle $\mathcal{O}_i$ is assumed to be a member of a convex set, either with smooth boundary or non-smooth boundary. The free space is defined as

$$\mathscr{F} = \mathscr{W} \setminus \bigcup_{i=1}^{m} \mathcal{O}_i. \tag{9}$$

The robot starts moving from an initial position $\mathbf{p_s} \in \mathscr{F}$ to the desired goal position $\mathbf{p_g} \in \mathscr{F}$ located at some distance from any of the surfaces of an obstacle.

The goal of the navigation method is to determine the control signal $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T$ which will guide the robot's position $\mathbf{p}(t)$ towards the desired position $\mathbf{p_g}$ as $t \to \infty$ while keeping the robot free from a collision in its entire movement, which is formally defined as

$$\mathbf{p}(t) \in \mathscr{F}, \forall t. \tag{10}$$

IV. NAVIGATION ALGORITHM

A. Obstacle avoidance

The basic algorithm described in this section is similar to the magnetic-field-inspired navigation described in our previous works [15]-[16]. The robot, while moving in the vicinity of the obstacle, will induce an artificial electric current $\mathbf{l}_o$ on the obstacle as a projection of the robot's velocity onto the obstacle surface as depicted in Fig. 3. This can be mathematically expressed as follows

$$\mathbf{l}_o = \mathbf{l}_a - \frac{(\mathbf{l}_a^T \mathbf{r}_o)\mathbf{r}_o}{|\mathbf{r}_o|^2}, \tag{11}$$

in which $\mathbf{l}_a$ refers to a unit vector describing the robot's velocity direction while $\mathbf{r}_o$ refers to the closest distance to the obstacle. The artificial current on the obstacle will in turn produce a force whose property mimics the interaction force between the magnetic field and the moving charged particle described in Section II. To make the robot follows the direction of artificial current $\mathbf{l}_o$, the vector field $\mathbf{F}$ produced

by the artificial current is modified from its counterpart in nature as follows

$$\mathbf{F} = c \ \mathbf{l}_a \times (\mathbf{l}_o \times \mathbf{l}_a) \ f(|\mathbf{r}_o|, |\dot{\mathbf{p}}|). \tag{12}$$

Not only forcing the robot to follow the artificial current $\mathbf{l}_o$, the vector field $\mathbf{F}$ needs to ensure collision avoidance. This is accomplished by setting the scalar function $f(|\mathbf{r}_o|, |\dot{\mathbf{p}}|)$ to be proportional to the robot's velocity $v = |\dot{\mathbf{p}}|$ and inversely proportional to the robot-obstacle distance $r = |\mathbf{r}_o|$ once the obstacle is close enough to the robot as follows

$$f(|\mathbf{r}_o|, |\dot{\mathbf{p}}|) = \begin{cases} \frac{v}{r} & \text{if } r < r_l \\ 0 & \text{if } r \geq r_l \end{cases}, \tag{13}$$

where r_l specifies a limit distance.

The boundary-following vector field $\mathbf{F}$, described in (11)-(13), has several properties which were thoroughly described and proven in our previous works [15]-[16]. These properties can be summarised as follows:

1) Zero work: The vector field $\mathbf{F}$ will leave the robot's speed unaffected, guaranteeing that the kinetic energy of the system remains constant. Thus, a linear combination of the vector field $\mathbf{F}$ with a globally-stable goal attraction does not create local minima configurations.

2) Obstacle Boundary Following: Assuming that the robot's velocity direction $\mathbf{l}_a$ is never in line with the robot-to-obstacle distance vector $\mathbf{r}_o$, the vector field $\mathbf{F}$ is guaranteed to guide the robot towards the direction of the artificial current $\mathbf{l}_o$ which is parallel to the obstacle's boundary.

3) Collision Avoidance: Assuming that the robot's velocity direction $\mathbf{l}_a$ is never in line with the robot-to-obstacle distance vector $\mathbf{r}_o$, the scalar function $f(|\mathbf{r}_o|, |\dot{\mathbf{p}}|)$ in (13) ensures that the robot will never touch the surface of a convex-shaped obstacle.

Although the vector field in (12) ensures that the robot's distance to the convex-shaped obstacle theoretically will never be zero, it does not guarantee a safe distance between the robot and the obstacle surface since there is no component of vector field $\mathbf{F}$ which could repel the robot away from the surface when the robot is too close to the obstacle. This could lead to a problem in a real scenario, where the size of the quadcopter becomes important. Taking inspiration from the behaviour of a pair of current-carrying wires as described in Section II, a collision-avoidance vector field $\mathbf{F_a}$ introduced in our previous work [18] is employed. Besides the artificial current $\mathbf{l}_o$ in (11), the robot also induces another artificial current $\mathbf{l}_{o\perp}$ which is exactly the same as the current in (11), but in the opposite direction as depicted in Fig. 3. The collision-avoidance term $\mathbf{F_a}$ is described as follows

$$\mathbf{F_a} = \mathbf{l}_a \times (\frac{\mathbf{r}_o}{r} \times \mathbf{l}_{o\perp}) \ f(|\mathbf{r}_o|), \tag{14}$$

$$f(|\mathbf{r}_o|) = \begin{cases} c_\perp (\frac{1}{r} - \frac{1}{r_b}) \frac{1}{r^2} & \text{if } r < r_b \\ 0 & \text{if } r \geq r_b \end{cases}, \tag{15}$$

in which $c_\perp > 0$ specifies a constant while $r_b < r_l$ specifies a limit distance. It has been described in our previous work

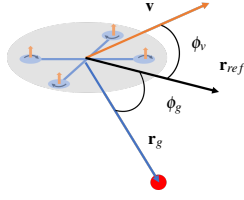Fig. 4. The geometric-control-based goal attraction involves the transformation of the robot's velocity vector $\mathbf{v}$ and distance to the goal $\mathbf{r}_g$ (shown as a red dot) with respect to a static reference vector $\mathbf{r}_{ref}$.

[18] that this avoidance vector field $\mathbf{F_a}$ also does no work. Therefore, the vector field described in (14)-(15) ensures that a repelling force is applied to the robot once it is too close to the obstacle surface without affecting the robot's speed since it works by steering the robot towards a direction away from the obstacle surface. The resultant vector field is given by $\mathbf{F}_o = \mathbf{F} + \mathbf{F_a}$.

B. Navigation towards the goal

A modified version of the geometric control described in our previous work [15] is used to perform navigation towards the goal. With respect to a reference vector $\mathbf{r}_{ref}$, the robot's velocity $\mathbf{v}$ can be expressed as matrix $\mathbf{R}_v \in SO(3)$, as shown in Fig. 4, given by

$$\mathbf{R}_v = \mathbf{I} + \hat{\omega}_v + \hat{\omega}_v^2 \frac{1}{1 + \cos\phi_v}. \tag{16}$$

Here, $\omega_v = \hat{\mathbf{r}}_{ref}\frac{\mathbf{v}}{|\mathbf{v}|}$, $\cos\phi_v = \mathbf{r}_{ref}^T\frac{\mathbf{v}}{|\mathbf{v}|}$, while $\mathbf{I} \in \mathbb{R}^{3\times3}$ stands for an 3×3 identity matrix. Similarly, the robot-to-goal vector $\mathbf{r}_g = \mathbf{p_g} - \mathbf{p}$ can also be expressed with respect to the reference vector $\mathbf{r}_{ref}$ as matrix $\mathbf{R}_g \in SO(3)$. To guide the robot's velocity from the direction specified by orientation matrix $\mathbf{R}_v$ to the direction specified by orientation matrix $\mathbf{R}_g$, the following control law is employed

$$\hat{\omega}_g = -K_\omega \log(\mathbf{R}_e). \tag{17}$$

Here, $\mathbf{R}_e = \mathbf{R}_g^T\mathbf{R}_v$ specifies an error matrix, $K_\omega \geq 0$ specifies a constant. The operation $\log(\mathbf{R})$ for any $\mathbf{R} \in SO(3)$ is defined as $\log(\mathbf{R}) = \frac{\beta}{2\sin\beta}(\mathbf{R} - \mathbf{R}^T)$ in which $\beta = \arccos(\frac{tr(\mathbf{R})-1}{2})$. The angular velocity $\omega_g \in \mathbb{R}^3$ in the frame of robot-to-goal vector $\mathbf{r}_g$ can be retrieved from the skew-symmetric matrix $\hat{\omega}_g$ by the help of eq. (6). The angular velocity ω_g can then be expressed in the static frame $\mathbf{r}_{ref}$ via the following matrix transformation $\omega_{ref} = \mathbf{R}_g\omega_g$. The desired angular velocity can be achieved by this vector field

$$\mathbf{F}_{gc} = \hat{\omega}_{ref}\mathbf{v}. \tag{18}$$

To control the magnitude of the robot's speed, a speed controller is added as follows

$$\mathbf{F}_v = -K_v(v - v_d)\mathbf{l}_a, \tag{19}$$

in which $K_v > 0$ specifies a constant while v_d is the robot's desired speed. Lastly, to make the robot starts moving and to ensure that the robot asymptotically reach the goal, a proportional-derivative (PD) controller is employed when the robot starts moving or when it is close to the goal as follows

$$\mathbf{F}_{pd} = -K_P(\mathbf{p} - \mathbf{p_g}) - K_D\dot{\mathbf{p}}, \tag{20}$$

where K_P and K_D are positive constants. Therefore, the total navigation vector field is as follows

$$\mathbf{F}_n = \begin{cases} \mathbf{F}_o + \mathbf{F}_{gc} + \mathbf{F}_v & \text{if } |\mathbf{r}_g| \geq r_{gl} \text{ and } v \neq 0 \\ \mathbf{F}_o + \mathbf{F}_{pd} & \text{if } |\mathbf{r}_g| < r_{gl} \text{ or } v = 0 \end{cases}, \tag{21}$$

where r_{gl} specifies a limit distance.

V. APPLICATION TO QUADCOPTER

To implement the navigation vector field $\mathbf{F}_n$ in the underactuated quadcopter system, we need to design a nested control algorithm as described in [19]. To produce movement in z direction, the proposed navigation vector field $\mathbf{F}_n$ is combined with a gravity compensation term $\mathbf{G} = m\begin{bmatrix} 0 & 0 & 9.8 \end{bmatrix}^T$ to generate the force $\mathbf{F}_Q = \mathbf{F}_n + \mathbf{G}$. We get the thrust T by performing a dot product between the force $\mathbf{F}_Q$ and a fixed vertical axis $\mathbf{e} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ as $T = \mathbf{F}_Q^T\mathbf{e}$.

To produce a planar movement, the navigation vector field $\mathbf{F}_n$ is projected onto a planar plane and weighted by a constant $K_{Pv} > 0$ to get the planar force $\mathbf{f}$ as follows

$$\mathbf{f} = K_{Pv}\mathbf{F}_{np}, \tag{22}$$

where $\mathbf{F}_{np} = \begin{bmatrix} F_{nx} & F_{ny} \end{bmatrix}^T$. This planar force $\mathbf{f} = \begin{bmatrix} f_x & f_y \end{bmatrix}^T$ can be produced by controlling the roll angle θ_r and the pitch angle θ_p. With the help of the illustration in Fig. 2b, we can infer that the force components in the direction of x-axis and y-axis are related to the pitch angle θ_p and the roll angle θ_r respectively. Assuming a low speed movement which corresponds to a small deflection angle, we get

$$\begin{aligned} f_x &= T\sin\theta_p \approx T\theta_p, \\ f_y &= T\sin\theta_r \approx T\theta_r. \end{aligned} \tag{23}$$

By substituting the value of the force in (22) to (23), we get the desired pitch and roll angles in order to produce the force $\mathbf{f}$ as follows

$$\begin{aligned} \theta_{p,d} &= \frac{K_{Pv}}{T}F_{nx}, \\ \theta_{r,d} &= \frac{K_{Pv}}{T}F_{ny}. \end{aligned} \tag{24}$$

Finally, the PD control is used as torque components τ_x and τ_y to ensure that the desired roll and pitch angle can be reached as follows

$$\begin{aligned} \tau_x &= -K_{P\theta}(\theta_r - \theta_{r,d}) - K_{D\theta}\dot{\theta}_r, \\ \tau_y &= -K_{P\theta}(\theta_p - \theta_{p,d}) - K_{D\theta}\dot{\theta}_p, \end{aligned} \tag{25}$$

where $K_{P\theta}$ and $K_{D\theta}$ specify positive constants. To force a constant yaw angle $\theta_y = 0$, we set the remaining torque component τ_z to be a similar PD control by setting $\theta_{y,d} = 0$. The rate of the angles $\dot{\theta} = \begin{bmatrix} \dot{\theta}_r & \dot{\theta}_p & \dot{\theta}_y \end{bmatrix}^T$ are related to the body-fixed angular velocity ω via a matrix $\mathbf{M}(\theta)$ which can be approximated as an identity matrix for the case of a small angle deflection, i.e. $\dot{\theta} = \mathbf{M}(\theta)\omega \approx \omega$. Finally, we get the rotors's speed $\mathbf{u}$ from the square-root of $\mathbf{u}_{sq}$, calculated from the thrust-torque vector Γ by inverting eq. (8).
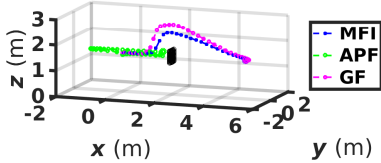
Fig. 5. The plot of the quadcopter's centre of mass trajectory in an environment consisting of a planar obstacle.

Fig. 7. The plot of the quadcopter's centre of mass trajectory in an environment consisting of multiple spherical-shaped obstacles.
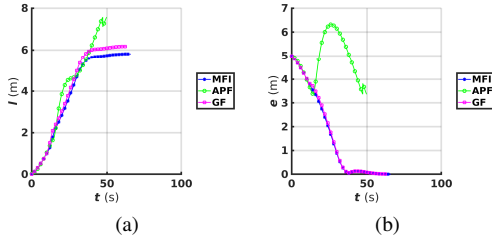

(a)   (b)

Fig. 6. The plot of (a) the quadcopter's trajectory length and (b) the position error for an environment consisting of a planar obstacle.

(a)   (b)

Fig. 8. The plot of (a) the quadcopter's trajectory length and (b) the position error for an environment consisting of multiple spherical-shaped obstacles.

VI. RESULTS AND ANALYSIS

We present the simulation results to evaluate the performance of the magnetic-field-inspired (MFI) navigation applied to the under-actuated quadcopter system. In the first part, the proposed algorithm is implemented to the dynamics model of the quadcopter as described in (7). In the second part, the proposed algorithm is used to navigate the realistic model of AscTec Pelican quadcopter in the Gazebo simulator including the use of its depth sensor with a limited field of view to sense the environment. We compare the performance of the proposed algorithm with several previous works, including the artificial potential field (APF) [20] and the gyroscopic force (GF) [13]-[14]. These algorithms are chosen since they fall under the category of reactive navigation and have the capability to be used in an environment with arbitrary-shaped obstacles. All the codes are implemented in the framework of the Robot Operating System (ROS) [21].

A. Simulation with the quadcopter's dynamic model

To test the performance of the algorithm, we start with a simple environment consisting of a planar obstacle. The trajectories of the quadcopter's centre of mass in this environment is shown in Fig. 5. It can be observed that the proposed MFI navigation algorithm is able to smoothly guide the quadcopter towards the goal without getting trapped in a local minimum configuration and, at the same time, maintain a collision avoidance with the obstacle. This is not the case for the APF method which fails to make the robot reaches the goal due to the cancellation of the goal attraction by the repulsive vector field from the obstacle despite achieving collision avoidance. The GF method performs better in successfully guiding the quadcopter towards the goal and achieve collision avoidance. In doing so, however, the GF forces the quadcopter to cover a longer trajectory in circumnavigating the obstacle as shown in Fig. 5. This
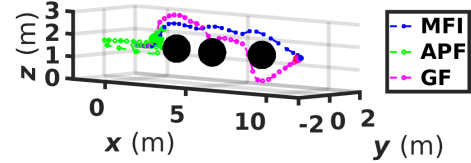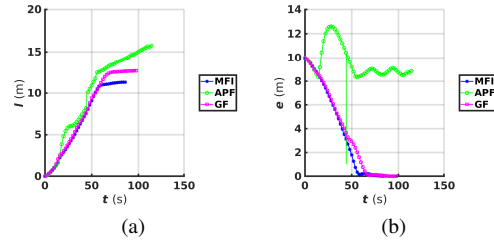
is mainly caused by the repulsive term used by the GF to help avoiding the robot to be too close to the obstacle which pushes the robot away from the obstacle. The MFI algorithm, on the other hand, is equipped with a collision-avoidance vector field which keeps the robot from getting too close to the surface by steering the quadcopter's direction of motion rather than pushing it away. This helps to avoid the quadcopter to be propelled too far from its initial trajectory.

To better compare the performance of the MFI and GF methods, we plot the length of the path covered by the quadcopter in Fig. 6a and the position error of the quadcopter with respect to the goal in Fig. 6b. From Fig. 6b, we can observe that both the MFI and the GF algorithm are able to guide the quadcopter to reach the goal in a similar convergence time. In Fig. 6a, however, it can be observed that the final length of the distance covered by the quadcopter under the guidance of the MFI is less than the the final covered distance when the quadcopter uses the GF method. This confirms our previous analysis which cites the use of a repulsive term in the GF as a cause of the big movement deviation of the quadcopter from its initial path, causing the robot to cover more distance.

In the second environment, we choose three spherical-shaped obstacles placed along the way to the goal. In Fig. 7, we can observe the trajectories covered by the robot under the guidance of the three navigation algorithms. Once again, the APF method fails to successfully guide the robot to reach the goal due to the entrapment in local minima. We can also observe that the MFI and GF are able to successfully navigate the robot towards the goal without colliding with the obstacles. Similar to the performance observed in the previous environment, the MFI achieves goal convergence and obstacle avoidance with a shorter covered trajectory
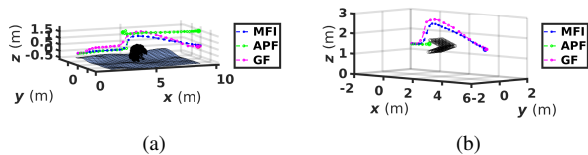
Fig. 9. The plot of the quadcopter's centre of mass trajectory in an environment consisting of (a) a model of a tree and (b) a sharp corner obstacle.

Fig. 11. The plot of the Pelican's centre of mass trajectory in an environment consisting of (a) a cylinder and (b) a cube from two different perspectives.
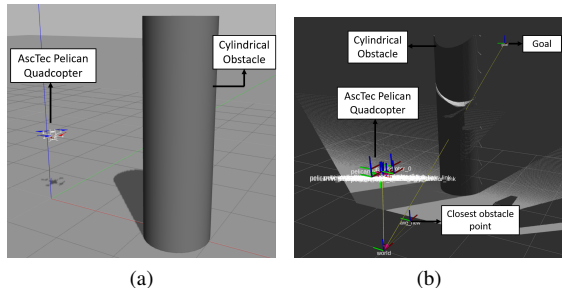

Fig. 10. (a) The simulation with the realistic model of Pelican quadcopter in Gazebo. (b) The R-Viz simulator shows how the quadcopter senses the environment using the RGB-D sensor.

compared to the GF. This can also be observed in the plot of the path's length shown in Fig. 8a. Moreover, Fig. 8b shows that the MFI is able to navigate the robot towards the goal in a shorter convergence time than the GF.

To further demonstrate the suitability of the algorithm to be used in an arbitrary-shaped obstacle, we show 2 more tested environments in Fig. 9 where the quadcopter is placed in an environment consisting of a realistic model of a tree (Fig. 9a) and even a sharp-cornered obstacle (Fig. 9b). Similar performance can be observed in these scenarios. Here, we can conclude that the proposed MFI method produces superior performance in terms of its ability to successfully achieve collision avoidance and goal convergence for the quadcopter with a short trajectory and fast convergence time in various environments including the one with arbitrary-shaped obstacles.

B. Simulation with the model of Pelican quadcopter

In the second simulation scenario, the MFI method was applied to the realistic model of the AscTec Pelican quadcopter in Gazebo simulator as shown in Fig. 10a with an onboard RGB-D camera as a sensor to detect the surrounding environment. The RGB-D camera with its limited field of view produces point cloud data of the local environment which can be processed to get the closest distance to the surrounding obstacle as shown in Fig. 10b. Relying only on this information, the MFI algorithm produces the control command which is fed to the rotors to produce the desired movement towards the goal.

Fig. 11 shows the plot of the Pelican's trajectory in the tested environments consisting of a cylinder (Fig. 11a) and a cube (Fig. 11b) under the guidance of the proposed MFI method. It can be observed that the proposed method is able
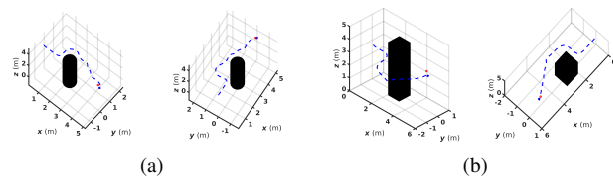
to smoothly steer the robot's movement away from colliding with the closest obstacle surface and reach the goal position without getting trapped in a local minimum. This performance is achieved despite the Pelican does not have access to the environment, including the geometry and location of the obstacles other than the goal position prior to the robot's movement, but rather exploits the local information retrieved from an onboard camera with a limited field of view. Due to the sensor's limitation, it is noted, however, that for a more difficult scenario, we may need to also adjust the quadcopter's yaw angle in order to better perceive the surrounding environment and produce the movement accordingly.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, a reactive robot navigation inspired by the natural phenomena of a magnetic field is used to guide an under-actuated quadcopter robot towards the goal in an unknown environment consisting of arbitrary-shaped convex obstacles. Mimicking the movement of an electric particle under the influence of a magnetic field, the robot is assumed to induce an artificial electric current on the closest obstacle point which will in turn steer the robot away from head-on collisions and follow the obstacle's boundary instead. Compared to similar algorithms in the literature, the proposed navigation is able to reactively generate motion commands relying only on local sensory information without prior knowledge of obstacle shapes or locations and without getting trapped in local minima. The simulation results, using the dynamic model of the quadcopter and the realistic model of the commercial AscTec Pelican micro-aerial vehicle validated the superior performance of our algorithm. In the future, the implementation of the algorithm in a more challenging scenario, possibly in an environment consisting of highly non-convex and even dynamic environments will be further investigated. An implementation to control the movement of multi-agent quadcopter systems will also be explored.

REFERENCES

[1] H. Gonzlez-Jorge, J. Martnez-Snchez, M. Bueno, Arias, and Pedor, "Unmanned Aerial Systems for Civil Applications: A Review," *Drones*, vol. 1, no. 1, 2017. [Online]. Available: http://www.mdpi.com/2504-446X/1/1/2

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[3] Y. Lu, Z. Xue, G.-S. Xia, and L. Zhang, "A survey on vision-based UAV navigation," *Geo-spatial Information Science*, vol. 21, no. 1, pp. 21–32, 2018. [Online]. Available: https://doi.org/10.1080/10095020.2017.1420509

[4] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 2643–2648.

[5] S. Grzonka, G. Grisetti, and W. Burgard, "A Fully Autonomous Indoor Quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, Feb. 2012.

[6] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 2452–2458.

[7] K. McGuire, G. d. Croon, C. D. Wagter, K. Tuyls, and H. Kappen, "Efficient Optical Flow and Stereo Vision for Velocity Estimation and Obstacle Avoidance on an Autonomous Pocket Drone," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1070–1076, Apr. 2017.

[8] A. A. A. Rizqi, A. I. Cahyadi, and T. B. Adji, "Path planning and formation control via potential function for UAV Quadrotor," in *Proc. Int. Conf. Adv. Robot. Intell. Syst.*, Jun. 2014, pp. 165–170.

[9] K. Chang, Y. Xia, K. Huang, and D. Ma, "Obstacle avoidance and active disturbance rejection control for a quadrotor," *Neurocomputing*, vol. 190, pp. 60 – 69, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231216000801

[10] H. Chen, K. Chang, and C. S. Agate, "UAV Path Planning with Tangent-plus-Lyapunov Vector Field Guidance and Obstacle Avoidance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 840–856, Apr. 2013.

[11] G. Garimella, M. Sheckells, and M. Kobilarov, "A stabilizing gyroscopic obstacle avoidance controller for underactuated systems," in *Proc. IEEE Conf. Decision Control*, Dec. 2016, pp. 5010–5016.

[12] E. W. Justh and P. Krishnaprasad, "Equilibria and steering laws for planar formations," *Syst. & control Lett.*, vol. 52, no. 1, pp. 25–38, 2004.

[13] L. Sabattini, C. Secchi, and C. Fantuzzi, "Collision avoidance using gyroscopic forces for cooperative Lagrangian dynamical systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 953–958.

[14] ——, "Collision avoidance for multiple Lagrangian dynamical systems with gyroscopic forces," *Int. J. Adv. Rob. Syst.*, vol. 14, no. 1, p. 1729881416687109, 2017. [Online]. Available: http://dx.doi.org/10.1177/1729881416687109

[15] A. Ataka, H. Lam, and K. Althoefer, "Reactive Magnetic-Field-Inspired Navigation for Non-Holonomic Mobile Robots in Unknown Environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6983–6988.

[16] ——, "Reactive Magnetic-Field-Inspired Navigation Method for Robots in Unknown Convex 3-D Environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3583–3590, Oct. 2018.

[17] D. Halliday, R. Resnick, and J. Walker, *Fundamentals of Physics Extended, 10th Edition*. Wiley, 2013. [Online]. Available: https://books.google.co.uk/books?id=DTccAAAAQBAJ

[18] A. Ataka, A. Shiva, H. K. Lam, and K. Althoefer, "Magnetic-Field-Inspired Navigation for Soft Continuum Manipulator," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 168–173.

[19] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 73. [Online]. Available: http://dblp.uni-trier.de/db/series/star/index.html#Corke11

[20] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 500–505.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, 2009, p. 5.