An Acad Bras Cienc (2024) 96(1): e20220822 DOI 10.1590/0001-3765202420220822
Anais da Academia Brasileira de Ciências | *Annals of the Brazilian Academy of Sciences*
Printed ISSN 0001-3765 I Online ISSN 1678-2690
www.scielo.br/aabc | www.fb.com/aabcjournal

ENGINEERING SCIENCES

An Augmented Reality Visualization System for Simulated Multirotor Aerial Vehicles

ÉDER A. DE MOURA, LUIZ CARLOS S. GÓES, ROBERTO GIL A. DA SILVA & ADSON A. DE PAULA

Abstract: Multirotors Aerial Vehicles are special class of Unmanned Aerial Vehicles with many practical applications. The growing demand for this class of aircraft requires tools that speed up their development. Simulated environments have gained increasing importance, as they facilitate testing and prototyping solutions, where virtual environments allow real-time interaction with simulated models, with similar behavior to real systems. More recently, the use of Augmented Reality has allowed an increasing experience of immersion and integration between the virtual world and a real scenario. This work proposes the use of Augmented Reality technology and a simulated model of a multirotor to create an interactive flight environment, aiming to improve the user experience in the analysis of simulated models. For this purpose, a smartphone was adopted as a hardware platform, a game engine is used as a basis for the development of the Augmented Reality application, that represents a numerical simulation of the flight dynamics and the control system of a multirotor, and a game controller is adopted for user interaction. The resulting system demonstrates that Augmented Reality is a viable technology that can be used to increase the possibilities of evaluating simulated systems.

Key words: augmented reality, mav, uav, simulation, quaternion attitude control.

INTRODUCTION

The Multirotor Aerial Vehicles (MAV), which are a special class of Unmanned Aerial Vehicles (UAVs), are rotary-wing aircrafts capable of hover in the air, and vertical take-off and landing (Bresciani 2008, Mahony et al. 2012, Idrissi et al. 2022). Currently, its use is already common in civil and military applications and a growing demand for new developments has required more and better tools for rapid prototyping of new ideas and uses. As detailed in Mairaj et al. (2019), the use of simulation environments is consolidating as a suitable tool for this scenario, since it provides possibilities for rapid evaluation of dynamics behavior and control strategies for different conditions.

Scientific and industrial research has used numerical simulation combined with high fidelity models for analysis of theoretical and practical aspects of dynamical systems (Jalon & Bayo 1994). The growth of computational processing power is allowing the use of real-time simulation of increasingly complex systems and their visualization through Computer Generated Images (CGI). In general, this strategy speeds up the understanding of the behavior of the system (Radu 2014, Khandelwal et al. 2019), minimizes the risk of damage to the device and reduces the total cost of the project.

In the CGI field, the Extended Reality (XR) is term that embraces the Augmented Reality (AR), Mixed Reality (MR), and Virtual Reality (VR) technologies. It is becoming common practice to incorporate

this class of tools for visualization and interaction with simulated systems into the research and development pipeline. The development and use of XR technologies requires the use of CGI aided by specialized hardware to allow the integration and interaction of virtual objects into real-world scenarios (Carmigniani & Furht 2011, Asaad 2021). To distinguish between variants of XR technologies, in VR the user is immersed into an interactive three-dimensional (3D) world, the AR provides an indirect live view of the physical world with 3D objects added to it and the MR is an AR environment where the 3D objects comprehend and interact with the real world.

The AR enhances the user's perception of reality, and it is characterized by three properties (Azuma 1997): i) combines real and virtual; ii) interactive in real-time; and iii) it is registered in 3D. Its long-term development encompasses the use of several strategies for blending virtual objects with the real world. In the early stages, the available computer processing power imposed restrictions to the task of 3D registration (Azuma 1997, Azuma et al. 2001, Billinghurst et al. 2015). These techniques have evolved from approximate formulations or the use of artificial markers to robust computer vision strategies (Hoff et al. 1996, Kato & Billinghurst 1999, Kuppala et al. 2020).

The development of AR-related technologies requires the use of hardware devices suited to the type of user experience desired in conjunction with a software platform capable of integrating signal and image processing with CGI processing tools. The required hardware for the AR experience depends on the application, which can range from a single camera to environments prepared with motion sensors, and specialized accessories (e.g. glasses, game controller and motion capture systems). Consisting of cameras, accelerometer, rate gyro, magnetometer, a high-quality display, a powerful processor unity and, also, encapsulated in a thin and lightweight device, smartphones has become a prominent platform for the development of AR applications. Game engines provide tools for manipulating 3D objects and were created to streamline the creation of games, but they also gained importance in the creation of interactive virtual environments in other areas, such as architecture, entertainment and engineering. More recently, game engines have added tools for developing AR applications (Linowes & Babilinski 2017, Khandelwal et al. 2019, Nowacki & Woda 2020, Asaad 2021), such as Vuforia (PTC 2022), Apple AR (Apple 2022) and Google AR (Google 2022), which is helping to popularize AR applications through the availability of standardized development tools and ease of exporting an application to various hardware platforms (e.g. desktop, mobile and embedded computers).

In this way, combining the possibilities arising from AR tools with the demand for new UAV applications, the scope of this work is to develop an AR visualization system for a simulated MAV, enabling an interactive and real-time user control of the simulation, using a smartphone as a hardware platform and a game engine to support the AR application.

MATERIALS AND METHODS

This section is dedicated to describing the proposed system and begins by presenting an overview of the modules that compose it, describes the notation adopted for the mathematical formulation, describes the MAV Simulation Module in its details and, finally, explains the Module of Augmented Reality and how it is used to present the simulation.

System description

The system is composed by two modules: the MAV simulation module and the AR Visualization Module. The MAV simulation module comprises the MAV dynamics, its attitude and position control, and receive the external game controller commands. The AR Visualization Module maintains the estimation of the user position in the scenario, update the relative pose of the 3D objects, and merges a coherent vision of the virtual elements with the real scenario in a single image to be present to the user.

For the MAV simulation, a MATLAB/Simulink® (version 2020a) simulation environment is performed on a desktop computer with a (soft) real-time execution constraint, to ensure the user experience compatible with the real MAV. Developed with the ARToolkit®, from the Unity 3D® game engine (version 2020.2.1f1), the AR visualization runs on a smartphone. The communication between the two modules is via a WiFi connection, and the gamepad controller is connected to the desktop computer by a 2.4 GHz wireless adapter, as shown in Figure 1a, b.

The smartphone is fixed on top of the game controller and its screen is used to view the AR projection. The AR projection is made on images of the local environment, captured with the rear camera of the smartphone. The game controller establishes wireless communication with the simulation, allowing the user to move freely in the real world.

A reliable communication is established between the simulation module and the AR Visualization Module through a TCP protocol, where a TCP server is started at the AR application (at the smartphone), listening a configurable TCP port.

The hardware utilized for the development and execution of the tests is composed by: a notebook, with an i5 processor and 16 Gb of RAM, running a Windows 11 operating system; a smartphone, model

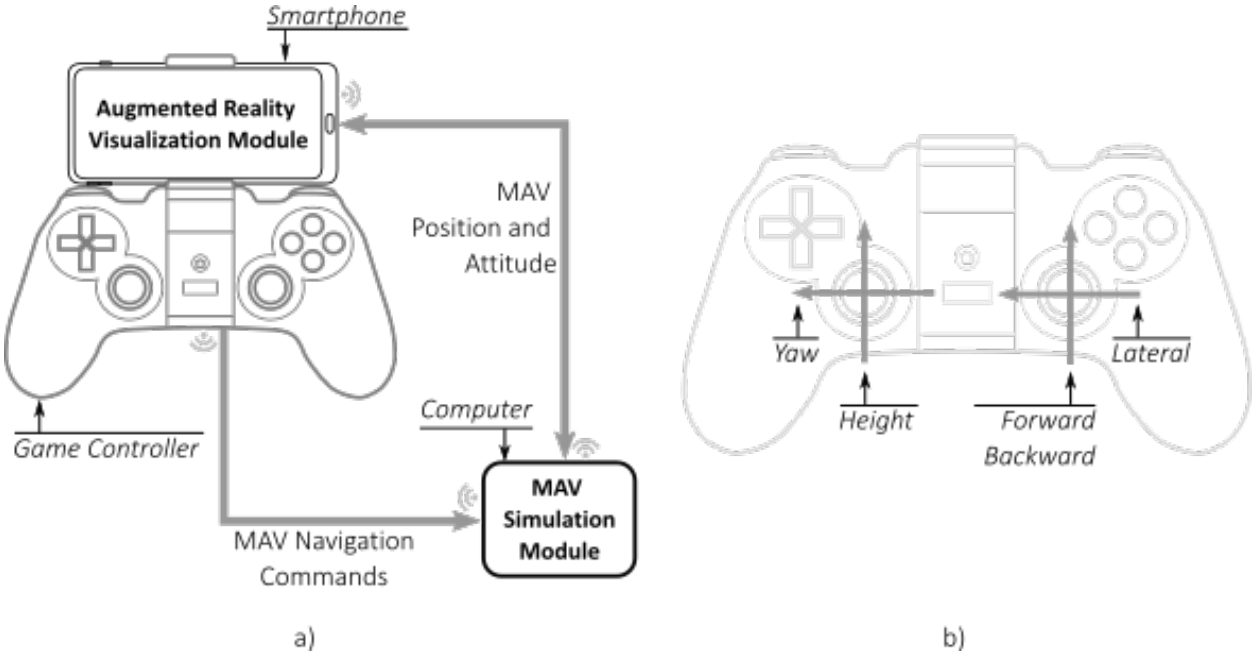

Figure 1. General description of the interaction and control of the AR system. a) Diagram of interaction between the AR visualization module, the simulation module and the game controller. b) Description of MAV navigation commands adopted on the game controller.

Poco X3 NFC, running the Android operating system (version 11); and an Ípega gamepad controller, model PG-9076.

Notation

Defining the previous conventions and notations, scalars are represented by italic letters ($a \in \mathbb{R}$), algebraic column vectors ($\mathbf{a} \in \mathbb{R}^n$) and and matrices ($\mathbf{A} \in \mathbb{R}^{p\times q}$) are represented by bold letters. Transpose matrices are indicated by $\mathbf{A}^{\mathsf{T}}$, and inverse matrices are denoted by $\mathbf{A}^{-1}$. An identity matrix $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is a square matrix with ones on the main diagonal and zeros elsewhere. A Cartesian Coordinate System (CCS) with the origin A is denoted by $\mathcal{S}_{\mathsf{A}}$, whilst a Direction Cosine Matrix (DCM) $\mathbf{D}^{\mathsf{B/A}} \in \mathsf{SO}(3)$ converts a vector projection representation from $\mathcal{S}_{\mathsf{A}}$ to $\mathcal{S}_{\mathsf{B}}$ (Markley & Crassidis 2014, p. 45). The SO (3) is the special orthogonal group of order 3, this implies that $\mathbf{D}^{\mathsf{A/B}} = (\mathbf{D}^{\mathsf{B/A}})^{\mathsf{T}} = (\mathbf{D}^{\mathsf{B/A}})^{-1}$. Furthermore, $\mathbf{q} \triangleq \left[\mathbf{q}_{1:3}^{\mathsf{T}}\ q_4\right]^{\mathsf{T}} \in \mathbb{R}^4$ is a quaternion of unitary norm adopted for attitude parametrization (Markley & Crassidis 2014, p. 28), for which: $\mathbf{q}^{\mathsf{T}}\mathbf{q} = 1$; $\mathbf{q}_{1:3} \triangleq \left[q_1\ q_2\ q_3\right]^{\mathsf{T}} = \boldsymbol{\varepsilon}\sin(\vartheta/2) \in \mathbb{R}^3$; $q_4 \triangleq \cos(\vartheta/2) \in \mathbb{R}$; being $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ the unitary Euler axis vector, and $\vartheta \in \mathbb{R}$ the Euler angle of rotation (Markley & Crassidis 2014, p. 45). The parametrization of a DCM by $\mathbf{q}$ is defined by:

$$\mathbf{D}^{\mathsf{B/A}}(\mathbf{q}) \triangleq \left(q_4^2 - \|\mathbf{q}_{1:3}\|\right)\mathbf{I}_3 - 2q_4[\mathbf{q}_{1:3}\times] + 2\mathbf{q}_{1:3}\mathbf{q}_{1:3}^{\mathsf{T}}. \tag{1}$$

The product of a pair of quaternions $\mathbf{q}^A$ and $\mathbf{q}^B$ can be represented by the following matrix product (Markley & Crassidis 2014):

$$\mathbf{q}^A \otimes \mathbf{q}^B = \left[\mathbf{q}^A \otimes\right]\mathbf{q}^B = \begin{bmatrix} q_4^A\mathbf{I}_3 - \left[\mathbf{q}_{1:3}^A\times\right] & \mathbf{q}_{1:3}^A \\ -\left(\mathbf{q}_{1:3}^A\right)^{\mathsf{T}} & q_4^A \end{bmatrix}\begin{bmatrix} q_1^B \\ q_2^B \\ q_3^B \\ q_4^B \end{bmatrix}, \tag{2}$$

where the skew-symmetric matrix version $[\mathbf{u}\times]$ of a vector $\mathbf{u} = [u_1\ u_2\ u_3]^{\mathsf{T}} \in \mathbb{R}^3$ is given by:

$$[\mathbf{u}\times] \triangleq \begin{bmatrix} 1 & -u_3 & u_2 \\ u_3 & 1 & -u_1 \\ -u_2 & u_1 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{3}$$

We also present here some properties for manipulating quaternion numbers, required for further development: (P1) $(\mathbf{q}^A \otimes \mathbf{q}^B) \otimes \mathbf{q}^C = \mathbf{q}^A \otimes (\mathbf{q}^B \otimes \mathbf{q}^C)$; (P2) $\mathbf{I}_q \triangleq [0\ 0\ 0\ 1]^{\mathsf{T}}$ is the identity quaternion, for which $\mathbf{I}_q \otimes \mathbf{q} = \mathbf{q} \otimes \mathbf{I}_q = \mathbf{q}$; (P3) $\|\mathbf{q}\| \triangleq \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$ is the quaternion norm; (P4) $\mathbf{q}^* = \left[-\mathbf{q}_{1:3}^{\mathsf{T}}\ q_4\right]^{\mathsf{T}}$ is the conjugated quaternion; and (P5) $\mathbf{q}^{-1} \triangleq \mathbf{q}^*/\|\mathbf{q}\|^2$ is the inverse of any quaternion having nonzero norm, where $\mathbf{q}^{-1} \otimes \mathbf{q} = \mathbf{q} \otimes \mathbf{q}^{-1} = \mathbf{I}_q$. It is also considered the Euler angles vector $\boldsymbol{\alpha} = [\phi\ \theta\ \psi]^{\mathsf{T}} \in \mathbb{R}^3$, in the sequence 1-2-3, to obtain the game controller commands and to interact with the AR Visualization Module. These angles represent, respectively, the roll, pitch and yaw of the MAV and they are converted to the quaternion representation and obtained from a quaternion, by the following relations[1] (Markley & Crassidis 2014):

1 To calculate the values of arctan(·) it is preferable to use numeric functions of the type atan2(numerator, denominator).

$$\mathbf{q} \triangleq \begin{bmatrix} \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \end{bmatrix}, \text{ and} \tag{4}$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \triangleq \begin{bmatrix} \arctan\left(\frac{-2(q_3 q_2 - q_1 q_4)}{-q_1^2 - q_2^2 + q_3^2 + q_4^2}\right) \\ \arcsin(2(q_3 q_1 + q_2 q_4)) \\ \arctan\left(\frac{-2(q_2 q_1 - q_3 q_4)}{q_1^2 - q_2^2 - q_3^2 + q_4^2}\right) \end{bmatrix}. \tag{5}$$

MAV simulation module

The simulation module comprises the simulation of the dynamics and the control system, using the inputs given by the user with the game controller. To evaluate AR Visualization Module an altitude and attitude control strategy is proposed, and a schematic view is presented in Figure 2.

The MAV dynamics modeling adopts two CCSs: $\mathcal{S}_R$ is the reference frame, fixed at the initial position with the *x*-axis pointing forward, *z*-axis pointing upward and *y*-axis completing the right-handed rule; and $\mathcal{S}_B$ is the body-fixed frame, that is positioned at the MAV center of mass and it is initially aligned with $\mathcal{S}_R$. Figure 3a presents the modeled 3D MAV, used in the simulation. All the 3D objects used were modeled with Blender 3D (release 3.1.2). Figure 3b shows the alignment of $\mathcal{S}_B$ with the vehicle body, from a top view perspective.

MAV Dynamics

According with Stevens et al. (2015), the flat-Earth equation of motion for any rigid-body adopts the following simplifications: (i) the Earth frame is an inertial reference frame; (ii) position is measured in a tangent-plan coordinate system; and (iii) the gravity vector is normal to the tangent plane and constant in magnitude. The MAV dynamics is modeled by a set of nonlinear differential equations

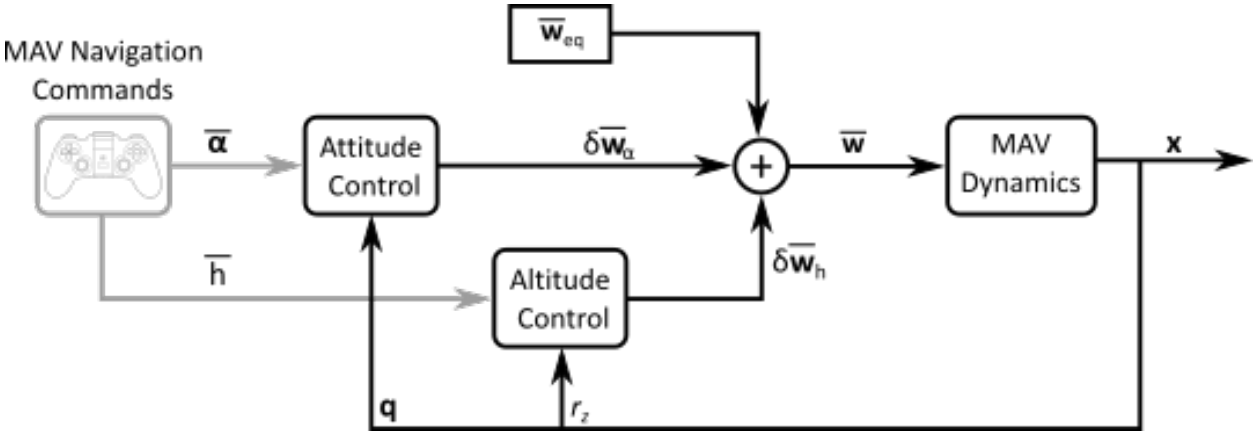

Figure 2. Representation of the MAV simulation module, composed by the simulation of dynamics and control from the commands obtained from the game controller.

and this work adopts the following model (Bresciani 2008, Oliveira 2011, Mahony et al. 2012, Stevens et al. 2015):

$$\begin{bmatrix} \dot{\mathbf{w}} \\ \dot{\mathbf{r}}_R \\ \dot{\mathbf{v}}_B \\ \dot{\mathbf{q}}_B \\ \dot{\boldsymbol{\omega}}_B \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_p}\left(-\mathbf{w} + k_p \bar{\mathbf{w}}\right) \\ \left(\mathbf{D}^{\mathrm{B/R}}(\mathbf{q})\right)^{\mathrm{T}} \mathbf{v}_B \\ \frac{1}{m}\left(\mathbf{F}_B^{\mathrm{C}} + \mathbf{D}^{\mathrm{B/R}}(\mathbf{q})\mathbf{F}_R^{\mathrm{g}} - \mu_d \mathbf{v}_B\right) - [\boldsymbol{\omega}_B \times]\mathbf{v}_B \\ \frac{1}{2}\boldsymbol{\Xi}(\mathbf{q})\,\boldsymbol{\omega}_B \\ \mathbf{J}^{-1}\left(\mathbf{T}_B^{\mathrm{C}} - [\boldsymbol{\omega}_B \times]\mathbf{J}\boldsymbol{\omega}_B\right) \end{bmatrix}, \tag{6}$$

with the state vector and the input vector being defined by:

$$\mathbf{x} \triangleq \left[\mathbf{w}^T \mathbf{r}^T \mathbf{q}^T \boldsymbol{\omega}^T\right]^T \in \mathbb{R}^{14}, \tag{7}$$

$$\mathbf{u} \triangleq \bar{\mathbf{w}} \in \mathbb{R}^4, \tag{8}$$

where $\mathbf{w} \triangleq [w_1\, w_2\, w_3\, w_4]^{\mathrm{T}} \in \mathbb{R}^4$ are the instantaneous rotation speed of the propellers, with $w_j \in [0, w_{max}]$, for $j = 1,\ldots,4$; $\bar{\mathbf{w}} \triangleq [\bar{w}_1\, \bar{w}_2\, \bar{w}_3\, \bar{w}_4]^{\mathrm{T}} \in \mathbb{R}^4$ are the input commands of rotation speed for the propellers; $\mathbf{r}_R \triangleq [r_x\, r_y\, r_z]^{\mathrm{T}} \in \mathbb{R}^3$ and $\mathbf{v}_B \triangleq [v_x\, v_y\, v_z]^{\mathrm{T}} \in \mathbb{R}^3$ are the linear position and velocity of

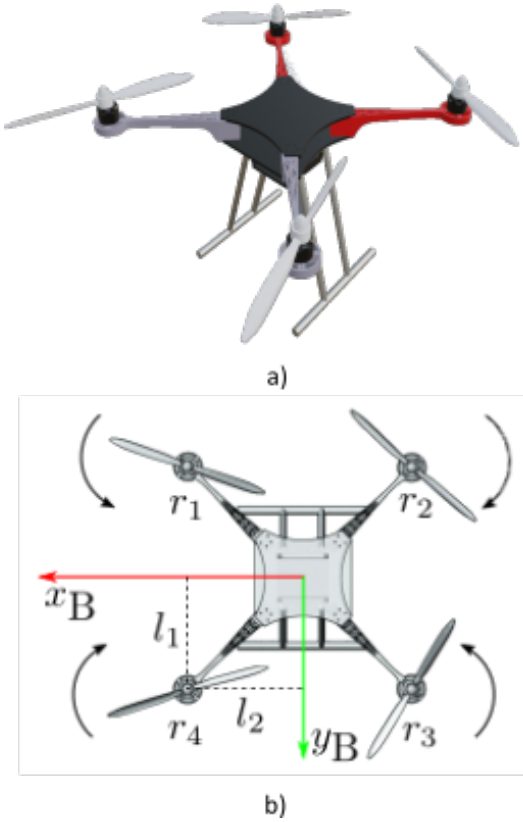

Figure 3. The model of the quadrotor MAV in X symmetrical configuration. a) 3D model used in the AR environment to represent the real drone. b) Top view of the MAV with the representation of the rotors distances for the x and y-axis, and the adopted positive direction of rotation.

$\mathcal{S}_B$ w.r.t. $\mathcal{S}_R$; $\mathbf{q}$ and $\boldsymbol{\omega}_B \triangleq [\omega_x\ \omega_y\ \omega_z]^\intercal \in \mathbb{R}^3$ are the attitude and angular velocity of $\mathcal{S}_B$ w.r.t. $\mathcal{S}_R$. The constants τ_p and k_p are, respectively, the settling time and the proportional gain for the motor and propeller assembly; μ_d is a friction constant due to air drag; $m \in \mathbb{R}$ and $\mathbf{J} \in \mathbb{R}^{3\times3}$ are, respectively, the mass and the rotational inertia matrix, a diagonal matrix defined by:

$$\mathbf{J} = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}. \tag{9}$$

The rotation of the propellers results in thrust forces $\mathbf{f} \triangleq [f_1\, f_2\, f_3\, f_4]^\intercal \in \mathbb{R}^4$ and, due the drag effect, a reaction torque $\mathbf{d} \triangleq [d_1\, d_2\, d_3\, d_4]^\intercal \in \mathbb{R}^4$, where:

$$f_i = k_f w_i^2, \tag{10}$$

$$d_i = (-1)^{(i+1)} k_d w_i^2, \tag{11}$$

for $i = 1,2,3,4$, with k_f and k_d being constants experimentally determined (Vasconez et al. 2016, Piljek et al. 2020). Thus, the modulus of the thrust force $F^C \in \mathbb{R}$, along the z-axis of $\mathcal{S}_B$, and the control torque $\mathbf{T}_B^C \in \mathbb{R}^3$ are obtained by the following relations:

$$F^C = [1\ 1\ 1\ 1]\mathbf{f}, \tag{12}$$

$$\mathbf{T}_R^C = \begin{bmatrix} -l_1 & -l_1 & l_1 & l_1 \\ -l_2 & l_2 & l_2 & -l_2 \\ k_{df} & -k_{df} & k_{df} & -k_{df} \end{bmatrix}\mathbf{f}, \tag{13}$$

where, as shown in Figure 3b, l_1 is the distance of the rotors to $\mathcal{S}_B$'s x-axis; l_2 is the distance of the rotors to $\mathcal{S}_B$'s y-axis; and the reaction torque is given by the fractional constant $k_{df} = \frac{k_d}{k_f}$. The remaining vectors are $\mathbf{F}_B^C \triangleq [0\ 0\ F^C]^\intercal \in \mathbb{R}^3$ and $\mathbf{F}_R^g \triangleq m[0\ 0\ g]^\intercal \in \mathbb{R}^3$, where the last one represents the force due to the action of gravity on the center of mass of the MAV.

The attitude kinematics is defined in terms of the following quaternion dependent matrix:

Table I. Parameters adopted for the MAV simulation.

Parameter	Value	Parameter	Value
m	1.0 $[kg]$	τ_p	3.5×10^{-3} $[s]$
J_{xx} and J_{yy}	0.015 $[kg \cdot m^2]$	k_p	1.0
J_{zz}	0.03 $[kg \cdot m^2]$	μ_d	1.5 $[N \cdot s/m]$
l_1	0.16 $[m]$	k_f	7.3×10^{-6} $[N \cdot rad^2/s^2]$
l_2	0.16 $[m]$	k_d	3.5×10^{-7} $[N \cdot m \cdot rad^2/s^2]$
g	9.81 $[m/s^2]$	w_{max}	10,212 $[rpm]$

$$\mathbf{\Xi}(\mathbf{q}) = \begin{bmatrix} q_4\mathbf{I}_3 + [\mathbf{q}_{1:3}\times] \\ -\mathbf{q}_{1:3}^{\mathsf{T}} \end{bmatrix} \in \mathbb{R}^{4\times 3}. \tag{14}$$

The modeled MAV is based on a UAV built with the F450 frame and Table I contains the parameters adopted in the simulation.

MAV Altitude and Attitude Control System

The altitude and attitude controllers are implemented by a discrete-time approximation of a Proportional, Integral, Derivative (PID) controller, over a linearized operation point, since the system is governed by nonlinear equations. The discrete-time PID controller is formulated by the following equation:

$$u(k) = P \cdot e(k) + I \cdot \sum_{j=0}^{k} T_s \cdot e(j) + D \cdot \frac{e(k) - e(k-1)}{T_s}, \tag{15}$$

where $e(k)$ is the computed error; k represents the kth discrete-time sample; T_s is the sampling time; and P, I and D are tuned constants of the controller. In this work, the gain values of the PID controllers were manually tuned.

The equilibrium point, over which the system is linearized, is given by the MAV in constant position and attitude, with zero linear and angular velocities and with the plane of the propellers parallel to the $x-y$ plane of $\mathcal{S}_R$. For this, all the propellers must have the same rotation speed and the sum of the forces produced must be equal, in module, to the total weight of the aircraft (Bouabdallah et al. 2004, Hussein & Abdallah 2018, Hasseni et al. 2019). Therefore, the total thrust at the equilibrium implies that:

$$\sum_{i=1}^{4} f_i = m \cdot g, \tag{16}$$

where, for equal thrust for each propeller, $f_i = \frac{1}{4} m \cdot g$. Consequently, from Eq. (10), the rotation command vector of the propellers at the equilibrium point is then computed by:

$$\bar{\mathbf{w}}_{\text{eq}} = \sqrt{\frac{m \cdot g}{4k_f}} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^4. \tag{17}$$

The altitude control aims to follow the reference h given by the game controller, as shown in Figure 2. The altitude error is given by:

$$e_h(k) = \bar{h}(k) - r_z(k) \in \mathbb{R}, \tag{18}$$

where $u_h(k)$ is given by Eq. (15), with $P_h = 5$, $I_h = 10$ and $D_h = 50 \times 10^3$, which results in:

$$\delta\bar{\mathbf{w}}_{\text{h}} = u_h(k) \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^4. \tag{19}$$

Attitude control uses the command references given by the game controller $\bar{\boldsymbol{\alpha}}$, as shown in Figure 2. These commands are the user input obtained by manipulating the game controller, shown in Figure 1b, and have the following physical representation: $\bar{\phi}$ implies the lateral movement; $\bar{\theta}$ implies the forward/backward movement; and $\bar{\psi}$ defines the MAV orientation. The MAV lateral and

forward/backward movement commands must be considered in alignment with the vehicle's current orientation.

Calculating the difference between two angles is not simply subtracting two values, as in the case of the difference between altitudes. Operating with differences in attitude representations requires maintaining the properties of the DCM and, in this work, this operation was performed with both values converted to quaternions. Thus, the attitude reference $\bar{\boldsymbol{\alpha}}$ need to be converted to the quaternion representation $\bar{\mathbf{q}}$, with Eq. (4). Taking $\mathbf{q}$ as the actual attitude of the drone, $\bar{\mathbf{q}}$ as the reference attitude to be followed and using the definition given by Eq. (2) with the property (P5), the attitude error in quaternion is obtained by (Younes & Mortari 2019):

$$\mathbf{q}_{\mathbf{e}_\alpha}(k) = \bar{\mathbf{q}}(k) \otimes \mathbf{q}^{-1}(k). \tag{20}$$

The quaternion error $\mathbf{q}_{\mathbf{e}_\alpha}$, given by Eq. (20), is converted to Euler angles representation using Eq. (5), resulting in $\mathbf{e}_\alpha(k) \triangleq \left[e_\phi(k)\ e_\theta(k)\ e_\psi(k)\right]^\mathsf{T} \in \mathbb{R}^3$. The obtained attitude error $\mathbf{e}_\alpha(k)$ is utilized to compute the attitude control vector $\mathbf{u}_\alpha(k) \in \mathbb{R}^3$, using Eq. (15). The attitude control vector $\mathbf{u}_\alpha$ represents the necessary torque to align the MAV according to command $\bar{\boldsymbol{\alpha}}$. It is adopted the following gains for the attitude controller: $P_\alpha^\phi = P_\alpha^\theta = P_\alpha^\psi = 10^4$, $I_\alpha^\phi = I_\alpha^\theta = I_\alpha^\psi = 500$ and $D_\alpha^\phi = D_\alpha^\theta = D_\alpha^\psi = 500$. The force variation $\delta\mathbf{f}$ of each propeller is then obtained by isolating $\mathbf{f}$ in Eq. (13), setting $\mathbf{T}_B^C = \mathbf{u}_\alpha(k)$. Thus, from Eq. (10), the attitude control input is given by:

$$\delta\bar{\mathbf{w}}_\alpha = \frac{1}{\sqrt{k_f}} \begin{bmatrix} \sqrt{\delta f_1} \\ \sqrt{\delta f_2} \\ \sqrt{\delta f_3} \\ \sqrt{\delta f_4} \end{bmatrix} \in \mathbb{R}^4. \tag{21}$$

Finally, using Eq. (17), Eq. (19) and Eq. (21), we can compute the command $\bar{\mathbf{w}}$ by:

$$\bar{\mathbf{w}} = \bar{\mathbf{w}}_{\text{eq}} + \delta\bar{\mathbf{w}}_{\text{h}} + \delta\bar{\mathbf{w}}_\alpha, \tag{22}$$

as shown in Figure 2. Also, the commands $\delta\bar{\mathbf{w}}_{\text{h}}$ and $\delta\bar{\mathbf{w}}_\alpha$ are limited to 30% of w_{max}.

Augmented Reality Visualization Module

The AR visualization module was software developed with the ARToolKit, that is integrated in the Unity 3D game engine, which runs on a smartphone device and promotes real-time integration between the 3D objects and the images captured by the smartphone camera, according to a programmed logic. The smartphone screen works as a camera that lets the user to see the mixed world provided by the AR experience.

The AR technology provided by the ARToolkit has a robust accuracy in the registration of the 3D objects in the real world. This makes it possible for the user to move in the real environment and see the projection, just as it would happen if a real object were present in the environment. In this way,

the user can change the point of view of the 3D objects, registered in the real world, by moving the smartphone in position and orientation. This ability to perceive the relative movement of the user in the real world is the result of the sensory fusion of the signals from the inertial sensors available on the smartphone and the image processing, which uses Artificial Intelligence resources. The user interaction with 3D objects will be done by touching the smartphone screen or manipulating the game controller. All of these features are provided as part of Unity 3D.

The construction of the 3D world within the Unity 3D makes use of the metric system for positioning and measuring the objects, and the attitude is expressed in degrees, using Euler angles. To integrate the simulation with the AR software, the simulation needs to send the MAV position and attitude data to the AR Visualization Module (running at the smartphone), at every instant of time, so that the spatial configuration of the 3D environment is updated. The Unity 3D adopts a left-handed CCS configuration, where *x*-axis pointing forward, *y*-axis pointing upward and *z*-axis completing the left-handed rule. The angles are also computed using the left-handed rule.

The virtual environment uses two CCSs to define the relative positions between the scenery and the 3D objects: 1) The global reference is given by the CCS $\mathcal{S}_O$, which is fixed in the real world and is defined in the position of the smartphone when the AR module starts; 2) The movable CCS $\mathcal{S}_V$ represents the reference position, with which all 3D objects will be related, and can be repositioned by the user's preference. The CCSs $\mathcal{S}_O$ and $\mathcal{S}_V$ are initially aligned and share the same origin, but $\mathcal{S}_V$ can be changed by user action in order to change the relative pose of virtual objects in the real world.

To place the simulation in the virtual environment, it is necessary to establish a relationship between the coordinate systems of the simulation and the virtual environment. In the simulation the MAV pose ($\mathcal{S}_B$) is related to the reference $\mathcal{S}_R$ and, to allow the repositioning of the simulated system in real world, an equivalence between $\mathcal{S}_R$ and $\mathcal{S}_V$ is defined. Nevertheless, the position and orientation of the 3D objects at the AR world must be related to $\mathcal{S}_O$. In fact, as $\mathcal{S}_V$ represents $\mathcal{S}_R$ in the virtual environment, each position or attitude received from the simulation needs to be converted from $\mathcal{S}_V$ to $\mathcal{S}_O$. Thus, the simulated MAV position $\mathbf{r}_R$ is converted to the virtual environment global position by:

$$\mathbf{p}_O = \mathbf{s}_O + \mathbf{D}^{O/V}\mathbf{r}'_V, \tag{23}$$

where $\mathbf{p}_O \triangleq [p_x\, p_y\, p_z]^\mathsf{T} \in \mathbb{R}^3$ is the composed position of the MAV in the AR environment, expressed in $\mathcal{S}_O$; $\mathbf{r}'_V \triangleq [r_x\, r_z\, r_y]^\mathsf{T} \in \mathbb{R}^3$ is the converted MAV simulated position ($\mathbf{r}_R$), expressed in $\mathcal{S}_V$; $\mathbf{s}_O$ is the position of $\mathcal{S}_V$ w.r.t. $\mathcal{S}_O$, and expressed in $\mathcal{S}_O$; $\mathbf{D}^{O/V}$ is the DCM to convert from $\mathcal{S}_V$ to $\mathcal{S}_O$. The $\mathbf{s}_O$ and $\mathbf{D}^{O/V}$ parameters are dynamically computed by the game engine and can be accessed by the application.

The MAV attitude $\boldsymbol{\alpha}$ is converted to the virtual environment global attitude $\boldsymbol{\gamma} \triangleq [\gamma_x\, \gamma_y\, \gamma_z]^\mathsf{T} \in \mathbb{R}^3$ by:

$$\boldsymbol{\gamma} = \boldsymbol{\beta} + \boldsymbol{\alpha}', \tag{23}$$

where $\boldsymbol{\alpha}' \triangleq \frac{180}{\pi}[-\phi\ -\psi\ -\theta]^\mathsf{T} \in \mathbb{R}^3$ is the MAV simulated attitude ($\boldsymbol{\alpha}$), converted to left-hand representantion and expressed in degrees; $\boldsymbol{\beta}$ is the attitude of $\mathcal{S}_V$ w.r.t. $\mathcal{S}_O$, obtained as a property from the game engine. The described vector relationships that result in the global position and attitude are presented in Figure 4.

Summarizing the workflow, the program execution consist in: 1) choose an real environment for the simulation projection; 2) load both applications, the AR visualization module (with the smartphone at a convenient start position) and the MAV simulation module; 3) set the AR visualization port for

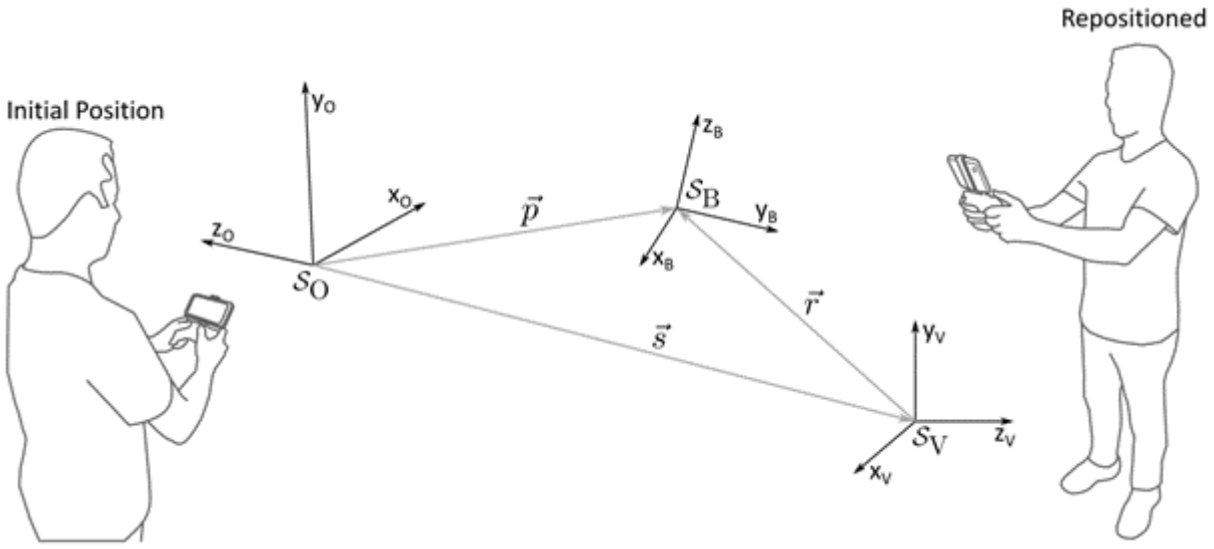

Figure 4. Vector relationship of CCSs for positioning 3D objects, according to the relationship established by the game engine and the simulation system. Illustration of the possibility of user movement in the real world, with respect to $\mathcal{S}_V$ and $\mathcal{S}_O$.

the external connection; 4) set this TCP port in the MAV simulation module; 5) start the simulation; 6) navigate the MAV using the game controller; and, if necessary, 7) adequate the origin $\mathcal{S}_R$ to a convenient position, during the execution, by change the $\mathcal{S}_V$ position.

RESULTS

The evaluation of the AR visualization module consists of analyzing the consistency of the simulation by its virtual representation and the user experience. In the tests carried out in the evaluation, the simulation (numerical integration and control) was adopted with a fixed step of $T_s = 1\,[ms]$.

To evaluate the consistency between the simulated model and the visualization, the first step was to determine the dimensional congruence of the AR environment. In this case, predefined positions and attitude were applied to MAV w.r.t. $\mathcal{S}_R$, and the 3D model response was evaluated. To this end, it was set positions to each axis of: ± 0.5 (m), ± 1.0 (m), and ± 2.0 (m); and attitude around each axis of: ± 30°, ± 45°, ± 90°, ± 180°. The results showed that the AR environment is consistent with the idealized values. However, in more practical situations, using positions that are too far away from the camera's location can cause distortions in the image's perspective. Furthermore, setting positions below the local ground level may imply unrealistic projections.

Figure 5 a to f demonstrates the proposed AR system used in indoor and outdoor environments. Each image is the result of the composition of three elements for the AR system: the real-world scenario, the 3D MAV and a system of coordinate axes, representing the origin $\mathcal{S}_R$ in the simulated system. In general, the simulation and visualization systems presented satisfactory results, concerning to usability, dynamic response and interaction capacity.

The use of this AR system in indoor environments has, in general, better lighting uniformity for the 3D objects, since, in this situation, the light projection tends to have a better distribution. The

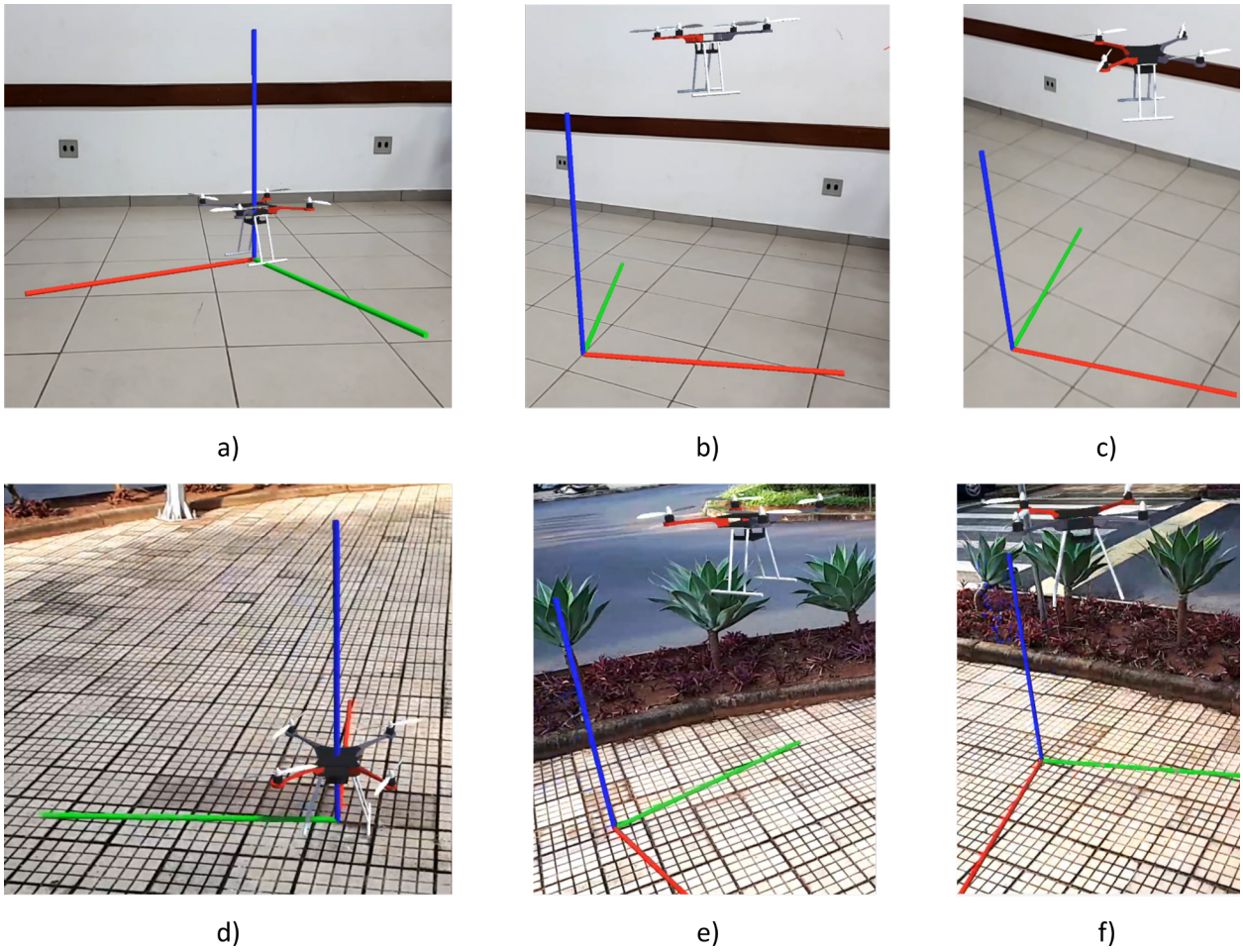

Figure 5. Real environment with 3D objects projected, composing the Augment Reality visualization system. a) Indoor initial position. b) Indoor flight scene. c) Indoor change of scene point of view. d) Outoor initial position. e) Outdoor flight scene. f) Outdoor change of scene point of view.

lighting system in the Unity 3D-ARToolkit estimates the direction and intensity of ambient light to define the lighting system for 3D objects in the scene, so the open environment tends to be more difficult to calibrate and position the lights for the 3D world.

Although the indoor environment has better lighting uniformity, in many cases, it has a smooth texture for the floor, walls and objects, which jeopardizes the tracking process. Estimation of the smartphone motion by the game engine and its ARToolkit depends of the environment texture for the image processing algorithm. A rough texture improves the results and fine-textured environments can cause an uncontrolled drift in the estimated position of the 3D objects in relation to the real world.

For the evaluation of the user experience, it is desirable that it be compatible with the use and of an equivalent real device. Seeing the environment through the smartphone screen gives the user a real sense of integration between the real and the virtual. This capacity has been achieved, especially, in textured environments, where the size and the position of virtual objects registered in the real

world has been shown to be more coherent. With this tool, it is possible, for example, to move around the objects and see them from different points of view.

The communication delay can affect the experience of the simulation. To minimize this effect, the smartphone is configure as a hot spot and the desktop computer connects directly to it.

In general, the proposal met the desired operational requirements and the final system allowed the simulation of a drone in augmented reality, through the interaction between a simulation, which incorporates the physics and electronics of a drone, and a visualization system with the virtual representation integrated into the real environment. The use of augmented reality still has some limitations, however, recent developments in this area have made it possible to use it beyond research laboratories, allowing this technology to be incorporated into various applications, without the need for an environment specially prepared for it.

DISCUSSION

The growth of augmented reality technologies and tools has allowed its use in several areas and has been the focus of investments by several companies. Augmented Reality presents itself as a disruptive interface system, whose flexibility allows combining the possibilities arising from computer graphics with immersion in a real environment. This work combined the use of this tool with the growing demand for drones, providing a visualization application that can be useful in several stages of the design of MAVs.

The proposed application fulfilled the initial objectives, in which an AR system allows the exploration of a simulation, not only from the point of view of numerical evaluation and performance criteria, but also provides an understanding of the system's behavior with an immersive experience in the real world.

The results demonstrated the feasibility of using Augmented Reality tools as a way to increase the possibility of evaluation of MAV projects, based only on simulation. The set of development tools presented by the Unity 3D game engine accelerated the software production process and the availability and quality of the sensors incorporated into smartphones made it an excellent hardware alternative for simpler applications. The combination of both technologies, accessible to the general public at a relatively low cost, allowed the development of an easy-to-use tool to aide the development of applications with MAVs and, also, with possibilities of extension to other areas, such as training of flights with MAVs and educational. In addition, its interactive interface allows the inclusion of non-technical professionals in the development process, providing an experience of use that resembles the real product.

This work can also be expanded and adapted to similar projects. In future versions, it is intended to adapt this project for the use of equipment dedicated to Augmented Reality immersion and with more resources. Comparison of simulated system behavior with a real device is also expected, evaluating both performance and user experience.

Acknowledgments

The author E. A. Moura thanks the financial support of Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), through the doctoral scholarship number 88882.180838/2018-01 from the Programa de Excelência Acadêmica (PROEX).

REFERENCES

APPLE. 2022. IOS - Augmented Reality. Available: Available at: https://www.apple.com/augmented-reality/.

ASAAD RR. 2021. Virtual reality and augmented reality technologies: A closer look. In: Int Res J Sci Tech Educ Mgmt 1: 10. doi.org/10.5281/zenodo.5652296.

AZUMA RT. 1997. A survey of augmented reality. Teleoperators and Virtual Environment. Pres Teleoper Virtual Environ 6(4): 355-385. doi: 10.1162/pres.1997.6.4.355.

AZUMA RT, BAILLOT Y, BEHRINGER R, FEINER S, JULIER S & MACINTYRE B. 2001. Recent advances in augmented reality. IEEE Comput Graph Appl 21(6): 34-47. ISSN: 1558-1756. DOI: 10.1109/38.963459.

BILLINGHURST M, CLARK A & LEE G. 2015. A Survey of Augmented Reality. In: Found Trends Hum-Comput Interact 8(2-3): 73-272. doi: 10.1561/1100000049.

BOUABDALLAH S, NOTH A & SIEGWART R. 2004. PID vs LQ control techniques applied to an indoor micro quadrotor. IEEE/RSJ IROS 2004 3: 2451-2456. doi: 10.1109/iros.2004.1389776.

BRESCIANI T. 2008. Modelling, identification and control of a quadrotor helicopter. Master's thesis, Lund University. ISSN: 0280-5316. Available at: https://lup.lub.lu.se/student-papers/record/8847641/file/8859343.pdf.

CARMIGNIANI J & FURHT B. 2011. Augmented Reality: An Overview. In: FURHT B (Ed), Handbook of Aug Real 3-6. ISBN: 978-1-4614-0064-6. doi.org/10.1007/978-1-4614-0064-6_1.

GOOGLE. 2022. Google AR & VR. Available at: https://arvr.google.com/.

HASSENI SEI, ABDOU L & GLIDA HE. 2019. Parameters tuning of a quadrotor PID controllers by using nature-inspired algorithms. Evolutionary Intelligence 14(1): 61-73. doi: 10.1007/s12065-019-00312-8.

HOFF WA, NGUYEN K & LYON T. 1996. Computer-Vision-Based Registration Techniques for Augmented Reality. Intel Robots and Comp Vision XV: Algorithms, Techniques, Active Vision, and Materials Handling 2904: 538-548. International Society for Optics and Photonics. doi: 10.1117/12.256311.

HUSSEIN A & ABDALLAH R. 2018. Autopilot design for a quadcopter. Khartoum: University Of Khartoum. doi: 10.13140/RG.2.2.10309.91364.

IDRISSI M, SALAMI M & ANNAZ F. 2022. A review of quadrotor unmanned aerial vehicles: Applications, architectural design and control algorithms. J Intell Robot Syst 104: 22. doi: 10.1007/s10846-021-01527-7.

JALON JG & BAYO E. 1994. Kinematic and Dynamic Simulation of Multibody Systems. Springer-Verlag. ISBN: 978-1-4612-7601-2. doi: 10.1007/978-1-4612-2600-0.

KATO H & BILLINGHURST M. 1999. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), p. 85-94. doi: 10.1109/IWAR.1999.803809.

KHANDELWAL P, SRINIVASAN K & ROY SS. 2019. Surgical education using artificial intelligence, augmented reality and machine learning: A review. 2019 IEEE Int Conf Consumer Electr 1-2. doi.org/10.1109/ICCE-TW46550.2019.8991792.

KUPPALA K, BANDA S & BARIGE TR. 2020. An overview of deep learning methods for image registration with focus on feature-based approaches. Inter J Image and Data Fusion. doi: 10.1080/19479832.2019.1707720.

LINOWES J & BABILINSKI K. 2017. Augmented Reality for Developers. Packt Publishing, 548 p. ISBN: 9781787286436.

MAHONY R, KUMAR V & CORKE P. 2012. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE Robot Autom Mag 19(3): 20-32. ISSN: 1558-223X. doi: 10.1109/MRA.2012.2206474.

MAIRAJ A, BABA AI & JAVAID AY. 2019. Application specific drone simulators: Recent advances and challenges. Sim Mod Prac & Theory 94: 100-117. ISSN: 1569-190X. doi: 10.1016/j.simpat.2019.01.004.

MARKLEY FL & CRASSIDIS JL. 2014. Fundamentals of Spacecraft Attitude Determination and Control. Springer New York York, XV, 486 p. ISBN: 978-1-4939-0801-1. doi: 10.1007/978-1-4939-0802-8.

NOWACKI P & WODA M. 2019. Capabilities of ARcore and ARkit platforms for AR/VR applications. Eng Depend Comp Syst & Net 987: 358-370. ISBN 978-3-030-19501-4. doi: 10.1007/978-3-030-19501-4_36.

OLIVEIRA MDLC. 2011. Modeling, identification and control of a quadrotor aircraft. Master's thesis, Czech Technical University in Prague. doi: 10.13140/RG.2.1.1344.2409.

PILJEK P, KOTARSKI D & KRZNAR M. 2020. Method for characterization of a multirotor UAV electric propulsion system. Appl Sci 10(22): 18. doi: 10.3390/app10228229.

PTC - DIGITAL TRANSFORMS PHYSICAL. 2022. Vuforia enterprise augmented reality (ar) software: PTC. Availabe at: http://vuforia.com/.

RADU I. 2014. Augmented reality in education: a meta-review and cross-media analysis. Pers Ubiquitous Comput 18(6): 1533-1543. DOI: 10.1007/s00779-013-0747-y.

STEVENS BL, LEWIS FL & JOHNSON EN. 2015. Aircraft control and simulation. J Wiley & Sons, 3 ed., 749 p. DOI: 10.1002/9781119174882.

VASCONEZ J, GREWAL JP, LEONARDO E, MARWA M, GUZMAN CA & MANKBADI RR. 2016. Propulsion system design for micro aerial vehicles. In: AIAA Atmospheric Flight Mechanics Conference. American Institute of Aeronautics and Astronautics 1: 18. DOI: 10.2514/6.2016-3714.

YOUNES AB & MORTARI D. 2019. Derivation of all attitude error governing equations for attitude filtering and control. Sensors 19(21): 4682. doi.org/10.3390/s19214682.

How to cite

MOURA ÉA, GÓES LCS, SILVA RGA & DE PAULA AA. 2024. An Augmented Reality Visualization System for Simulated Multirotor Aerial Vehicles. An Acad Bras Cienc 96: e20220822. DOI 10.1590/0001-3765202420220822.

Manuscript received on October 04, 2022; accepted for publication on July 09, 2023

ÉDER A. DE MOURA[1]
https://orcid.org/0000-0002-7593-8066

LUIZ CARLOS S. GÓES[1]
https://orcid.org/0000-0003-3770-0601

ROBERTO GIL A. DA SILVA[2]
https://orcid.org/0000-0003-0995-9915

ADSON A. DE PAULA[2]
https://orcid.org/0000-0002-0373-1920

[1]Instituto Tecnológico de Aeronáutica, Departamento de Engenharia Mecânica, Praça Marechal Eduardo Gomes, 50, 12228-900 São José dos Campos, SP, Brazil

[2]Instituto Tecnológico de Aeronáutica, Departamento de Projeto de Aeronaves, Praça Marechal Eduardo Gomes, 50, 12228-900 São José dos Campos, SP, Brazil

Correspondence to: **Éder Alves de Moura**
E-mail: edermoura@ufu.br

Author contributions

Éder Alves de Moura is responsible for the development and execution of the research, as well as for the writing of the article. Luiz Carlos Sandoval Góes, Roberto Gil Annes da Silva and Adson Agrico de Paula are advisor of the research project and are responsible for reviewing the article.