Vision based real-time obstacle avoidance for drones while following a moving target

Francisco Basílio de Aguiar Gomes
franciscobagomes@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

September 2020

Abstract

The problem of this project consists of real-time obstacle avoidance in autonomous flights. It is a relevant topic considering the increase of drone usage in multiple domains such as rescue missions or critical environments explorations and the need to obtain a collision-free system. This work focus on the development of a drone control system using as external sensors a camera and an IMU to ensure a safe flight while following a target. Whereas some traditional approaches focus on full space reconstruction, this work explores a different, reactive and less demanding computational method where it will be used image data to compute the Time-to-Collision (TTC). In a first stage, it was designed a vision strategy where the target is identified, the TTC is computed and the zones in the image with a low TTC are determined. In a second stage, it was implemented a cascade control strategy that uses a Model Predictive Control (MPC) controller to control the drone position and an attitude controller to set up the rotors speed guaranteeing flight stability. The MPC algorithm uses the target information from the vision method to lead the drone to a position that ensures a correct tracking in real-time and, simultaneously, use the TTC data to generate potential fields in the vehicle reference frame ensuring obstacle avoidance. The results in the multiple flights performed in different environments show that the drone can accurately avoid obstacles while tracking a target. However, it was verified that, in certain environments, the developed strategy reveals some limitations that culminate in wrong relative distance estimates threw the obstacle.

Keywords: Drone, Model Predictive Control, Computer Vision, Time-to-Collision, Obstacle Avoidance, Target Tracking.

1. Introduction

In recent years, the world has been evolving in a way that drones are becoming widespread. Navigation algorithms and control theory are the main tools to continuously advance this field of research [1], leading to innovations in multiple domains such as rescue missions [2], powerlines monitoring [3], or critical environments exploration [4]. Drones represent an important aid in various applications while, simultaneously, reduce the human risk possibly associated with a dangerous task.

The motivation for this work comes from the problem of obstacle avoidance, on the risks associated with the environment in an autonomous flight, and on the importance of a flawless system that can ensure new standards in drone usage.

1.1. Problem formulation

The focal point of this project is to develop a real-time obstacle avoidance system for an aerial vehicle (drone) capable of following a target while avoiding obstacles. In order to have a collision-free flight, the time-to-collision is computed leading the drone to avoid the obstacles that emerge by adjusting its controls. With the purpose of obtaining a smooth flight (and maintain the target in sight), using predictive control, while the implemented system aims to achieve a collision-free solution, an optimization problem arises. Onboard sensors provide information on the state of the drone which, together with imagery data, allow the system to solve the Optimal Control Problem (OCP).

1.2. Objectives

According with the main goal of the present work, the following objectives are summarized: (i) Collect and analyze imagery data in real-time, identify the target, calculate the distance to it and compute the TTC. (ii) Develop a control system that generates controls leading to a secure and stable flight, while following a target. (iii) Evaluate the performance of the developed system in a simulation environment.

1.3. Proposed approach

The proposed approach for developing an obstacle avoidance system relies on a strategy that enables

the drone to react to obstacles in a static environment. The system to be implemented will use image information from a monocular camera as a basis for decision-making combined with attitude data provided by an IMU. The inherent problem with the usage of monocular cameras is that it cannot perceive depth from the image, so, from the camera perspective, a small object near the drone moving slowly has the same representation in the image plane as a bigger object moving faster that is far from the drone. The suggested strategy pretends to overcome this problem using features extracted from the image and TTC calculation to infer where the drone must avoid going. After the time-to-collision computation, an attitude controller together with an MPC generate controls to guarantee a stable and secure path for the aircraft while following the target.

1.4. Outline

The remainder of this document is organized as follows: Section 2 presents the related work in the areas of obstacle avoidance, TTC calculation, and predictive control. The experimental setup, strategy and implementation are presented in Section 3. The experimental results and discussion are in Section 4 and conclusions are in Section 5.

2. Related Work

There are a few methods that can be used to perform autonomous flights, however, these differ in some aspects, such as computational power or external data needed. Approaches for solving a collision problem are based on external sensors where, for example, image information is received from a stereo camera allowing to extract depth information [5] or multiple sensor data is fused to detect obstacles [6]. Besides these classical procedures, there are learning proposals, where a controller learns the best action in each particular situation [7].

Guidance methods like SLAM [8] or others that use image analysis like optical flow [9] are examples of strategies that enable the drone to avoid obstacles. Space reconstruction methods with stereo cameras [10] or monocular cameras [11] are used to integrate surrounding information and perform a safe flight. In addition, path planning algorithms [12] in an obstacle avoidance approach are useful where the path is adjusted in the presence of obstacles as in [5].

Time-to-collision is the time left for a collision to happen and can be represented by a value associated to a pixel [13], or with the relative size difference of an object in consecutive frames [14]. The authors in [14] show that with this approach is possible to detect obstacles and avoid collisions.

In [15] is represented two Model Predictive Control strategies capable of generating trajectories, by solving an optimization problem, and iteratively improving the solution during the runtime of the process. Alternatively, in [16] another method using MPC is suggested where a reinforcement learning technique was used to develop and train a neural network with generated data from the MPC together with online trajectory optimization.

3. Implementation

The objective resides in implementing an algorithm capable of avoiding obstacles while following a target without reconstructing the surroundings as the methods mentioned before. The strategy adopted in the present work consists on using a cascade control approach with an MPC controller as an outer loop to generate optimal trajectories and an attitude controller as an inner loop to guarantee flight stability. The determination of insecure regions is performed by TTC estimation and, when TTC is relatively small, the MPC controller will generate trajectories in a way that the drone will avoid the danger zone while keeping track of the object. The implemented method is compiled in the following stages: (i) Camera obtains an image to be analyzed. (ii) The target is identified and the TTC is computed. (iii) The MPC controller generates optimal commands. (iv) The attitude controller generates rotors speed guaranteeing stability and trajectory tracking. (v) The phases (i)-(iv) are repeated at each time step of the strategy.

3.1. Experimental Setup

The system consists of a drone (a quadrotor) with an IMU and a camera rigidly attached to it aligned with an arm of the vehicle.

In order to test and validate the implemented strategy, a simulation world in Gazebo with the Micro Aerial Vehicle (MAV) framework RotorS that provides the drone with all the dynamics and physical properties was used.

3.2. Image Analysis

An important element of the strategy is the image analysis. From the image information is possible to identify the target and compute the distance to it. With the OpenCV library[1] algorithms is extracted the area and position of the target in the image. Then, it is necessary to compute the TTC and perceive where are the danger zones.

TTC computations

In the avoidance strategy, obstacles are associated with a zone in the image plane with low TTC. In a static environment, the movement of features between frames will increase with the distance to the

[1] Viewed: https://opencv.org/, May 2020, computer vision library

estimated Focus of Expansion (FOE)[2] as displayed in Figure 1.

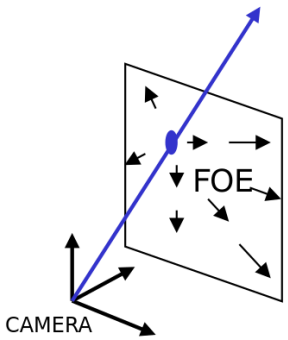


Figure 1: FOE in the image plane [17].

With the estimation of the FOE together with the pixel movement between frames ($\|(u, v)\|$), it is obtained an expression that computes the TTC for a pixel k [13], in a given frame:

$$TTC_k = \frac{\|x_k - FOE\|}{\|(u_k, v_k)\|}, \tag{1}$$

where x_k represents the 2D location of pixel k and $\|(u_k, v_k)\|$ gives the movement of pixel k from the previous frame to the current frame. To compute the features, necessary for FOE estimation and TTC calculation, is required to perform certain steps.

The first step is to detect features. Oriented FAST and rotated BRIEF (ORB) algorithm from the OpenCV library is used, and the best features are selected based on Harris Score[3]. Then, it is necessary to perform feature matching. However, it is mandatory to match features in two aligned frames, i.e., to have the same roll and pitch orientation in the camera over two consecutive frames. Figure 2 illustrates the problem:

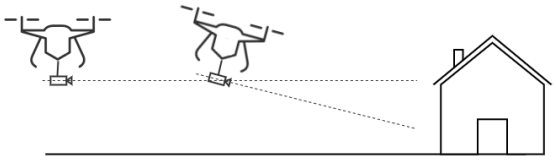
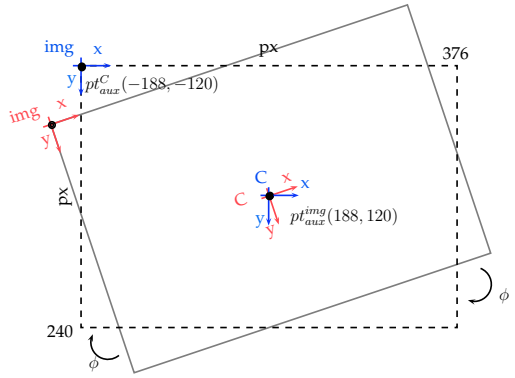
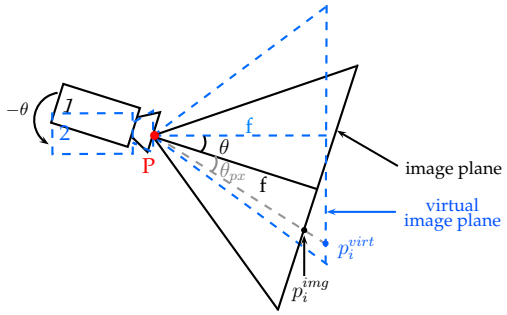


Figure 2: Camera attitude variation example during flight.

In the absence of a camera stabilizer, a feature reprojection technique is used to guarantee that the differences in both images are due to translation motion only. The method consists in a two-step virtual rotation as if the camera heading is parallel to the ground. The two-step rotation is based on the values of the IMU.The first rotation is along the optical axis with an angle $roll_{IMU}$ and the second rotation is along the horizontal line that crosses the camera center (point P in Figure 3(b)) and is parallel to the image plane with an angle $pitch_{IMU}$. Consequently, these two rotations result in a translation of the pixels x_i in the image plane and are represented in Figure 3.



(a) Image plane before and after a roll rotation of ϕ with frames $\{img\}$ and $\{C\}$ and auxiliary points in each frame respectively: pt^C_{aux} and pt^{img}_{aux}.



(b) Pitch rotation of $-\theta$ along the horizontal line in the image plane that crosses the middle point P.

Figure 3: Example rotation R_{roll} (a) and R_{pitch} (b) in the image plane.

With the information of Figure 3(a) the projected points in a virtual plane can now be expressed by

$$x_i^{proj} = R_{roll}(x_i - pt^{img}_{aux}) - pt^C_{aux}, \tag{2}$$

with pt^{img}_{aux} the image center in the $\{img\}$ frame and pt^C_{aux} the top-left corner of the image in the $\{C\}$ frame. The camera rotation in Figure 3(b) will alter the y component of the pixels x_i^{proj} in equation (2). Therefore, the projected points in the virtual plane, from the information in Figure 3, are expressed as

$$x_i^{virt} = \begin{bmatrix} x_i^{proj}(1) \\ h_{img}/2 + f\ tan(\theta_{IMU} + \theta^{img}_{px}) \end{bmatrix}, \tag{3}$$

where $x_i^{proj}(1)$ refers to the first coordinate of the projected point x_i^{proj}, h_{img} is the image height, i

[2]the point that represents where is the drone going in 3D space

[3]Harris algorithm is used to rank features

denotes the feature i, f is the focal length and θ_{px}^{img} is given by

$$\theta_{px}^{img} = tan^{-1}\left(\frac{x_i^{proj}(2) - h_{img}/2}{f}\right). \quad (4)$$

The mentioned steps are followed by feature matching. The selected ORB features, with the best Harris Score, are matched with the features selected in the previous frame by brute-force[4] and the match is only considered if the feature moved between frames. With the matches is possible to obtain the FOE. From each pair of features is obtained a line equation of the form

$$z = a + tn, \quad (5)$$

where a is the position of a point on the line and n is the direction vector. As the scalar t alters, z returns the locus[5] of the line. The distance of an arbitrary point p to this line is:

$$\mathrm{D}\ (z = a + tn, p) = \|(a - p) - ((a - p) \cdot n)n\| \quad (6)$$

The FOE will result in the point in the image that minimize the distance to multiple lines (resulting from multiple matches). Hence, computing the square of equation (6) and considering K lines it results in the following:

$$D(p; A, N) = \sum_{j=1}^{K} (a_j - p)^T \left(I - n_j n_j^T\right)(a_j - p). \quad (7)$$

Equation (7) is quadratic in p, so, deriving it in respect to p and rearranging it results:

$$\begin{aligned} Rp &= q \qquad \text{where} \\ R &= \sum_{j=1}^{K} \left(I - n_j n_j^T\right) \quad \text{and} \\ q &= \sum_{j=1}^{K} \left(I - n_j n_j^T\right) a_j. \end{aligned} \quad (8)$$

For obtaining point p, that corresponds to the FOE, least squares method is applied:

$$FOE = \widehat{p} = (R^T R)^{-1} R^T q. \quad (9)$$

However, during flight, FOE estimation is relatively unstable leading to imprecise TTC calculations [13]. To accomplish stability, the moving average method

$$FOE^j = \frac{FOE^j + FOE^{j-1} + \ldots + FOE^{j-N+1}}{N}, \quad (10)$$

was used to reduce high-frequency noise, where were considered the last N FOE points. Moreover, a variation larger than 100 pixels from the average FOE is also filtered.

Finally, the TTC is calculated according with the equation (11) [13]

$$TTC_{feature_i^j} = \frac{\|x_i^j - FOE^j\|}{\|x_i^j - x_i^{j-1}\|}\, t_r, \quad (11)$$

where the numerator refers to the distance between feature i and FOE (in the frame j), and the denominator represents the distance between the match of feature i in consecutive frames (as in equation (1)). The t_r factor is the time difference between the two consecutive frames j and $j-1$. It is indeed confirmed, from equation (11), that a feature moving far from the FOE represents a minimum threat (TTC large). Otherwise, the same movement nearby the FOE can describe a dangerous situation.

Lastly, instead of looking for each feature individually, the proposed approach will attend to groups of features and identify the danger zones in the image. The image division is represented in Figure 4.

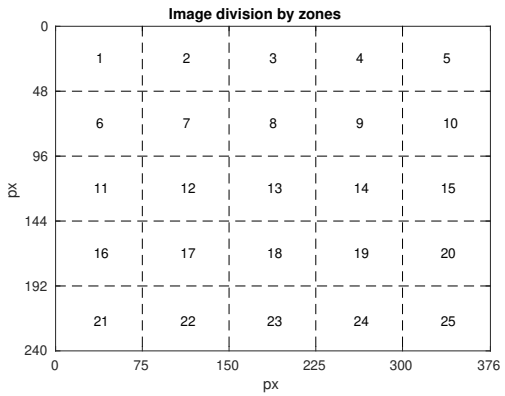


Figure 4: Image division in areas of 3600 px.

The algorithm identifies the zones in front of the drone that have a small TTC and leads the MPC to generate paths that avoid those specific zones. The TTC in each of the zones is given by the average TTC of the selected features in that zone

$$TTC_{\text{zone}i} = \frac{1}{k_i} \sum_{n=1}^{k_i} TTC_{feature_n}, \quad (12)$$

where k_i is the number of keypoints in zone i.

3.3. Drone Control

The control of the drone is based on a combination of controllers: a PD controller for attitude control with an MPC controller for trajectory generation. As a first part of this process, the two

[4] match only occurs if the feature i_A in frame A as the best match with the feature i_B in frame B and vice-versa

[5] set of all points (commonly, a line or curve), whose location satisfies one or more specified conditions

controllers were implemented in a MATLAB environment and the ACADO Toolkit [18] was used to generate C/C++ code for the MPC problem.

MPC Controller

The MPC controller is responsible for generating optimal trajectories by solving an optimization problem. To formulate the MPC problem it is established the state vector

$$\boldsymbol{x} = [x \quad y \quad z \quad v_x \quad v_y \quad v_z \quad \phi \quad \theta \quad \psi], \tag{13}$$

where $p^T = [x\ y\ z]$ is the position of the drone, $v^T = [v_x\ v_y\ v_z]$ the velocity of the vehicle and $q^T = [\phi\ \theta\ \psi]$ the orientation of the drone in the world frame. The control input is defined as

$$\boldsymbol{u} = [\phi_{cmd} \quad \theta_{cmd} \quad \psi_{cmd} \quad F_T], \tag{14}$$

where $F_T = k_F \sum_{i=1}^{4} \omega_i^2$, with k_F the thrust coefficient, is the total thrust force exerted by the four rotors. The commanded angles and force are then limited by the respective set

$$\begin{aligned} \mathbb{U} = \{\boldsymbol{u} \in \mathbb{R}^4 | \begin{bmatrix} -\frac{\pi}{4} \\ -\pi \\ 0 \end{bmatrix} \leq \\ \leq \begin{bmatrix} \phi_{cmd}, \quad \theta_{cmd} \\ \psi_{cmd} \\ F_T \end{bmatrix} \leq \begin{bmatrix} \frac{\pi}{4} \\ \pi \\ 28 \end{bmatrix} \}. \end{aligned} \tag{15}$$

The attitude model used was based on [15] resulting in the attitude model

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_\phi}\left(k_{p\phi}\phi_{cmd} - \phi\right) \\ \frac{1}{\tau_\theta}\left(k_{p\theta}\theta_{cmd} - \theta\right) \\ \frac{1}{\tau_\psi}\left(k_{p\psi}\psi_{cmd} - \psi\right) \end{bmatrix}. \tag{16}$$

Finally, the full drone model is expressed as

$$f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} v \\ -g + (RF_T + F_{ext})/m \\ \dot{q} \end{bmatrix} \in \mathbb{R}^9, \tag{17}$$

where $g = [0\ 0\ 9.81]\ m/s^2$ is the gravitation force, m is the quadrotor mass, F_{ext} represents the external forces (i.e. wind) acting on the drone and R is the rotation matrix

$$R = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix}, \tag{18}$$

where c and s are, respectively, shorthand forms for cosine and sine.

To establish a complete MPC problem it is also necessary to determine a cost function. Since the MPC controller is responsible for generating trajectories, the objective function consists in the difference between the current pose and the reference or desired pose

$$\mathcal{C}(x, u) = \|p - p_{des}\| + \|q - q_{des}\|. \tag{19}$$

The OCP is now defined as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \int_{t=0}^{T} \|p(t) - p_{des}(t)\| + \|q(t) - q_{des}(t)\|\ dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ & \boldsymbol{u}(t) \in \mathbb{U} \\ & \boldsymbol{x}(0) = \boldsymbol{x}(t_0). \end{aligned} \tag{20}$$

Attitude Controller

The attitude controller will perform drone stabilization to achieve the desired orientation. The goal is to have a null roll and pitch values ($\phi = \theta = 0$) so that the drone preserves a movement close to the hover condition. So, from the OCP in (20) it results

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \int_{t=0}^{T} \|p(t) - p_{des}(t)\| + \|\psi(t) - \psi_{des}(t)\|\ dt \\ \text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\ & \boldsymbol{u}(t) \in \mathbb{U} \\ & \boldsymbol{x}(0) = \boldsymbol{x}(t_0). \end{aligned} \tag{21}$$

Image Integration in the Control Problem

The obstacles are estimated from the TTC calculation together with the danger zone associated as in Figure 4. To relate this information in the OCP and guarantee a safe flight it is necessary to have a reasonable distance between obstacle and drone. Therefore, the adopted approach, consists of using a potential field in the cost function that repulses the drone from dangerous zones. The potential term is hyperbolic making the cost function to increase when the drone is at a certain distance d to the obstacle. The objective function is now

$$\begin{aligned} \mathcal{C}(x, u) = \|p - p_{des}\| + \|\psi - \psi_{des}\| + \\ + \sum_{k=0}^{\mathcal{O}-1} \left(\frac{d}{\|p - p_{crit_k}\|} \right)^{p_w}, \end{aligned} \tag{22}$$

where p_{crit_k} is the position associated to the obstacle k resulting from the center point of a zone of the image, as in Figure 4, with a small TTC, $\mathcal{O}$ is the number of obstacles calculated during flight and p_w is the power that makes the cost increase if the distance to the critical point $\|p - p_{crit_k}\|$ is smaller

than d. The OCP to be exported from MATLAB is

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \int_{t=0}^{T} \|p(t) - p_{des}(t)\|_Q + \|\psi(t) - \psi_{des}(t)\|_P + \\
& + \sum_{k=0}^{\mathcal{O}-1} \left(\frac{d}{\|p(t) - p_{crit_k}(t)\|} \right)_J^{p_w} dt \\
\text{s.t.} \quad & \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \\
& \boldsymbol{u}(t) \in \mathbb{U} \\
& \boldsymbol{x}(0) = \boldsymbol{x}(t_0),
\end{aligned} \tag{23}$$

where Q, P and J are weight matrices.

The tracking part, represented by the first two terms in equation (22), directly influences the generated trajectory. When the target is identified, the area and the center of the target in the image plane are computed. With this information, it is possible to know the drone distance and the yaw orientation, ψ, to the target. Therefore, at every position of the drone along time, the target position, the desired position and the position of the obstacles are represented in the drone frame. The goal is to maintain the target centered with a pre-determined area in the image plane leading to the p_{des} and ψ_{des}.

OCP Solution and Code export

To test the MPC problem in the Robot Operating System (ROS), ACADO Toolkit [19] was used to generate the C/C++ code. For solving the OCP (23) a multiple shooting technique is used leading to the discretization of the system model and constraints in $N = 40$ time steps $t_0, ..., t_{39}$ with duration $dt = 0.05s$ which results in a prediction horizon of $2s$. A real-time iteration Gauss-Newton method is auto-generated and uses an algorithm for solving the quadratic programs (QPs). For last, ACADO CODE GENERATION exports multiple files containing the global values and functions together with a template to run the MPC algorithm.

4. Results

The results from some drone flights illustrate the performance of the solution in the Gazebo environment. First, the TTC computations are verified and followed by a validation with real video data. Finally, with the target detection in the image plane and the TTC outcome, the control problem in equation (23) is solved in real-time, where it is used an attitude controller for attitude stabilization, leading the drone to avoid obstacles while following the target.

4.1. Vision Results

In the Gazebo environment, a camera that acquires 10 images per second is mounted on the drone. The present situation consists of a drone flying towards a house as exemplified in Figure 5.

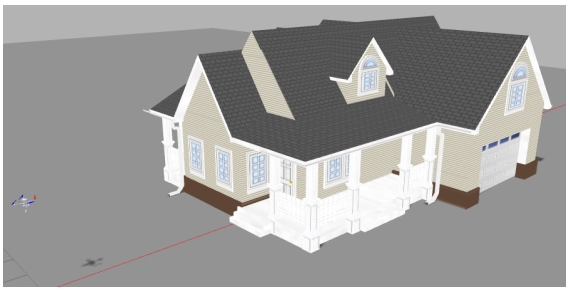


Figure 5: Drone flight in direction of a house in Gazebo simulation.

This environment is used to verify the TTC computations. From the Gazebo data, the real time-to-collision is obtained and the differences to the estimated TTC are displayed in Figure 6.

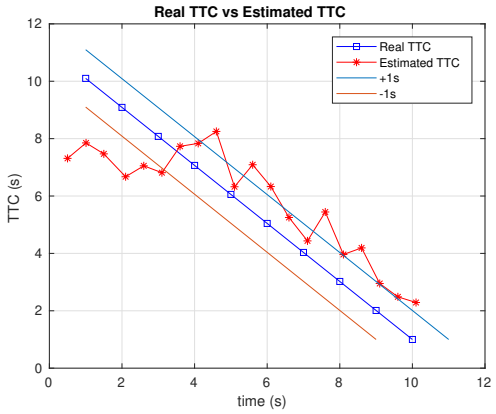


Figure 6: TTC results from drone flight in Figure 5.

From Figure 6 it is verified that the computed TTC has similar behavior to the real TTC. However, after the first 4s, the estimated TTC is larger than the real TTC. This event is because the TTC computed above uses the data from all zones of the image, therefore, feature points far from the image center can contribute to the observed increase in the TTC. Additionally, some miss computations influence the presented results, and this fact is verified in a new trial, in the calculation of TTC by zone exhibited in Figure 7, where only the results in the 3x3 central zones of the Figure 4 are displayed.

From Figure 7 it is possible to verify that the TTC evolves, in general, according to the real time-to-collision. Nevertheless, there are some differences from the real results and these are exhibited in Figure 8.

The results in Figure 8 show that the TTC computations have some errors that lead the system to estimate a collision without full precision.

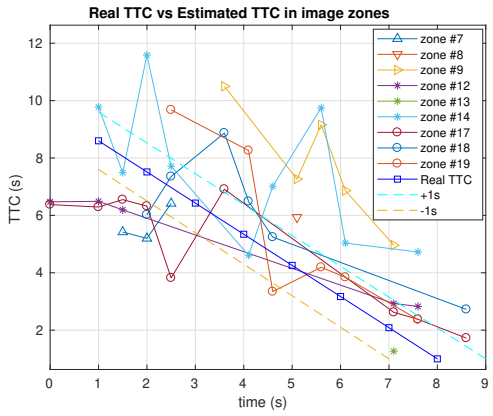


Figure 7: TTC results in image zones from the drone flight in Figure 5.

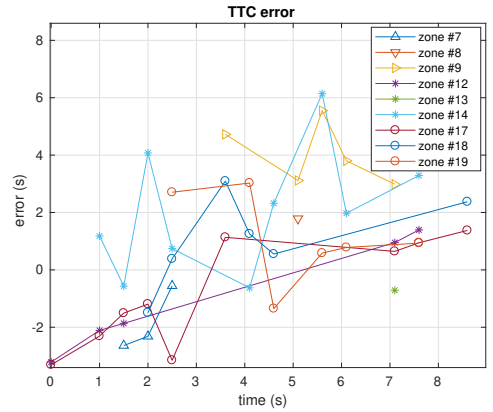


Figure 8: TTC error in image zones from the drone flight in Figure 5.

TTC validation with real data

The TTC calculations were also validated with real video data. The test environment was a situation where the video starts at the beginning of a hall and finishes colliding into an obstacle. The video was recorded on a moving platform, with a phone camera, recording at $30Hz$, with a size of 1920x1080 px. Then, a central region of interest with size 960x540 px was analyzed. Figure 9 illustrates the situation in three different time frames together with the marked features in each situation.

From Figure 9 it is possible to observe that the algorithm has the expected performance in the TTC computation. When the camera is far from the obstacle, the pixels have an associated TTC greater than 8s. When the camera gets closer to the obstacle, the TTC values are smaller as predicted.

In this situation, the time-to-collision was computed in each of the 25 zones of the image, producing the results in Figure 10.

From Figure 10, it is visible that the estimated TTC has similar behavior to the real TTC. The results in some zones are close to the real value, namely, the data from zones 12, 13, and 18 show identical evolution. The discrepancy in the results of zone 8 compared with the other zones can be explained by the mismatching of some features leading to an increase of the TTC.

From a global perspective, the outcome with the real data are in accordance with the one obtained in the simulation environment, nonetheless, the differences to the real value may lead the drone to misjudge the environment in front.

4.2. Obstacle Avoidance Results

To verify the performance of the system, two different flights were performed in a simulated environment as illustrated on Figure 11(a) and 11(b). In both environments, the target maintains the height during the experiments. Its movement in the xy plane is represented for the tree situation in Figure 12 followed by the movement of the drone in 10 trials in Figure 13[6].

Furthermore, in both situations, the target circumvents the obstacle with a safe distance of $30cm$. To evaluate the performance of the solution, 100 tests for both environments were executed. The safety distances for each trial are displayed in the boxplots in Figure 14 and in Table 1.

Table 1: Drone flight success rate and the median of the safety distance in both environments along 100 tests.

Env	Success (%)	$\tilde{d}$ (m)	Outliers
wall	82	1.46	1
tree	99	1.32	7

As for the one wall situation, the success rate is 82%. The 18 tests that failed were due to imprecise TTC computations or/and insufficient features in the image plane, that led the drone to not identify some regions as dangerous. The information on the distance to the obstacle reveals that the avoidance maneuver is relatively consistent while avoiding the wall. The outlier can be explained by a situation where an obstacle is only on the border of a zone, leading the algorithm to consider that zone as occupied and producing an avoidance maneuver very far from the obstacle.

Regarding the tree situation, in all the 100 trials, a danger zone was identified. However, the success rate was 99% due to one trial where the considered obstacle was on the right side of the tree, creating a potential field that led the drone to execute a safety maneuver to the left side. In these circumstances, the target was occluded by the tree, leading the drone to stop moving because the target was not on sight. Nonetheless, the safety distance results indicate that the drone accurately recognizes a danger

[6]for the wall situation, the movements are analogous

(a) Camera far from the obstacle. (b) Camera half-way to the obstacle. (c) Camera close to the obstacle.

Figure 9: Three different stages in a hall 'flight'. The colored dots correspond to the marked matches by the vision algorithm with the following TTC: Green points: TTC $>$ 8s. Yellow points: 4s $<$ TTC $\leq$ 8s. Red points: TTC $\leq$ 4s.

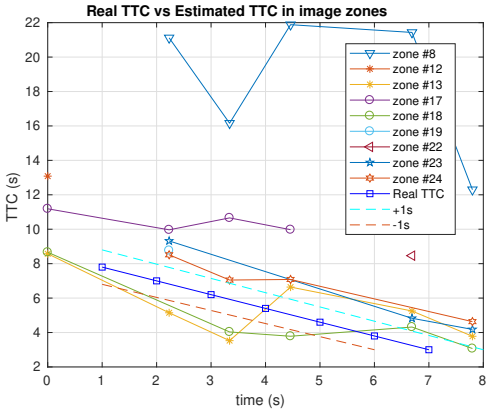


Figure 10: TTC results from video in the hall.

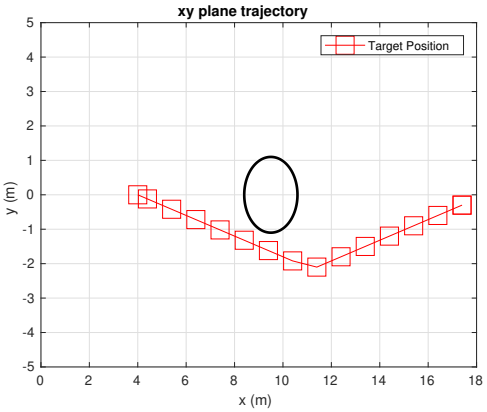
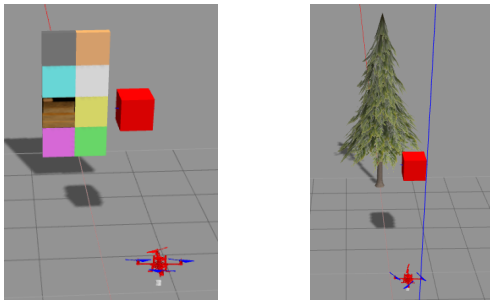


Figure 12: Movement of the target in tree flight.



(a) One wall obstacle. (b) Tree obstacle.

Figure 11: Two obstacle situations in Gazebo environment.

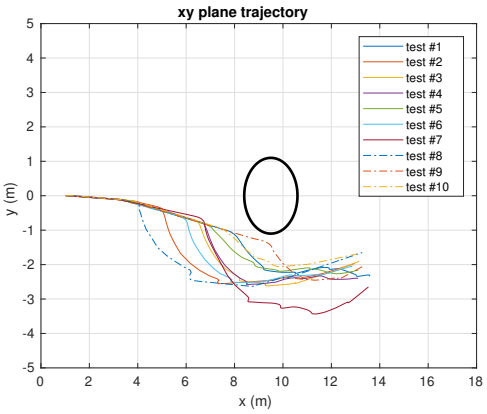


Figure 13: Movement of the drone in tree flight in 10 different trials.

zone in the image plane. The three outliers below 0.55m are explained in situations where the obstacles were considered on top or bottom of the tree leading to a potential field that does not repel the drone to a safer distance. Moreover, the remaining four outliers can be explained by the same fact as the outlier in the one wall situation.

4.3. Computational Time

In order to fully evaluate the developed system, the last parameter to be analyzed was the computational cost. A longer simulation, with three obstacles, was performed and both the vision and the MPC iteration time were extracted. The results are in Figure 15.

From Figure 15, it is clear that the vision algorithm and an MPC real-time iteration are both fast with a computational time smaller than 0.1s. The vision algorithm results are due to the usage of ORB features that contribute to a fast matching between frames. The MPC iteration result represents a solution of the problem in equation (23), where there is an optimization problem and constraints that have to be fulfilled. Moreover, it should be noticed that the size of the state and the number of control inputs are also important in the duration of the com-

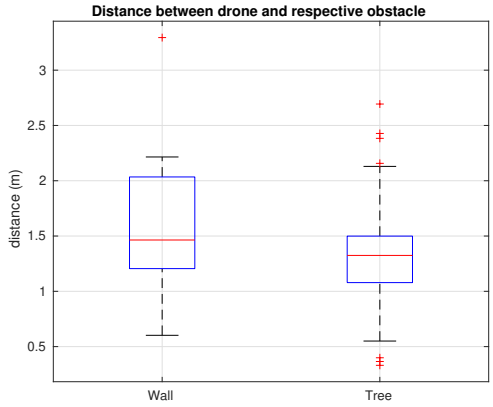


Figure 14: Drone safety distance in both situations along 100 tests.

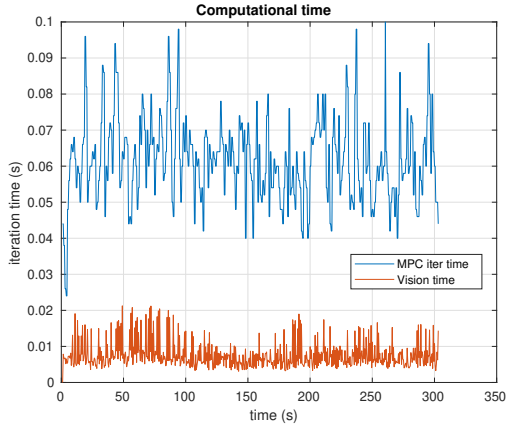


Figure 15: Computational time for both MPC and Vision iterations.

putational time.

One of the specific aspects of the implemented strategy is that a full space reconstruction is not needed. The strategy relies on considering obstacles, from the vision algorithm, in each iteration. This means that, in every iteration, the obstacles are derived from the TTC calculations. However, other strategies could be used, such as, space reconstruction strategies that enable the drone to perceive in full the surrounding environment. A comparison between the vision strategy used and some of the existing reconstruction strategies in the literature is displayed in Table 2, where similar systems and processors were used. The associated time cost of the present work (p.w.) is presented for the final experiment (denoted by sim) and for the experiment with real data (denoted by real).

Besides the small differences in the systems used, it is also relevant to remember that, the strategy of the present work in both simulation and real video, uses a region of interest of the image. Although both Monocular ORB-SLAM [20] and Stereo LSD-SLAM [10] are not used to avoid obstacles, the

Table 2: Iteration computational time and standard deviation in different vision strategies.

Strategy	Resolution(px)	iter(ms)	std(ms)
p.w. (sim)	752 x 480	7.6	3.7
p.w. (real)	1920 x 1080	26.9	2.4
Mono ORB--SLAM [20]	512 x 384	496	218
Stereo LSD--SLAM [10]	620 x 184	70.5	-

results show that the monocular solution is not adequate to a real-time implementation while the stereo solution is faster and can integrate a real-time application to avoid obstacles. Nonetheless, it is notable that the strategy used in this work has a smaller computational cost than the two presented strategies.

5. Conclusions

The goal of this project is to develop a real-time obstacle avoidance system for a drone while following a target using a monocular camera and an IMU. The work was performed with an MPC controller, implemented to generate commands according to the TTC computations, the IMU information, and the position and size of the target in the image plane. It was shown that the problem of obstacle avoidance was solved, since it was designed a system that can, in real-time, track a target, understand the world in front of the drone due to TTC computations, avoid obstacles, and solve an MPC problem leading to a generated safe trajectory. The scale problem, inherent to a monocular camera, was solved with the TTC approach. The developed system revealed a constant behavior in the avoidance maneuvers performed, nonetheless, there are some details in the detection strategy, in both hardware and implementation, that can be adjusted and improved to make the system flawless.

Concerning future work, several aspects can be experimented and improved. The first point that could be modified is the usage of the created algorithm only on drones with a monocular camera. A depth sensor or a stereo camera (different drone) could be used to add distance information. Next, the target detection method can be changed to another strategy that is not color-dependent, avoiding wrong detection events due to the environment colors. Moreover, an algorithm that 'searches' the target when this one is occluded, is an interest procedure that must be done to increase robustness of the strategy. The final step is to implement and test the developed system in a real flight experiment.

References

[1] Chad Goerzen, Zhaodan Kong, and Bernard Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65, 2010.

[2] Piotr Rudol and Patrick Doherty. Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery. In *2008 IEEE aerospace conference*, pages 1–8. IEEE, 2008.

[3] James W Waite, Thorkell Gudmundsson, and Dimitar Gargov. UAV power line position and load parameter estimation, May 19 2015. US Patent 9,037,314.

[4] G Saggiani, F Persiani, A Ceruti, P Tortora, E Troiani, F Giuletti, S Amici, M Buongiorno, G Distefano, G Bentini, et al. A UAV system for observing volcanoes and natural hazards. *AGUFM*, 2007:GC11B–05, 2007.

[5] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft UAVs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 807–814. IEEE, 2008.

[6] C Giannì, M Balsi, S Esposito, and P Fallavollita. Obstacle detection system involving fusion of multiple sensor technologies. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:127, 2017.

[7] Vahid Behzadan and Arslan Munir. Adversarial reinforcement learning framework for benchmarking collision avoidance mechanisms in autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 2019.

[8] Sergio García, M Elena López, Rafael Barea, Luis M Bergasa, Alejandro Gómez, and Eduardo J Molinos. Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 205–210. IEEE, 2016.

[9] Kimberly McGuire, Guido De Croon, Christophe De Wagter, Karl Tuyls, and Hilbert Kappen. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. *IEEE Robotics and Automation Letters*, 2(2):1070–1076, 2017.

[10] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.

[11] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[12] Yuncheng Lu, Zhucun Xue, Gui-Song Xia, and Liangpei Zhang. A survey on vision-based UAV navigation. *Geo-spatial information science*, 21(1):21–32, 2018.

[13] Chaoqun Wang, Wei Liu, and Max Q-H Meng. Obstacle avoidance for quadrotor using improved method based on optical flow. In *2015 IEEE International Conference on Information and Automation*, pages 1674–1679. IEEE, 2015.

[14] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *2013 IEEE International Conference on Robotics and Automation*, pages 1750–1757. IEEE, 2013.

[15] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC-PapersOnLine*, 50(1):3463–3469, 2017.

[16] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 528–535. IEEE, 2016.

[17] Jorge S. Marques and José Santos-Victor. *Lecture slides, Optical Flow.* Instituto Superior Técnico, Image Processing and Vision, 2019.

[18] D. Ariens, B. Houska, and H.J. Ferreau. Acado for matlab user's manual, 2010–2011.

[19] B. Houska, H.J. Ferreau, M. Vukov, and R. Quirynen. Acado toolkit user's manual, 2009–2013.

[20] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.