IEEE Transactions on Human-Machine Systems, Volume 52, Number 2, April 2022, pp. 157-168
DOI:10.1109/THMS.2022.3142107

OAST: Obstacle Avoidance System for Teleoperation of UAVs

Hugo Courtois*‡, Nabil Aouf†, Kenan Ahiska* and Marco Cecotti§

***Abstract*—This paper presents a novel flight assistance system, OAST (Obstacle Avoidance System for Teleoperation), whose main role is to make teleoperation of small multirotor UAVs safer and more efficient in closed spaces. OAST allows the operator to avoid obstacles while keeping a liberty of movement. The UAV is controlled through a 3D haptic controller and OAST amends the user input to increase safety and efficiency of the teleoperation. The design of OAST is verified in computerized experiments. Moreover, a simulation involving 20 participants is carried out to validate the proposed scheme. This experiment shows that OAST improves the completion time of the scenarios by 41% on average while reducing the workload of the operator from 57 to 27 points on the NASA Task Load Index test. The number of collisions with the environment is all but eliminated in these scenarios.**

***Index Terms*—Obstacle Avoidance, Unmanned Aerial Vehicle, Teleoperation, Human-Robot Interaction**

I. Introduction

TELEOPERATION of UAVs is an important subject since some tasks cannot be automated easily and require a human operator to fly the UAV. It is thus important to improve teleoperation capabilities [1]. A critical challenge of teleoperation for small drones in closed environments is obstacle avoidance, mostly because the sensory information available to the operator is limited. On the other hand, the nature of teleoperation makes global path planning unnecessary, since the operator determines the trajectory of the UAV.

The objective of this paper is to propose a system that facilitates teleoperation of the UAV by avoiding obstacles while keeping a liberty of movement. Multirotor UAVs are considered because their agility allows them to be used in closed spaces. There are two ways to achieve obstacle avoidance during teleoperation of a UAV: if a haptic controller is used to operate the UAV, the haptic feedback can be used to help the operator avoid obstacle. The second option is to filter the input of the user to achieve a collision free trajectory. The haptic feedback can then be used to inform the operator about this filtering.

Manuscript received on
*Hugo Courtois and Kenan Ahiska are with the Centre for Electronic Warfare, Information and Cyber, Cranfield University, Defence Academy of the UK, Shrivenham, UK
†Nabil Aouf is with the Department of Electrical and Electronics Engineering, City University of London, London, UK
§Marco Cecotti is with the Advanced Vehicle Engineering Centre, Cranfield University, Cranfield, UK
‡Corresponding author: Hugo Courtois: h.r.courtois@cranfield.ac.uk
This work was part of a PhD thesis at the Centre for Electronic Warfare, Information and Cyber, Cranfield University, Defence Academy of the UK, Shrivenham, UK. The authors thank the EPSRC for providing funding.

In order to perform obstacle avoidance through the haptic feedback of a compatible controller, artificial force fields have been used (e.g., the distance between the obstacle and a robot arm is used to create the field [2]). These artificial fields for UAVs have been studied extensively, notably the Parametric Risk Field (PRF) [3]. A comparison using force and stiffness feedback with these force fields took place using 2D joysticks [4]. A force feedback was directly applied to the end effector which was actively deflected. On the other hand, a stiffness feedback modifies the spring constant of the spring that links the end effector to the centre of the controller workspace. The work of Lam, Mulder, Van Paassen, *et al.* concluded that haptic feedback is able to improve safety and with proper tuning can also reduce workload for the operator [4]. Even though the aforementioned work took place in 3D, the obstacle detection and haptic feedback were limited to 2D only. In the work of Courtois and Aouf, the PRF was extended to 3D with the addition of a scheme to reduce the effect of false positives created by obstacles of symmetric shapes [5].

A comparison of four force feedback algorithms [6] has found the most efficient to be the "Time To Impact" (TTI). The TTI algorithm computes the time before an impact by considering the distance between the UAV and the obstacle, and the current velocity of the UAV. The haptic force is then inversely proportional to the the time to impact. In a following study, a pure stiffness algorithm was introduced [7]. The authors showed that this algorithm both reduces the number of collisions and the workload of the operator. A limitation of this method is that the obstacles are only detected in 6 directions, which means that obstacles not along the x, y, or z axes do not generate any feedback. Moreover, this is a pure stiffness feedback, thus if the displacement of the controller is small, the feedback will also be small. A pure stiffness approach is also vulnerable to moving obstacles for the same reason: if the controller is not deflected, and a moving obstacle gets close, then the user does not feel any feedback.

To summarize the results of the experiments conducted in those studies, the haptic feedback generally improves safety, meaning it reduces the collisions with the environment [4]–[7]. However, a reduction of the workload of the operator depends on the algorithm used and its tuning. The scenario completion time is either increased [4] or unchanged compared to the no feedback case [5]–[7]. While the safety is generally improved with the help of the haptic feedback, none of those methods has managed to eliminate collisions, which is a problem for small UAVs due to their fragility. A hypothesis is that, since a haptic feedback reduces the position of all obstacles to a single vector with three components, it is difficult to obtain

a feedback that is precisely adapted to the location of the obstacles. Moreover, the tuning of the haptic feedback is delicate to achieve a good balance between safety of the UAV and ease of use for the operator: more ease of use might reduce safety while stronger feedback might reduce ease of use [4], [5], [8].

The methods presented above are not tied to a particular detection system. Some methods linking obstacle detection and haptic feedback exist (e.g., the haptic force can be directly derived from the optical flow through the concept of optical impedance [9]).

The second option for achieving obstacle avoidance is to override the input of the user to achieve a collision free trajectory. The method proposed in this paper falls in this category. In the work of Hua and Rifaï [10], the desired velocity of the UAV was adapted so that it does not collide with a single obstacle. This method was extended to several obstacles by Omari, Hua, Ducard, *et al.* [11]. The method requires distance to the obstacles, thus lidar sensors are suitable to implement it, as well as cameras. This work relies on a specific control law being applied to the UAV and assumes a velocity controller.

A similar approach, called DKB, was adopted by Hou and Mahony [12], where the distance to an obstacle was used to alter the desired velocity of the UAV so that it stops before hitting an obstacle. The authors showed that, if perfect velocity tracking is assumed, then obstacle avoidance is guaranteed. In the same paper, the authors provided a comparison between their method and two other algorithms using haptic feedback for performing obstacle avoidance. This experiment showed that the DKB outperformed the two other algorithms when considering the total time to complete the task and the number of collisions, among others metrics. The decrease in workload observed on average was however not statistically significant. Contrary to the algorithm proposed in this paper, the DKB involves a 3D haptic controller in the admittance framework.

In the work of Odelga, Stegagno, and Bülthoff [13], an RGB-D camera is used to create a robot centric, circular grid map of obstacles around the robot. Obstacles are tracked using a bin-occupancy filter. A discrete set of options for the movement of the UAV is defined and model predictive control with a fixed time horizon is used to perform the obstacle avoidance. Similar to the last method, the velocity of the UAV is altered to avoid the obstacles. This method is interesting since the use of model predictive control means that we expect the obstacle avoidance to behave well with more complex obstacles, however this is not tested in the paper, where a single obstacle with simple shape is used. The work of Odelga, Stegagno, and Bülthoff [13], similar to the method proposed in this paper, involves a probabilistic map, meaning that obstacles are updated continuously through the flight. Moreover, an experimental validation with a real UAV is proposed by the author [13]. However, the evolution of both the obstacle avoidance performances and the computation load in more complex scenarios are not specified. The impact of the method on the workload of the operator is also not evaluated: it would be interesting to see if or how the value of the time horizon would affect the behaviour of the operator. The runtime of the method is not specified. In particular, the change of runtime with the number of tracked obstacles is absent.

Several conclusions can be drawn from the existing literature. Firstly, overriding the input from the user to a safer value might be more efficient than relying exclusively on haptic feedback. This is the approach that is chosen in this work. The user input can be the velocity of the UAV [10], [12], [13]; however, the sensors often provide the positions of the surrounding obstacles. Thus, the question that needs to be answered is: given an obstacle position, what velocity should the UAV adopt to not collide with this obstacle? A solution to this problem involves some sort of planning ahead while considering the dynamics of the UAV. Model Predictive Control can be used [13], or other specific control schemes can be adopted [10]. It is argued that this problem can be separated in two different issues: finding a safe position for the UAV and navigating to this safe position. The navigation part of the problem, meaning the transformation of a desired position into UAV rotors commands, can be solved by a position controller for which a large body of literature exist (e.g., backstepping [14] or based on geometric methods [15]). It is proposed to decouple those two issues, and focus on the first part of the problem: finding a safe position for the UAV. This way, the obstacle avoidance method is not attached to a specific controller, making the scheme more flexible. The problem of ensuring that the safe position is fully reachable when taking the dynamics of the UAV into account is not explicitly tackled in this paper. However, the context of UAV teleoperation means that a trajectory is rarely followed to its end and the position controller constantly reevaluates the rotors commands to reach the safe position.

To summarize, our algorithm aims to compute a safe position by filtering the position command from the operator.

The drawback of achieving a safer operation this way is that the operator does not have an absolute control over the UAV anymore. The degree of the problem is difficult to evaluate theoretically, which is why it is necessary to perform an experiment involving human operators. In this paper, we will present an experiment to test if OAST imposes too many restrictions on the UAV movements.

The contribution of this paper is twofold. First, we propose an integrated solution to improve the remote operation of UAVs: Obstacle Avoidance System for Teleoperation (OAST). This approach takes advantage of a compact probabilistic representation of the environment, the Normal Distribution Transform Occupancy Map (NDT OM), to compute a collision free command consistent with the original command so that both the workload of the operator and the task completion time are reduced. Second, an experimental validation of OAST is carried out in simulated environments, including a quantitative and qualitative analysis of the influence of our algorithm on the performance of 20 participants working on representative tasks.

The general setting of this paper is described in Section II. The components of the proposed system are described in Section III-A. In Section IV, our scheme is tested in computerized experiments. The proposed method is then evaluated in an experiment involving 20 untrained operators, analyzed in

Section V.

II. Problem settings

In the context of this paper, the teleoperation of a UAV using a haptic controller is considered. An algorithm that computes a safe position objective by filtering the position command from the operator is proposed.

For clarity, in the rest of the paper, the world reference frame is denoted w, while the haptic controller reference frame is denoted j. A superscript is used to indicate that a variable belongs to a given reference frame.

The human pilot operates the UAV using a 3D haptic controller, by specifying a position objective $\mathbf{x}_\mathbf{r}^\mathbf{w}(t)$ for the UAV. In order to compute the position objective, the position of the end effector of the haptic controller is rotated around the z axis in accordance with the desired yaw. The resulting vector is then added to the position of the UAV.

This position objective is filtered by OAST, creating a safe desired position $\mathbf{x}_\mathbf{rf}^\mathbf{w}(t)$. The UAV controller is used to reach this position. In practice, $\mathbf{x}_\mathbf{r}^\mathbf{w}(t)$ may change quickly. OAST does not wait to reach the safe position $\mathbf{x}_\mathbf{rf}^\mathbf{w}(t)$ before considering a new position. The system always uses, as soon as it is available, the filtered desired position. Note that, if the target position can be safely reached, OAST does not modify the position objective. Also, if the user does not indicate a desired direction of movement by keeping the end effector of the haptic controller at the centre of the workspace, OAST does not calculate a safe position regardless of the presence of moving obstacles. This was a choice to ensure that the user is always in control of the movements of the robot.

A UAV equipped with a realistic 3D lidar scanner and a camera is considered in this paper. Only the lidar is used by OAST, while the camera is used exclusively to provide a visual feedback to the human operator. No a priori knowledge of the world is assumed. OAST uses the point clouds from the lidar and the odometry to build an internal map. Note that a realistic 3D lidar is used, meaning a limited vertical field of view ($30°$) is assumed in the experiments.

III. Description of the system

A. System architecture

The system architecture is composed of five parts and is illustrated in Fig. 1. The map update, the obstacle avoidance module, and the command interpreter run in parallel at the specified frequencies, meaning that in between the reception of two consecutive desired positions from the 3D haptic controller, OAST can compute several safe positions to adapt to quick changes in the map and in the UAV position. This means that the system continuously responds to the commands of the operator. In this section, the different parts of OAST are described.

B. The map

The map is used to store the location of the obstacles surrounding the UAV. The map is based on the NDT OM framework [16], extended to allow unlimited movement of the robot by recentering the map when the robot gets too far from the center. NDT OM provides a compact representation of the environment: each point cloud is merged into the current map, in a manner by which the map size has a fixed upper bound that does not depend on the number of point clouds received nor their sizes. This upper bound only depends on the maximum number of Gaussians that are located in a fixed size cuboid, which constitutes the map. Notably, the maximum memory footprint does not change with time. Moreover, NDT OM does not require a static environment. The fusion process between two NDT representations, once odometry is available, operates around $100\,\mathrm{Hz}$. Note that the NDT OM algorithm can allow odometry to be computed directly from the point cloud acquisition [17]. As a result of this process, a set of Gaussian distributions describing the obstacles around the UAV is obtained.

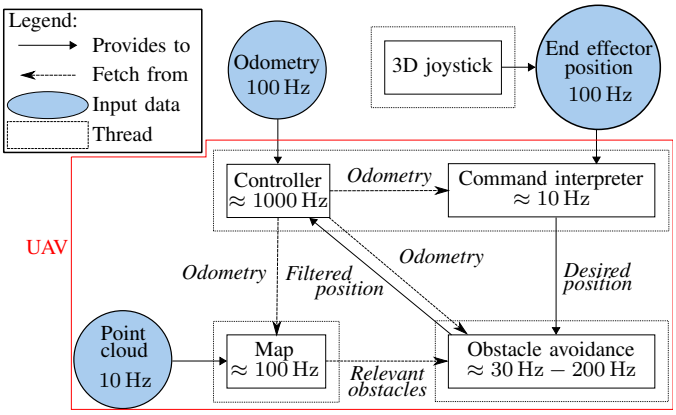

Fig. 1. Complete system for UAV obstacle avoidance

The cells containing those Gaussians are relatively small compared to the environment: their length is $0.3\,\mathrm{m}$ for the experiments done in this work. This means that complex obstacles would be represented using many different ellipsoids, allowing a detailed representation of the environment. Moreover, it is possible to increase the map resolution by reducing the cell size, with the downside of a higher computational burden.

C. The UAV controller

The controller used is a bespoke implementation of the work from [15] used as a position controller.

Odometry computation is a widely studied field using cameras [18], [19] or lidars [20], with multiple implementations available [21]–[26]. While the map module would be able to compute the odometry of the robot as mentioned above, in a real setting, odometry would be computed using different available sources (IMU, camera, lidar). In order to separate evaluation of the odometry from the evaluation of the obstacle avoidance system, the ground truth odometry will be used in the simulations of this paper.

D. The 3D haptic controller

The user operates the UAV through the 3D haptic controller in impedance mode by setting a position objective. The control scheme used by OAST is inspired from the work of Omari,

Hua, Ducard, *et al.* [11]: the workspace of the manipulator is divided in two, an inner cuboid part, where the UAV stays still to allow for hovering, and an outer part, where the scheme is a position controller with the position target moving with the UAV, which is similar to a velocity controller as presented in the work of Lam, Mulder, Van Paassen, *et al.* [4].

The desired displacement of the UAV is proportional to the displacement of the end effector of the haptic controller aligned in yaw to the UAV. The position of the UAV is taken as a reference to allow unlimited movement, meaning that the UAV will effectively follow the yaw-adjusted direction pointed to by the joystick when the end effector is outside of the virtual cuboid. Let $\mathbf{p^j}(t) = [p_x^j(t), p_y^j(t), p_z^j(t)]$ be the normalized position of the end effector such that each component is between -1 and 1, r^* be the normalized edge length of the inner cuboid relative to the workspace size and h^j be the function:

$$h_{r^*}^j(x^j) = \begin{cases} 0, & \text{if } |x^j| < r^*, \\ x^j - sign(x^j)r^*, & \text{if } |x^j| \geq r^*, \end{cases} \quad (1)$$

with $x^j \in \mathbb{R}$, $r^* \in [0,1]$ and $sign$ a function returning the sign of its argument. Then, assuming that the desired yaw is θ_y, the desired position is given by:

$$\mathbf{x_r^w}(t) = \mathbf{x_{UAV}^w}(t) + \mathbf{R}_z(\theta_y)\mathbf{K}_s \begin{bmatrix} h_{r^*}^j\left(p_x^j(t)\right) \\ h_{r^*}^j\left(p_y^j(t)\right) \\ h_{r^*}^j\left(p_z^j(t)\right) \end{bmatrix}, \quad (2)$$

where $\mathbf{x_r^w}(t)$ is the desired position for the UAV, $\mathbf{x_{UAV}^w}(t)$ is the current position of the UAV, $\mathbf{K}_s$ is a positive definite scaling matrix and $\mathbf{R}_z(\theta_y)$ is a rotation matrix of θ_y radians around the z^w axis. This desired position is then provided to the position controller described in the section above.

In order to avoid a wind up effect, the distance between $\mathbf{x_r^w}(t)$ and $\mathbf{x_{UAV}^w}(t)$ is limited to a maximum of $1\,\mathrm{m}$. This is not a limitation in practice since OAST is supposed be used as a mixed system, during which the operator continuously operates the UAV. Thus, desired positions far from the UAV do not make sense in this context.

There are two forces acting on the end effector:

1) A spring links the end effector to the center of the workspace. The spring coefficient of this spring is $12.5\,\mathrm{N\,m^{-1}}$.
2) A force gives a haptic cue to the operator about the trajectory adjustment performed by OAST.

Those two forces are described in more details in Section III-F. Moreover, automatic gravity compensation is performed by the manipulator. This means that the UAV can hover when the operator releases the end effector.

E. The obstacle avoidance module

1) General description: The obstacle avoidance module is the core of the proposed system. Given a position of the UAV $\mathbf{x_{UAV}^w}(t)$, a desired position $\mathbf{x_r^w}(t)$ generated by the command interpreter and a map, the objective of the obstacle avoidance module is to find a collision free desired position $\mathbf{x_{rf}^w}(t)$ for the UAV. Let $\mathcal{G}$ denote the set of Gaussian distributions associated to the dangerous obstacles: $\mathcal{G} = [\mathcal{N}(\boldsymbol{\mu}_i^{\boldsymbol{w}}, \boldsymbol{\Sigma}_i)]_{i\in[\![1,n]\!]}$. In order to define ellipsoids to represent the obstacles, a probability threshold is set for the cumulative distribution function of the Gaussian distribution. From this threshold, a length r_L is computed using the chi-square inverse cumulative distribution function so that the normal distribution of mean $\boldsymbol{\mu}^{\boldsymbol{w}}$ and covariance $\boldsymbol{\Sigma}$ is represented by the ellipsoid defined by the points $\mathbf{x^w}$ obeying:

$$r_L^2 \geq (\mathbf{x^w} - \boldsymbol{\mu}^{\boldsymbol{w}})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x^w} - \boldsymbol{\mu}^{\boldsymbol{w}}). \quad (3)$$

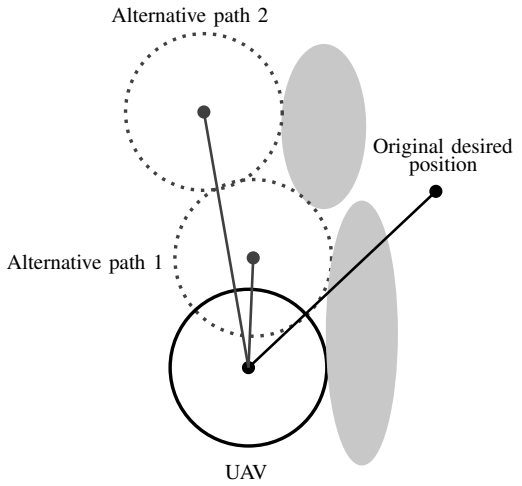

Fig. 2. Illustration of different possibility of trajectories for avoiding the obstacles. The proposed approach selects the path according to Algorithm 2

There are two constraints on the filtered position $\mathbf{x_{rf}^w}(t)$: the path between $\mathbf{x_{UAV}^w}(t)$ and $\mathbf{x_{rf}^w}(t)$ should be collision free and $\mathbf{x_{rf}^w}(t)$ should be as similar to the original desired position $\mathbf{x_r^w}(t)$ as possible. This second criterion is explained below.

The problem is illustrated in Fig. 2. The filled ellipsoids represent the obstacles and the solid circle is the UAV. The best direction to get closer to the desired position depends on the criterion that is optimized: minimizing the proximity to the original objective results in the first path, while maximizing the length of the free path would lead to the second path. A third strategy is to pass below the obstacle, which provides the shortest path. However, this would imply traveling in a direction that is very different from the one specified originally: the UAV would modify its direction by more than $90°$. This illustrates the trade-off between optimal path and the will of the user. Given the teleoperation context, we argue that it makes more sense to produce a command consistent with the input of the user rather than the absolute shortest path. It is hypothesized that an algorithm providing an intuitive position objective might reduce the workload of the user while reducing the task completion time. In particular, this means that the search for an alternative path is limited to a zone reasonably close from the original direction specified by the user.

Note that the safe position computed by OAST is based on a local analysis of the environment of the UAV, meaning that it is possible to inspect dead ends or corners: the operator has fine control over the trajectory of the UAV.

The proposed algorithm aims to find a trade-off between minimizing the angle between the current travel direction and the new one, and maximizing the obstacle free distance

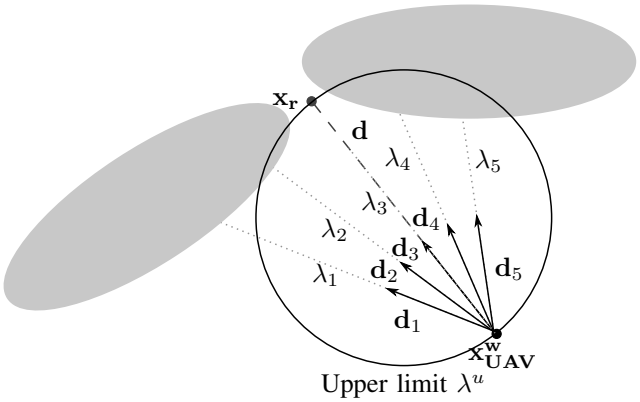

Fig. 3. Illustration of the alternative directions considered for $\theta_{\max} = 30°$ and $\theta_s = 15°$

along the new direction of travel. It is important to note that only a change of norm and direction is considered for the direction of travel. In other words, only paths along a straight line are investigated. The main reason for this is the context of teleoperation. Given the high rate of incoming desired positions, more complex trajectories would only be followed for a very short time, until the next user input, which limits their usefulness. For this same reason, the next position cannot be visualized by the operator since the pilot is supposed to continuously lead the UAV.

The proposed method is divided in three main steps:

1) Directions of interest are chosen using a regular grid in spherical coordinates.
2) For each direction, the distance to the closest obstacle in this direction is computed.
3) The direction that maximizes the distance to the closest obstacle while minimizing the angle with the original direction is chosen as the new direction of travel.

In the following, the UAV is represented as a sphere of radius r_{UAV}. Let $\mathbf{d} = \mathbf{x_r^w} - \mathbf{x_{UAV}^w}$ be the desired direction of travel.

Let $\theta_{\max} < 90°$ represent the maximum angle between $\mathbf{d}$ and the new direction that this algorithm aims to compute. The extreme directions defined by $\mathbf{d}$ rotated by $\pm\theta_{\max}$ are called $\mathbf{d}_u$ and $\mathbf{d}_l$, respectively. An angular step θ_s is chosen to discretize the directions between $\mathbf{d}_u$ and $\mathbf{d}_l$. The resulting set of normalized directions is called $\mathcal{S}_\mathbf{d}$.

For each element $\mathbf{d}_i$ of $\mathcal{S}_\mathbf{d}$, the maximum distance λ_i that the UAV can cross before being in collision with an obstacle is computed. This step is illustrated in Fig. 3. In this drawing, the limit angle $\theta_{\max}$ is $30°$ and the step θ_s is $15°$. Note that $\mathbf{d}_3$ is aligned with $\mathbf{d}$ in this figure.

The distance $\lambda_{i,i\in[\![1,5]\!]}$ is computed for each direction $\mathbf{d}_{i,i\in[\![1,5]\!]}$ so that λ_i is the distance available to the UAV along $\mathbf{d}_i$ before a collision occurs. Computing this distance efficiently is not trivial and is the subject of the next section.

2) Computation of the obstacle free distance in a direction: Let $d_e(\mathbf{x^w}, \boldsymbol{\mu^w}, \boldsymbol{\Sigma})$ be the distance between a point x and an ellipsoid of mean $\boldsymbol{\mu^w}$ and covariance $\boldsymbol{\Sigma}$. The line describing the trajectory of the UAV can be described by $\mathbf{x_{UAV}^w} + \lambda\mathbf{d}$ for a real λ. In order to find the distance before a collision between the obstacle and the UAV, the following equation has to be solved for λ:

$$r_{\text{UAV}} = d_e(\mathbf{x_{UAV}^w} + \lambda\mathbf{d}, \boldsymbol{\mu^w}, \boldsymbol{\Sigma}). \tag{4}$$

As noted in the work of Hart [27], solving d_e involves solving a polynomial of degree 6 which cannot be done analytically. Iterative methods are however available, as a Newton approximation can be used [27]. In order to solve the problem in real time, the proposed algorithm instead increases each ellipsoid by the radius of the UAV. The result is not an ellipsoid, but it is nonetheless approximated by the closest ellipsoid, with has the same mean and eigenvectors of the original one, but different eigenvalues. This allows an analytical solution to be found.

The new eigenvalues are computed from the enlarged ellipsoid using 3 points on the original ellipsoid, called ellipsoid fitting parameters. In the rest of this paper, each of those points is described by the two corresponding parametric angles $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi]$ in the coordinate frame of the ellipsoid. The complete algorithm to compute the new eigenvalues b_1, b_2, b_3 of the covariance matrix of the approximated increased ellipsoid using the eigenvalues a_1, a_2, a_3 of the initial covariance matrix and the ellipsoid fitting parameters (i.e. three pairs of angles $(\theta_j, \varphi_j)_{j\in[\![1,3]\!]}$) is presented in Algorithm 1.

Algorithm 1: Algorithm to compute the eigenvalues of an increased ellipsoid

Function incEl$\left(\left[a_1, a_2, a_3\right]^t, r_{\text{UAV}}, (\theta_j, \varphi_j)_{j\in[\![1,3]\!]}\right)$**:**
  $\mathbf{D} = \begin{bmatrix} 1/a_1 & 0 & 0 \\ 0 & 1/a_2 & 0 \\ 0 & 0 & 1/a_3 \end{bmatrix}$;
  for $i \leftarrow 1$ **to** 3 **do**
    $\mathbf{l}_i = \begin{bmatrix} \cos(\theta_i)\cos(\varphi_i) \\ \cos(\theta_i)\sin(\varphi_i) \\ \cos(\theta_i) \end{bmatrix}$;
    $\mathbf{p^c}_i = \sqrt{r_L^2 a_i} \circ \mathbf{l}_i$;
    $\mathbf{p}_i = \mathbf{p^c}_i + r_{\text{UAV}} \frac{\mathbf{Dp^c}_i}{\|\mathbf{Dp^c}_i\|}$;
    $\mathbf{q}_i = \mathbf{p}_i \circ \mathbf{p}_i$; // The Hadamard product is noted $\circ$
  end
  $\mathbf{A} = \left[\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3\right]^t$;
  $\mathbf{k} = \left[r_L^2, r_L^2, r_L^2\right]^t$;
  Solve the linear system $\mathbf{Av} = \mathbf{k}$;
  return $\left[1/\mathbf{v}(1), 1/\mathbf{v}(2), 1/\mathbf{v}(3)\right]^t$;

Those new eigenvalues combined with the eigenvectors from the covariance matrix $\boldsymbol{\Sigma}$ of the original Gaussian distribution provide a new covariance $\boldsymbol{\Sigma}^*$ matrix representing an ellipsoid that approximates the original ellipsoid increased by the radius of the UAV. The initial problem of finding the distance before a collision in a given direction $\mathbf{d}$, can now be solved by finding the intersection between the trajectory of the UAV and the increased ellipsoid. In other words, (4) can be replaced by:

$$r_L^2 = (\mathbf{x_{UAV}^w} + \lambda\mathbf{d}_i - \boldsymbol{\mu^w})^t \, \boldsymbol{\Sigma}^{*-1} \, (\mathbf{x_{UAV}^w} + \lambda\mathbf{d}_i - \boldsymbol{\mu^w}). \tag{5}$$

As mentioned above, (5) cannot be used directly because the original ellipsoid that is enlarged is not an ellipsoid.

(5) is a second degree polynomial in λ, which can be solved to get the distance before a collision would occur in the direction of $\mathbf{d}_i$. To summarize, for each element $\mathbf{d}_i$ of $\mathcal{S}_\mathbf{d}$, the maximum distance λ_i that the UAV can cross before being in collision with an obstacle is calculated by solving (5).

3) Computation of the filtered position objective: Using the distances before collision in each direction $\mathbf{d}_i$, the filtered position objective can now be computed. The final suitable direction, named $\mathbf{d}^*$, is chosen among the $\mathbf{d}_i$.

An upper bound λ_i^u is introduced on the distance along a direction $\mathbf{d}_i$:

$$\lambda_i^u = \mathbf{d}_i \cdot \mathbf{d}. \tag{6}$$

Should the length λ_i in a direction be large, this upper limit prevents the UAV from going too far in a given direction, depending on how different this direction is from $\mathbf{d}$. This upper limit has been drawn in black in Fig. 3 for all possible directions. On this figure, λ_1 is higher than the upper limit, meaning the UAV will be limited in direction $\mathbf{d}_1$, even though the obstacle would allow the UAV to go further.

For all directions $\mathbf{d}_i$, the ratio r_i is computed so that:

$$r_i = \min\left(1, \frac{\lambda_i}{\lambda_i^u}\right). \tag{7}$$

If this ratio is equal to 1, it means that the space available in the considered direction has reached the limit. In contrast, if this ratio is less than 1, then an obstacle is putting a constraint on the movement in this direction. Since the objective is to search for free paths, the considered directions are all $\mathbf{d}_i$ whose ratio r_i is equal to 1. If there is none, the direction with the highest ratio r_i is chosen. If there are several directions $\mathbf{d}_i$ so that r_i is equal to 1, then $\mathbf{d}^*$ is the one that is the closest to $\mathbf{d}$ (i.e. the one that maximizes λ_i^u).

In other words, $\mathbf{d}^*$ is the direction that:

1) leads to enough space for its λ_i to reach the projection of $\mathbf{d}$ on itself,
2) has the smallest angle with $\mathbf{d}$.

Given an initial set of directions and norms $(\mathbf{d}_i, \lambda_i)_{i\in[\![1,n]\!]}$, the computation of the alternative direction $\mathbf{d}^*$ and the length λ^* is summarized in Algorithm 2.

Once λ^* and $\mathbf{d}^*$ are computed, the new filtered position objective $\mathbf{x}_{\mathbf{rf}}^{\mathbf{w}}(t)$ is computed as:

$$\mathbf{x}_{\mathbf{rf}}^{\mathbf{w}}(t) = \mathbf{x}_{\mathbf{UAV}}^{\mathbf{w}}(t) + \lambda^* \mathbf{d}^* \tag{8}$$

F. The haptic feedback module

The objective of the haptic feedback is to facilitate the interaction between the operator and the obstacle avoidance system. Since the proposed obstacle avoidance algorithm modifies the desired position provided by the user, the feedback provides a haptic representation of this modification. As such, the role of the haptic feedback is purely informational.

More precisely, the feedback is a force proportional to the difference between the desired position $\mathbf{x}_\mathbf{r}^\mathbf{w}$ and the filtered position $\mathbf{x}_{\mathbf{rf}}^{\mathbf{w}}$. The user can then feel how the algorithm modifies the input, which in turn helps understanding the resulting movement of the UAV.

Algorithm 2: Algorithm to compute an alternative direction given a set of directions and the maximal distance before collision along each of them

```
p* = 0 ;                    // The best projection
r* = 0 ;                    // The highest ratio
for i ← 1 to q do
    λ_i^u = d_i · d;
    if λ_i^u is too small then continue;
    r_i = min(1, λ_i / λ_i^u);
    if r_i ≥ r* then
        if r* ≥ 1 then
            if λ_i^u > p* then
                p* = λ_i^u, λ* = λ_i^u, d* = d_i;
            end
        else
            r* = r_i, λ* = λ_i, d* = d_i;
        end
    end
end
return λ* d*;
```

The haptic feedback uses two tuning parameters: the maximal force that can be displayed f_m and the maximal distance l_m between $\mathbf{x}_\mathbf{r}^\mathbf{w}$ and $\mathbf{x}_{\mathbf{rf}}^{\mathbf{w}}$ that would display this force. Let $\mathbf{d}_s = \mathbf{x}_{\mathbf{rf}}^{\mathbf{w}} - \mathbf{x}_\mathbf{r}^\mathbf{w}$. The equation governing the displayed force feedback $\mathbf{f^j}_u$ is:

$$\mathbf{f^j}_u = \begin{cases} f_m \frac{\mathbf{d}_s}{l_m}, & \text{if } \|\mathbf{d}_s\| < l_m, \\ f_m \frac{\mathbf{d}_s}{\|\mathbf{d}_s\|}, & \text{else.} \end{cases} \tag{9}$$

G. Implementation notes

The whole system is separated in two programs. The first one is the 3D haptic controller, which runs on the computer of the operator and sends data to the UAV. The second program runs on the UAV. It is composed of the command interpreter, the UAV controller, the map and the obstacle avoidance module. Our system receives two external types of data: the point clouds from the lidar and the odometry. Since this paper aims to evaluate the obstacle avoidance scheme independently and odometry computation is outside the scope of this paper, it is desirable to decouple the odometry generation from the proposed architecture. For this reason, an external source of odometry is used as a ground truth in the following developments rather than the odometry from the NDT OM framework.

IV. Testing of the obstacle avoidance module

In order to demonstrate the results of the algorithm described in Section III-A, several tests are performed in simulated environments.

A. Adopted parameters, software and hardware

The obstacle are detected by a simulated lidar scanner, the VLP-16 [28] to which Gaussian noise with zero mean and a

TABLE I
VALUES OF THE PARAMETERS USED IN THE SIMULATION

Parameter	Description	Value
$\theta_{\max}$, $\phi_{\max}$	Maximum angle for alternative path search	$80°$
θ_s	Angle step between the directions for alternative path search	$5°$
$\mathbf{K}_s$	Scaling matrix converting joystick coordinates into position objective	$\mathrm{diag}(1.0)$
r^*	Radius of the neutral sphere in joystick workspace	0.2
	Radius of the joystick workspace	1
r_{UAV}	Radius of the UAV with safety margin	60 cm
f_m	Maximal force of the haptic feedback	3.6 N
l_m	Maximal distance between $\mathbf{x}_\mathbf{r}^\mathbf{w}$ and $\mathbf{x}_\mathbf{rf}^\mathbf{w}$ considered in haptic feedback computation	1 m
	Map width and depth	15 m
	Map height	10 m
	NDT map resolution	0.3 m

standard deviation of 1 cm is added. This lidar is a realistic sensor to be embedded on the small UAVs considered, due to its light weight and small size. Unless mentioned otherwise, the values of all the parameters used in the simulations of this paper are described in Table I.

The algorithms are implemented in C++ as described in Fig. 1. The interface between sensors are managed using the Robot Operating System (Kinetic) [29] middleware. The operating system is Ubuntu 16.04 64 bits. The simulator used is Gazebo [30], and the package which implements the UAV model is RotorS [31]. The computer used has an Intel Core i7 6700 processor, an NVIDIA Quadro K2200 graphic card and 16 GB of RAM.

The UAV itself is a Pelican from Asctec, with default characteristics as defined by the RotorS package [31]. The radius of the UAV model is approximately 32.4 cm.

B. Verification of OAST performances

In this section, note that the haptic controller is not used, the input is being provided by the simulation directly.

The runtime of OAST mainly depends on the number of Gaussian distribution to analyze. The worst observed runtime in our testing was inferior to 30 ms per iteration.

1) Alternative path when facing a single wall: The first environment created in Gazebo is composed of a single wall 4 m away from the starting position of the UAV. The desired trajectory $\mathbf{x}_\mathbf{r}^\mathbf{w}(t)$ is defined by (the zero y component has been hidden to improve readability):

$$\mathbf{x}_\mathbf{r}^\mathbf{w}(t) = \begin{cases} \begin{bmatrix} 0 & 2\min(1, t/8) \end{bmatrix}^t, & \text{if } 0\,\mathrm{s} \leq t < 15\,\mathrm{s}, \\ \begin{bmatrix} t-15 & 2+(t-15)/10 \end{bmatrix}^t, & \text{if } 15\,\mathrm{s} \leq t < 20\,\mathrm{s}, \\ \begin{bmatrix} 5 & 2.5 \end{bmatrix}^t, & \text{if } 20\,\mathrm{s} \leq t. \end{cases} \tag{10}$$

Note that the wall is 2.5 m high and the maximum desired height caps at 2.5 m as well. This means that the UAV should collide with the wall without any assistance (the UAV should arrive at the wall around the 19 s mark, with an altitude of 2.4 m according to (10)).

The actual trajectory of the UAV is plotted in Fig. 4 against the desired trajectory $\mathbf{x}_\mathbf{r}^\mathbf{w}(t)$ described by (10) and the filtered trajectory $\mathbf{x}_\mathbf{rf}^\mathbf{w}(t)$ computed by Algorithm 2 and (8).

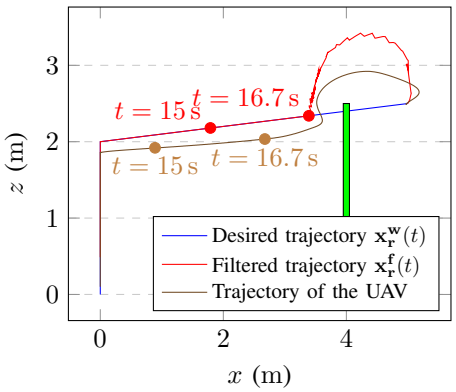

Fig. 4. Behavior of the alternative path finding algorithm in a simple scenario involving a single wall. The UAV trajectory, input and algorithm output are plotted. The wall is plotted in green

Although the desired trajectory passes through the wall, the UAV successfully avoids the obstacle. The UAV first starts by advancing toward the wall. The proposed algorithm modifies the desired trajectory when the UAV arrives too close to the wall: since the direction of travel is upward and forward, the closest free path is upward. The OAST architecture thus makes the UAV go upward, and tilts the direction of travel forward as soon as the wall does not obstruct this direction anymore.

This simple example highlights a desirable characteristic of the proposed approach: when blocked against a planar surface, the generated trajectories will typically slide against the obstacle in a direction that is close to the desired direction. This makes maneuvering around obstacles easier and allows the user to be less precise: in a narrow corridor, even if the user does not input a direction of travel perfectly aligned with the walls, the UAV will still advance as if this was the case.

2) Alternative path when following a corridor: This specific case is illustrated with a second simulated environment, involving two parallel walls separated by 1.3 m, with the UAV flying in between them. The desired position is given by:

$$\mathbf{x}_\mathbf{r}^\mathbf{w}(t) = \begin{cases} \begin{bmatrix} 0 & 0 & 2t/10 \end{bmatrix}^t, & \text{if } 0\,\mathrm{s} \leq t < 10\,\mathrm{s}, \\ \begin{bmatrix} 1.2\,(t-10) & -1.1 & 2 \end{bmatrix}^t, & \text{if } 10\,\mathrm{s} \leq t < 13\,\mathrm{s}, \\ \begin{bmatrix} 1.2\,(t-10) & 1.1 & 2 \end{bmatrix}^t, & \text{if } 13\,\mathrm{s} \leq t < 16\,\mathrm{s}, \\ \begin{bmatrix} 7 & 0 & 2 \end{bmatrix}^t, & \text{if } 16\,\mathrm{s} \leq t. \end{cases} \tag{11}$$

In other words, the UAV takes off then goes forward, but with a lateral component that changes direction at $t = 13\,\mathrm{s}$. The trajectory of the UAV is presented in Fig. 5.

It is shown that in this case, the UAV remains in between the two walls despite the desired position being behind the walls. Moreover, the UAV slides along the walls as discussed above.

3) Behavior of the algorithm with a fixed input in a complex environment: Finally, a more complex case is examined in a third environment. The third environment is depicted in Fig. 6. This time, a desired position is not provided, but a simulated joystick input is sent to the UAV. This joystick input is simply $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ after take off, meaning that the only command of the UAV is to go forward. Note that the walls are placed in

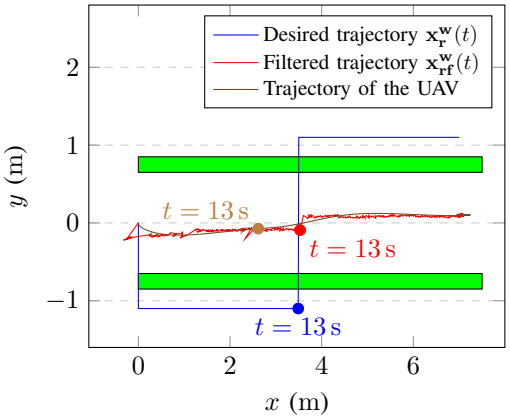

Fig. 5. Behavior of the alternative path finding algorithm in a simulated environment involving two walls. The walls are plotted in green

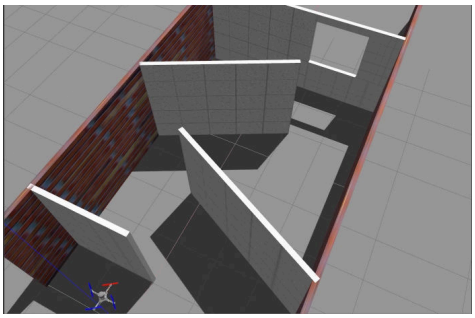

Fig. 6. Picture of the third simulated environment composed of multiple walls and a window

the way of the UAV and the final window available in the wall is slightly shifted compared to the position of the incoming UAV.

The resulting trajectory of the UAV and the filtered position are shown in Fig. 7. It is shown that the UAV is able to successfully avoid the obstacles on its way. It is possible to differentiate between two types of behavior from this figure: the very first wall encountered by the UAV is exactly perpendicular to the desired direction of travel. OAST is able to avoid this wall since the free space on the right is inside the cone of search (this zone is represented in Fig. 3), similar to what is observed in Fig. 4. The second type of behavior occurs when a solution is not directly visible, which is the case for the second and third walls encountered by the UAV. In those circumstances, the preferred direction computed by Algorithm 2 is along the wall since the wall is slightly slanted (similar to what is observed in Fig. 5). Those results illustrate a limitation of OAST: should the walls be slanted the other way around, the UAV might not pass the obstacle (depending on its position when encountering the walls) and end up in a dead end. This is however by design: since the user might want to actually inspect those dead ends, there is a necessary trade off between the assistance provided by the algorithm and the liberty of movement made available to the user. The results shown in Fig. 7 illustrate to which degree the user's input is amended. Recall that the input from the joystick is fixed as going forward in the x direction. The $1.3\,\mathrm{m}$ wide window is a challenging obstacle with regard to the fixed input provided.

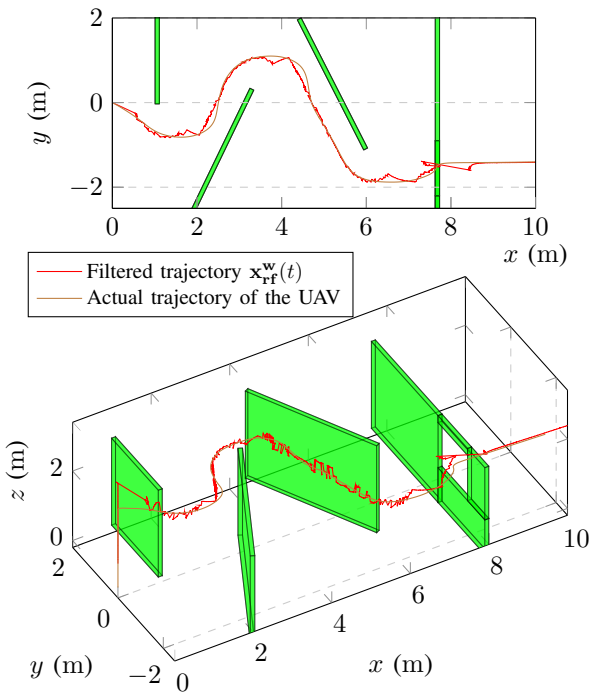

Fig. 7. Behavior of the alternative path finding algorithm in the third simulated environment. The trajectory of the UAV is plotted as well as the filtered position computed by our algorithm. The walls are plotted in green except the lateral ones. Two views are shown. The input from the joystick is constant equal to $\begin{bmatrix}1 & 0 & 0\end{bmatrix}$ after take off

V. Experimental validation of OAST with human operators

The last section shows that OAST is able to avoid obstacles on the path of the UAV. As the proposed flight assistance system is designed for teleoperation, it is important to investigate its impact on human performances. To this end, a computerized experiment involving human was performed.

A. Hardware and software

The software and hardware used in these experiments are described in Section IV-A.

The haptic controller used is the omega.3 from Force Dimension. The workspace of this device has dimensions of $11\,\mathrm{cm}$, $12\,\mathrm{cm}$ and $7\,\mathrm{cm}$ along the x, y and z axes respectively. Note that this controller only has 3 degrees of freedom, the yaw is controlled with the keyboard. The operator has access to two keys on the keyboard to increase or decrease the desired yaw by $5°$. The updated yaw is then transmitted to the UAV controller.

The library used to interface the haptic controller is HAPI version 1.3 along with the official SDK for the omega.3 version 3.7.3.

For these experiments, a forward facing camera with a resolution of 640×480 pixels and a field of view of $80°$ is placed on the UAV. The video feed from this camera is shown to the operator.

B. Scenarios

Three scenarios were developed, called S1, S2 and S3. These scenarios are shown in Fig. 8. The third scenario is

(a) S1 (b) S2 (c) S3

Fig. 8. Different scenarios used in the experiment (ceiling is not visible)

longer and combines elements from the first two. It is expected to be more challenging. Indeed, the windows of the third scenarios are $1.3\,\mathrm{m}$ wide while some corridors are only $1.4\,\mathrm{m}$ wide. This makes the teleoperation of the UAV challenging given its diameter of around $65\,\mathrm{cm}$.

C. Experimental protocol

The experiments involved 20 people from Cranfield University. Each participant was requested to give informed consent before participating and this research was approved by the Cranfield University Research Ethics System. During the briefing, each participant was asked, for each scenario, to achieve the following two objectives:

1) minimize the number of collisions between the UAV and the environment,
2) reach the end of the scenario as fast as possible.

Should a collision occur, the UAV would be frozen for $5\,\mathrm{s}$, then brought back where it was $5\,\mathrm{s}$ before the collision.

Two schemes are tested: with OAST as described in Section III-E (AS) and without this system (no AS). In other words, in the first case, $\mathbf{x}^{\mathbf{w}}_{\mathbf{rf}}$ is sent to the UAV controller while in the second case $\mathbf{x}^{\mathbf{w}}_{\mathbf{r}}$ is sent instead.

The parameters recorded to assess the performance of the schemes are the number of collisions (N_{col}), the time to complete the run (T_c) and the average distance to the closest lidar point over a scenario (D_{avg}). The subjective workload is assessed using the NASA Task Load Index (TLX) [32] with pairwise comparison between the weights.

Each participant is first given $5\,\mathrm{min}$ to get familiar with the controls in a training scenario, without the obstacle avoidance system. Then, OAST is activated in the training scenario and the user gets $5\,\mathrm{min}$ to get familiar with it as well. A test session is composed of 6 runs with or without the obstacle avoidance system: the first three are practice runs (one per scenario) and the last three are the recorded runs. At the end of a test session the participant fills a NASA TLX form for this session. The runs are grouped by session for two reasons: to minimise the learning effect of going through the same scenario back to back, and to allow participants to have a better appreciation of the scheme for the NASA TLX. Each participant does two sessions, one with OAST and one without. In order to get a full factorial experiment, the number of participants is even and the order of the sessions changes between each participant. This protocol yields 6 runs and 2 NASA TLX evaluations per person.

TABLE II
RESULTS OF A FULL FACTORIAL ANOVA ON THE TWO FIXED EFFECTS: METHOD AND SCENARIO

Fixed effect	N_{col}	D_{avg}	T_c	TLX
Method	***	·	***	***
Scenario	***	***	***	*NA*
Interaction method × scenario	·	***	**	*NA*

p-values are indicated by '***', '**', '*' and '·' if respectively $p \leq 0.001$, $0.001 < p \leq 0.01$, $0.01 < p \leq 0.05$ and $p > 0.05$ (not significant).

D. Experimental results

Results were analysed with the R language [33], using generalized linear mixed models with the *lme4* [34] package. Each parameter was modelled separately using the scenarios and methods as fixed factors plus an intercept. The participants were modelled as random effects. The confidence level used for statistical significance is 0.95. The initial model was created by including an interaction term between method and scenario, which was dropped if its contribution to the model was not shown to be significant with the ANOVA. If the algorithm proves having a statistically significant influence, post-hoc tests are carried out using package *lsmeans* [35] to determine the nature of this influence per scenario. Unless mentioned otherwise, each generalized linear mixed model is using a family of Gaussian distribution.

The *p*-values from the full factorial ANOVA are presented in Table II. For these experiments, the method factor has two levels (with the avoidance system and without it) and the scenario factor has three levels.

1) Safety metrics - number of collisions and average minimal distance to an obstacle: In order to avoid detecting the propeller's blades with the lidar, the minimum distance reported by the lidar is $35\,\mathrm{cm}$.

The numbers of collisions are presented in Fig. 9. The number of collision with OAST activated is always zero, except for one collision in scenario 1. Without OAST, the number of collisions depends on the scenario. The third scenario, expected to be the most difficult, contains the highest number of collisions. Since the number of collisions is a count, it is modeled as a Poisson distribution. The reduction in collisions provided by the algorithm is statistically significant ($p < 0.0001$) for each scenario. It is interesting to note that while human operators are sensitive to the difficulty of the scenario, the algorithm is not.

Moreover, it is important to remember that the simulated sensor used to scan the environment is a lidar which could realistically be embedded on a small UAV [28]. In particular, this lidar has a vertical field of view of $30°$. This experiment shows that, given perfect odometry, such a restricted field of view still allows obstacle detection in tasks that involve

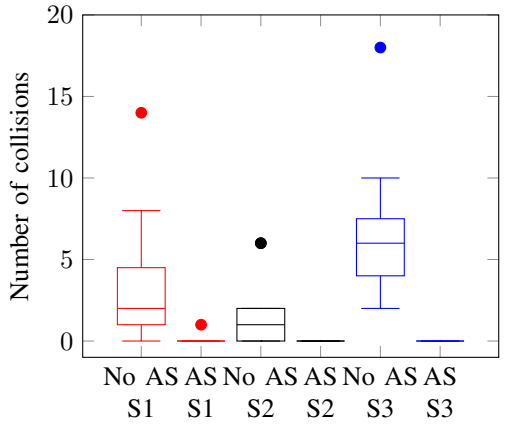

Fig. 9. Number of collisions with and without OAST. Two outliers of 67 and 25 collisions in scenario 3 without OAST are omitted from this graph

verticality (scenarios 2 and 3 in particular). In other words, the trajectories taken by the participants allowed a sensor with a vertical field of view of 30° to capture enough volume to successfully avoid collisions in a closed environment.

The average distance to the closest lidar point shows no clear trend, in contrast with what would be expected if a method based on potential fields was used, which tends to maximize the distance to obstacles. With the obstacle avoidance system, the median is lower in scenario 1 and 2 but higher in scenario 3. The ANOVA shows no significant influence of the algorithm on the average distance to the closest lidar point ($p = 0.82$) although a significant interaction term is present between the presence of the algorithm and the scenario. Post-hoc test reveals a significant difference in scenario 3 by a small margin (the lower confidence level with the obstacle avoidance algorithm is $1.28\,\mathrm{m}$ against an upper confidence level of $1.27\,\mathrm{m}$ without the algorithm). The objective is to allow for as much liberty as possible for the operator, which means that the average distance to the closest obstacle should not be expected to increase because the movement pattern of the operator should not change significantly. This distance might decrease if the operator explicitly relies on the avoidance scheme to steer the UAV in the right direction while keeping as close to an obstacle as possible.

Such an analysis is difficult to perform from the averaged distance to the closest lidar point alone, so histograms of the distance to the closest lidar point for all participants are shown in Fig. 10. Those histograms show that the UAV spends slightly more time close to obstacles when OAST is activated. This suggests that participants did not only rely on the avoidance system to perform emergency avoidance, they also rely on it to perform trajectory planning.

2) Efficiency metric - time elapsed: The time to complete each scenario is presented in Fig. 11. This figure shows that the avoidance system reduces the time needed to complete the task in all three scenarios. This reduction is statistically significant for the three scenarios (p-values for scenarios 1,2 and 3 are respectively 0.0001, 0.0008 and < 0.0001) and amounts to $39\,\%$, $40\,\%$ and $45\,\%$ for scenarios 1, 2 and 3, respectively. Moreover, the variance of this duration is reduced as well for all scenarios as seen on Fig. 11. This suggests that the obstacle

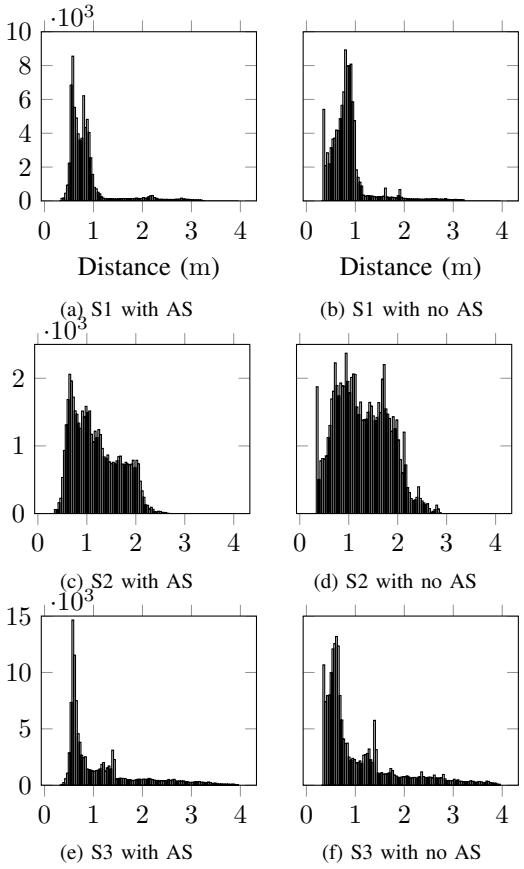

Fig. 10. Histograms of the distance to the closest lidar point for all users with (on the left) and without (on the right) OAST

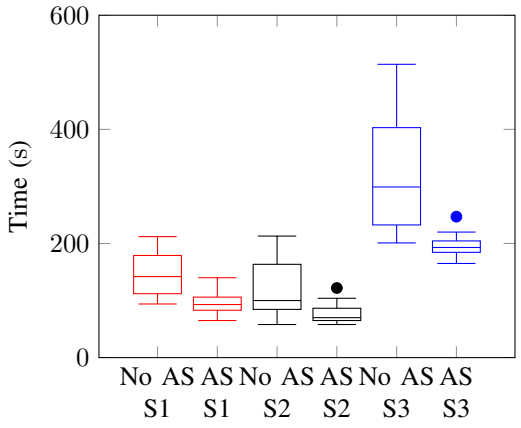

Fig. 11. Time needed to complete the task with and without OAST. An outlier of 794 s for S3 with no AS is omitted to improve clarity.

avoidance algorithm negates the differences in difficulty of the scenario and skills of the user.

E. Workload : NASA TLX results

The results of the NASA TLX evaluation are provided in Fig. 12. The full test (including pairwise comparison of the factors) was performed. The NASA TLX score is modeled using a linear mixed model with a family of Gaussian distribution. The package and conditions for the least square mean post-hoc test are described in Section V-D. The workload is shown to be reduced by the obstacle avoidance algorithm and

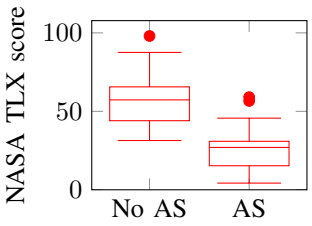

Fig. 12. NASA TLX workload score with and without the obstacle avoidance system

this reduction is statistically significant (p-value < 0.0001), from 57 points to 27 points of NASA TLX score according to the post-hoc test.

F. Interpretation of the results

These experiments show that OAST improves safety by reducing collisions. OAST also reduces the time required to complete the scenarios. Those elements are consistent with the reduction of workload that is observed with the algorithm. Indeed, considering some of the NASA TLX metrics: a reduction of the number of collisions is likely to reduce frustration, a safety net might reduce mental demand and a lower time of completion is likely to improve the performance feeling.

The reason behind those improvements is considered to be the liberty of movement of the operator, which could explain the faster completion of the scenarios, lower workload and similar average distance to the closest lidar point. This however cannot be generalized based on those experiments.

It is interesting to note that the proposed algorithm seems to suppress the difference of skills between the participants: the number of collisions and the time needed to complete the scenarios have a reduced variance compared to runs performed without the algorithm. A possible explanation is that the advantages of OAST are greater when the skill of the operator is lower or the situation is more difficult to handle. Considering the number of collisions and the time elapsed, the difference made by the algorithm is more important in scenario 3 than in the simpler scenario 2.

VI. Conclusion

In this paper, an integrated flight assistance assistance system for intelligent teleoperation, OAST, is proposed. OAST amends the user input to prevent collisions with the environment. The new trajectory computed by OAST aims to remain consistent with the original trajectory in order to reduce the workload of the operator and the task completion time. The algorithm is extensively tested in a simulated environment through a human experiment involving inexperienced operators. In this experiment, OAST is shown to improve both the safety of the UAV and the efficiency of the operator, while reducing the incurred workload.

An interesting direction for future work is the implementation on a physical UAV which would fuse the odometry coming from the map with the one computed from the onboard sensors to allow real time operation.

References

[1] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," *Autonomous Robots*, vol. 11, no. 1, pp. 9–18, Jul. 2001.

[2] O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[3] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in UAV teleoperation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 6, pp. 1316–1330, Nov. 2009.

[4] T. M. Lam, M. Mulder, M. M. Van Paassen, J. A. Mulder, and F. C. Van Der Helm, "Force-stiffness feedback in uninhabited aerial vehicle teleoperation with time delay," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 3, May 2009.

[5] H. Courtois and N. Aouf, "Haptic feedback for obstacle avoidance applied to unmanned aerial vehicles," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2017, pp. 417–424.

[6] A. M. Brandt and M. B. Colton, "Haptic collision avoidance for a remotely operated quadrotor UAV in indoor environments," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, Oct. 2010, pp. 2724–2731.

[7] R. M. Philbrick and M. B. Colton, "Effects of haptic and 3d audio feedback on operator performance and workload for quadrotor UAVs in indoor environments," *Journal of Robotics and Mechatronics*, vol. 26, no. 5, pp. 580–591, Feb. 2014.

[8] T. Lam, M. Mulder, and M. Van Paassen, "Haptic interface for UAV collision avoidance," *International Journal of Aviation Psychology*, vol. 17, no. 2, pp. 167–195, Dec. 2007.

[9] R. Mahony, F. Schill, P. Corke, and Y. S. Oh, "A new framework for force feedback teleoperation of robotic vehicles based on optical flow," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 1079–1085.

[10] M. Hua and H. Rifaï, "Obstacle avoidance for teleoperated underactuated aerial vehicles using telemetric measurements," in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 262–267.

[11] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Bilateral haptic teleoperation of an industrial multirotor UAV," in *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe*, F. Röhrbein, G. Veiga, and C. Natale, Eds., Cham: Springer International Publishing, 2014, pp. 301–320.

[12] X. Hou and R. Mahony, "Dynamic kinesthetic boundary for haptic teleoperation of VTOL aerial robots in complex environments," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 5, pp. 694–705, May 2016.

[13] M. Odelga, P. Stegagno, and H. H. Bülthoff, "Obstacle detection, tracking and avoidance for a teleoperated UAV," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2984–2990.

[14] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 2247–2252.

[15] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conference on Decision and Control (CDC)*, Dec. 2010, pp. 5420–5425.

[16] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, Sep. 2013.

[17] T. Stoyanov, M. Magnusson, and A. J. Lilienthal, "Comparative evaluation of the consistency of three-dimensional spatial representations used in autonomous robot navigation," *Journal of Field Robotics*, vol. 30, no. 2, pp. 216–236, eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21446.

[18] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part i - the first 30 years and fundamentals," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[19] F. Fraundorfer and D. Scaramuzza, "Visual odometry : Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics Automation Magazine*, vol. 19, no. 2, pp. 78–90, Jun. 2012.

[20] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Found. Trends Robot*, vol. 4, no. 1, pp. 1–104, May 2015.

[21] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[22] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 15–22.

[23] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *CoRR*, vol. abs/1607.02565, 2016. arXiv: 1607.02565.

[24] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[25] A. Segal, D. Hähnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems*, J. Trinkle, Y. Matsuoka, and J. A. Castellanos, Eds., The MIT Press, 2009.

[26] M. Magnusson, "The three-dimensional normal-distributions transform: An efficient representation for registration, surface analysis, and loop detection," PhD thesis, Örebro universitet, 2009.

[27] J. C. Hart, "Distance to an ellipsoid," in *Graphics Gems II.1*. P. S. Heckbert, Ed., Academic Press, 1994, pp. 113–119.

[28] *Velodyne vlp-16 lite*, http://velodynelidar.com/vlp-16-lite.html, Accessed: 2017-05-05.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, Jan. 2009, p. 5.

[30] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, Sep. 2004, 2149–2154 vol.3.

[31] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS - a modular Gazebo MAV simulator framework," in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, Ed. Cham: Springer International Publishing, Feb. 2016, pp. 595–625.

[32] S. G. Hart and L. E. Stavenland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," in *Advances in Psychology, Human Mental Workload*, P. A. Hancock and N. Meshkati, Eds., vol. 52, Elsevier, 1988, ch. 7, pp. 139–183.

[33] R Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013.

[34] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting linear mixed-effects models using lme4," *Journal of Statistical Software*, vol. 67, no. 1, pp. 1–48, Oct. 2015.

[35] R. V. Lenth, "Least-squares means: The R package lsmeans," *Journal of Statistical Software*, vol. 69, no. 1, pp. 1–33, Jan. 2016.

Hugo Courtois has received his Ph.D. degree from the Centre for Electronic Warfare information and Cyber in Cranfield University, U.K. in 2019. His research interests include haptic technologies, mapping and sensor fusion.

Nabil Aouf is currently the lead of the Robotics and Machine Intelligence activities at City University of London, U.K. He leads the Robotics, Autonomy and Machine Intelligence (RAMI) group. He has authored more than 180 publications in high caliber in his domains of interest.His research interests include aerospace, information fusion and vision systems,guidance and navigation, tracking, and control and autonomy of systems. Prof. Aouf is an Associate Editor of four journals including an IEEE Transaction Journal.

Kenan Ahiska received B.S., M.S., and Ph.D. degrees in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2010, 2012, and 2016, respectively. He has background on control theory and its applications. He has works on modeling, guidance control and navigation of unmanned vehicles, optimal and model predictive control of robotic systems. Since 2018, he has been a research fellow in guidance and control in Cranfield University, Defence and Security at Shrivenham, the U. K.

Marco Cecotti received the PhD degree in electrical engineering from Oxford Brookes University, UK, in 2013. He worked for Tata Motors and Dyson on several automotive projects, focused on vehicle control and driver assistance systems. He is now a Lecturer with the School of Aerospace, Transport and Manufacturing at Cranfield University, UK. His research interests include vehicle trajectory control, path planning, localisation and sensor fusion. He is a member of the IEEE.

OAST: Obstacle Avoidance System for Teleoperation of UAVs

Courtois, Hugo

2022-01-27

Courtois H, Aoufy N, Ahiska K, Cecotti M. (2022) OAST: Obstacle Avoidance System for Teleoperation of UAVs. IEEE Transactions on Human-Machine Systems, Volume 52, Number 2, April 2022, pp. 157-168

https://doi.org/10.1109/THMS.2022.3142107