Medeiros *et al. Robomech J* (2021) 8:14
https://doi.org/10.1186/s40648-021-00200-w

ROBOMECH Journal

RESEARCH ARTICLE Open Access

3D pointing gestures as target selection tools: guiding monocular UAVs during window selection in an outdoor environment

Anna C. S. Medeiros[1*], Photchara Ratsamee[2], Jason Orlosky[2], Yuki Uranishi[2], Manabu Higashida[2] and Haruo Takemura[2]

Abstract

Firefighters need to gain information from both inside and outside of buildings in first response emergency scenarios. For this purpose, drones are beneficial. This paper presents an elicitation study that showed firefighters' desires to collaborate with autonomous drones. We developed a Human–Drone interaction (HDI) method for indicating a target to a drone using 3D pointing gestures estimated solely from a monocular camera. The participant first points to a window without using any wearable or body-attached device. Through the drone's front-facing camera, the drone detects the gesture and computes the target window. This work includes a description of the process for choosing the gesture, detecting and localizing objects, and carrying out the transformations between coordinate systems. Our proposed 3D pointing gesture interface improves on 2D interfaces by integrating depth information with SLAM and solving ambiguity with multiple objects aligned on the same plane in a large-scale outdoor environment. Experimental results showed that our 3D pointing gesture interface obtained average F1 scores of 0.85 and 0.73 for precision and recall in simulation and real-world experiments and an F1 score of 0.58 at the maximum distance of 25 m between the drone and building.

Keywords: Gestural interface, Human–Drone interaction, Gesture development process, Pointing gesture, Object selection

Introduction

In emergency scenarios such as fires, a group of professionals called first responders arrives on the scene to gather as much information as possible about the current situation. In particular, certain pieces of information are critical to evaluating the situation, many of which can only be obtained from an aerial view. One of our first steps was to talk to the firefighters at the Kobe Fire Academy and determine what their primary needs were. We found that certain information, e.g., apartment type, gas leaks, the presence of living beings, could help improve the chances of a better outcome if found more quickly.

A first responder sometimes uses manually controlled unmanned aerial vehicles (UAVs) to get a top view of the situation, but it is tough for a UAV pilot to navigate inside a building without direct visual contact. Autonomous UAVs can potentially perform this type of task.

Before a UAV gets inside a building, a point of entry, like a window, has to be chosen. Human decisions should guide this window choice because the information gathered will first come from that entrance area, and it is up to the first responder team to access the situation and choose which area to investigate first.

The present work involves an outdoor environment, where a UAV should understand which window is selected by the human so that the UAV can go inside

*Correspondence: anna@lab.ime.cmc.osaka-u.ac.jp
[1] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, Japan
Full list of author information is available at the end of the article

and perform a given task. Controlling a UAV in interactive systems depends on the available resources and requirements of the task at hand. For example, wearable devices might not be the optimal solution for long jobs where they would run out of batteries [1]. Other options could be the traditional Command-Line [2]: every time the UAV required a human input in the middle of a task, they could provide it by typing a command. However, the downsides to this are clear, as the human might need to interrupt their activities to communicate with the UAV. One possible solution that could mitigate the need for wearable devices and prevent significant interruptions would be the use of gesture input.

The process of associating a gesture to a particular action or function of the system is not trivial, because it must take into account several factors such as ergonomics, intuitiveness, and objectivity [3, 4]. In past works, we analyzed which gesture fits best the purposed scenario, using our Gesture Development Process [5], validating it in general tasks and with a diverse group of users [6], and an elicitation study with the targeted users [7].

The most basic feature of Human–Drone interaction is a user or system signaling a UAV to change its position [8]. The present work presents a system that enables UAVs equipped with a monocular camera to determine which window of a building was selected by a human user, in large scale, indoors or outdoors. Figure 1 illustrates the purpose. We developed two applications: one using only 2D information (no depth information), and another using 3D information captured from a point cloud obtained from a monocular simultaneous localization and mapping (SLAM) system. Experimental evaluation is made available in simulation and also in a real-world setting. The list of contributions include:

- Human–Drone interaction elicitation study on firefighters' experience in a first-response situation to discover a suitable gesture.
- Development of a monocular-based 3D pointing gesture interface proposed application. We include object recognition and modify ORB-SLAM's output to include the current frame's respective point cloud to precisely estimate the target location in a large-scale environment.
- Verification and validation of our proposed method on simulated and real-world (large-scale) fire brigade scenario. We also provide a dataset of pointing to windows outdoors.

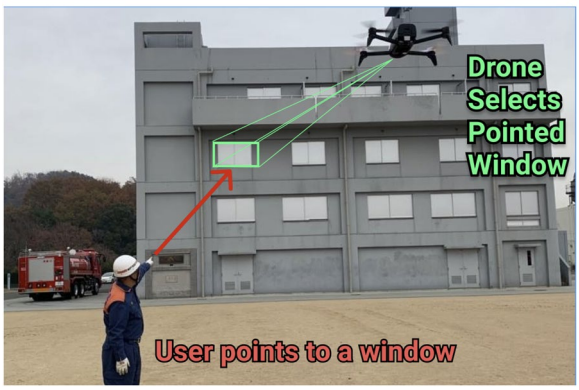

Fig. 1 UAV selects the pointed window. This collaboration scenario could enable first responders to gather more information about a specific location

Research background

Defining the pointing gesture

In the present work, we use the gesture for "Select" previously presented in [5] in a very specific domain of application: as a collaboration interface between UAVs and Firefighters. To double-check that the same gesture could be used for "Select" in this particular environment, we performed an elicitation study [7] with seven Firefighters.

The first part of the study consisted of a UAV hovering next to a building in different positions. Each participant was requested to signal the UAV to enter a window. In the second part of the Study, an interview was conducted with each firefighter. We collected their opinions on future applications of drones in firefighting and interfaces for drones control systems.

All participants wished to collaborate using natural interactions with an autonomous UAV that could quickly arrive at the fire scene and gather useful data for first responders such as building type, fire source, type of fire, and victims present. Results (Fig. 2) showed that all firefighters used a pointing gesture and voice commands to indicate a specific window within an average of 3 s.

Given the results of this elicitation study the author became interested in building a gesture interface that used Pointing Gestures to indicate a window to a drone. Besides the scenario here presented, there are many possible scenarios where drone systems would benefit from interaction with humans in the course of a task. This work aims to provide guidance for natural Human–Drone interactions when relying solely on the UAV camera, without external devices, in large-scale interaction scenarios. Pointing Gestures are studied as a means to estimate the correct Target Object.

Fig. 2 An elicitation study with firefighters showed that pointing gesture and voice commands were unanimously used to specify windows to a UAV

Related work

Gesture recognition [9] has been studied with various devices such as RGB cameras, depth cameras [10, 11], radar-based sensors [12], capacitive sensors [13], sensorized gloves [14], electromyography sensors [15–18], Wi-Fi [19] and others. Each device has its own limitations, and in this section we will summarize the use of these devices on detecting pointing gestures and/or robot control.

The work of Tolgyessy [20] investigates the use of pointing gestures to give target locations to Roomba-like robots. It uses an RGBD sensor (Kinect) mounted on a 2-DoF robot to interpret the pointing gesture from a person standing 2–3 m from the setup. They used third-party software to get the body points from the user. In order to choose which data to use to determine the pointing gesture, they investigated combinations of wrist-elbow, wrist-shoulder, and wrist-head. Wrist-elbow gave a better result in terms of accuracy. The limitations were that the user was always facing the robot, distance was limited (up to 3 m), and the user had to hold the pointing gesture with one hand while the other hand performed another gesture that enabled the robot to act on the pointing gesture. The user also had to point to a place inside the field-of-view (FoV) of the robot, which limited the pointed places to very short distances in the indoor environment. The system was not suitable for outdoor use because of Kinect's hardware limitations.

The work of DelPreto [16] presents a gesture-based interface for human-robot interaction using wearable sensors. "A real-time clustering algorithm processes streaming data while only maintaining a key subset of observations. This building block is used to adaptively threshold muscle and motion signals, detecting arm stiffening and fist clenching from EMG signals and rotation gestures from inertial measurement unit (IMU) signals. These predictions are fused with IMU processing to interpret left, right, up, or down gestures." In this work, participants used the interface for real-time robot control by remotely piloting a UAV around obstacles. Some of the sensors used here required the participants to be connected through cables to a data acquisition (DAQ) device, thus restricting the applicability of this means outside of a controlled environment. This means of interaction also won't provide enough input to infer location to a robot; therefore, it cannot be used to detect pointing gestures.

The work of Liu [21] presents a method for automatically detecting the teacher's pointing gesture on the electronic whiteboard. They detect pointing gestures inside

an indoor short-ranged environment, like a classroom. They use third-party software to identify body poses, and image form filtering to identify the whiteboard, if the hand point is inside the whiteboard, the system identifies the teacher pointing out the whiteboard. This system is limited to an indoor environment where the location of the individual is very near the whiteboard (target) so that they overlap.

The work of Gromov [22] uses a pointing gesture to control UAV movement until a desired location. The experiment takes place indoors on a small environment. This system uses an IMU sensor on one arm and the other arm holds a button to control the UAV's movements. It is not possible to point directly to the desired location. The change in the user's arm position indicates change in the UAV's positioning.

The work of Chen [23] uses a model of an environment to plan the trajectory of an UAV, using Augmented Reality (AR). Chen's system requires users to import a 3D scene for performing their desired drone trajectory plan, therefore the environment must be previously known. It also requires a planar surface to place the 3D model of the scene, before the trajectory planning can start. The user must move the mobile device around the model to define its trajectory. The need for a direct manipulation of a tablet or smartphone is another constraint in this work.

The work of Chen [24] uses mobile AR devices to allow for UAV position control. The user drags a UAV's model cast shadow or a slider bar on the touchscreen to plan the UAV's trajectory. Chen L.'s system uses an ad-hoc tracking system with QR code-based visual markers on the floor. Therefore the drone flying area is quite limited due to the need of visual markers on the floor. The system Can be used in indoor or outdoor environments but requires the total attention of the user to a mobile device, occupying user's both hands.

There are multiple studies that research the use of gestures to navigate a UAV [16, 20, 22, 25]. However, we investigate the collaboration between UAVs and humans to relieve human task load and reduce UAV's error by processing periodical human-input. This collaboration takes shape in an outdoor environment, where long distances make the use of depth cameras difficult due to hardware limitations. So the present work is focused not only on short-scaled interactions (like [11, 16, 20–22]), where the user, gesture capturing device (camera) and interaction targets are within 2 m of each other. But also on large-scaled interactions where larger distances are observed either between user-camera or user-targets. Moreover, lidars like Hokuyo [26] cannot gather enough information on the pointing gesture and building's windows simultaneously. Furthermore, we cannot use wearable devices like electromyography sensors to determine where a user is pointing, or mobile devices like smartphone and tablets, as this could also impair dexterity or task performance in a firefighting environment. The use of visual markers on a first response scenario is also unrealistic given the time constraints. Therefore, the present work focuses on using a monocular camera to produce a system that can enable correct window selection decisions in indoor and outdoor environments without using external devices, even on low-cost UAVs. Table 1 shows a comparison between the present and related work.

Methods

We explain the proposed pointing gesture detection methods in this section. We define a pointing gesture as "a gesture in which the user extends their arm away from their body, towards the desired target object (window) to be selected." In previous efforts [7], we built a naive solution to detect pointed objects, that solution had limitations resulting from the lack of depth information usage. In this work, we address such limitations by integrating depth information from a monocular SLAM algorithm (ORB-SLAM with modified output). An overview of both application scenarios can be seen in Fig. 3a.

Table 1 Comparison of related works

Work	Gesture	GDP	Indoor	Outdoor	Werable	Depth cam	Monocular cam	Short-scale	Large-scale
[11]	✓	–	✓	–	–	✓	–	✓	–
[20]	✓	–	✓	–	–	✓	–	✓	–
[16]	✓	–	✓	–	✓	–	–	✓	–
[21]	✓	–	✓	–	–	–	✓	✓	–
[22]	✓	–	✓	–	✓	–	–	✓	–
[23]	–	–	–	✓	✓	–	✓	✓	✓
[24]	✓	–	✓	✓	✓	–	✓	✓	–
Present	✓	✓	✓	✓	–	–	✓	–	✓

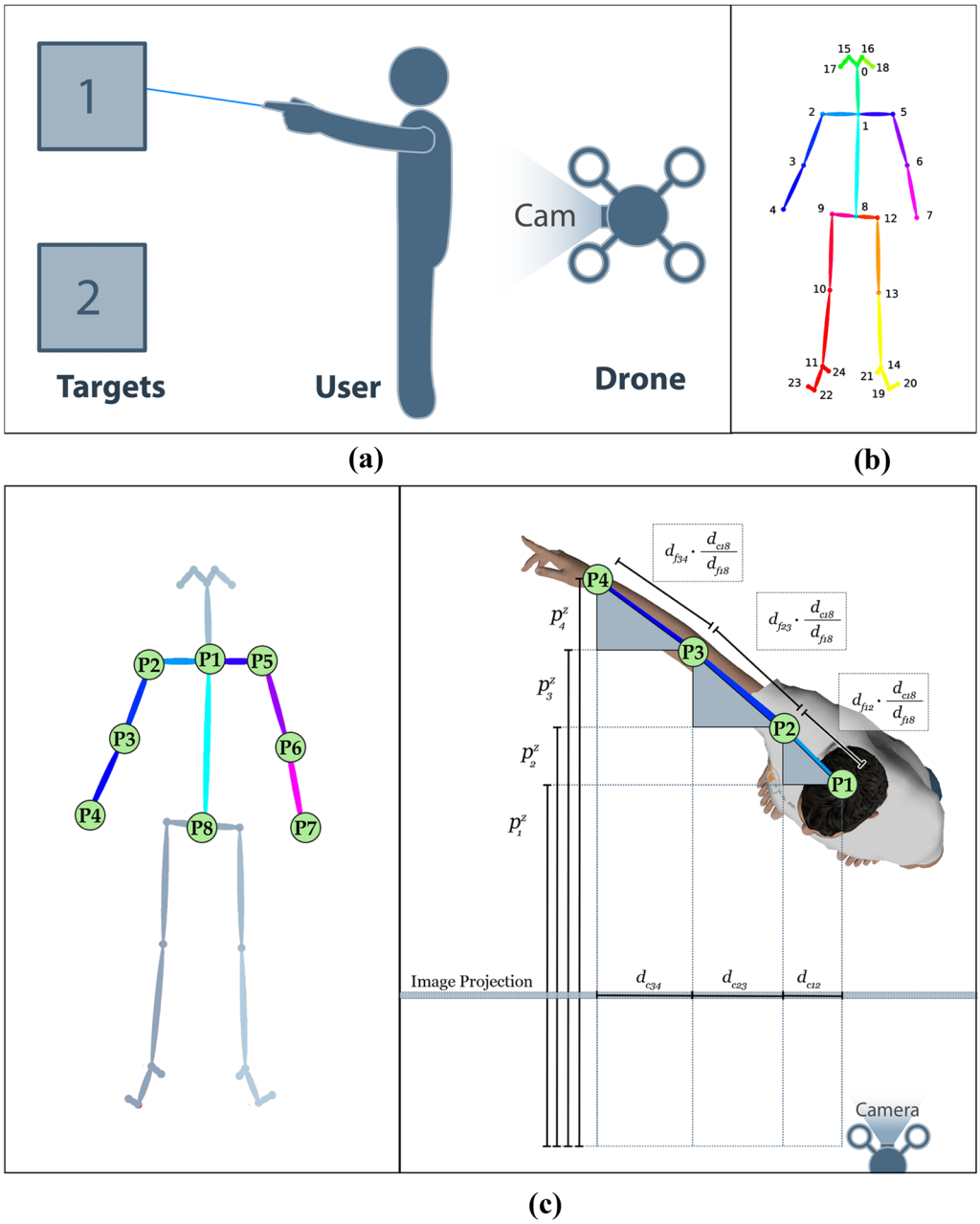

Fig. 3 **a** Scenario Overview There are four essential elements: the user, the targets, camera, and processing unit. The user interacts with the camera in the form of pointing towards the desired target. The targets are types of visual content. For example, this includes a window that the processing unit has been trained to recognize using object detection algorithms. The processing unit is any computational system that processes the images from the camera and sends signals to the UAV. The Camera is the image feed in the moving UAV. **b** Pose Output Format: BODY-25. **c** Left-side: Points used from BODY-25 model. Right-side: Intuition on depth estimation of keypoints. When pointing backwards (transverse and sagittal planes), the size of the user's arm on the Image Projection ($d_{c12} + d_{c23} + d_{c34}$) is deformed by perspective. In this scenario, $d_{c12} + d_{c23} + d_{c34}$ is then smaller than $\left(d_{f12} \times \frac{d_{c18}}{d_{f18}}\right) + \left(d_{f23} \times \frac{d_{c18}}{d_{f18}}\right) + \left(d_{f34} \times \frac{d_{c18}}{d_{f18}}\right)$, which is the expected size of the arm when the user is pointing sideways (arms on the frontal plane). Please refer to Eqs. 8, 9, 10. Using a right-angle triangle intuition (see the three grey right angle triangles in the image), we can estimate the depth change between P_1^z and P_2^z, P_3^z, P_4^z

2D pointing gesture interface approach

This approach was first presented in [7], where object detection and pose estimation were combined to decide which object is selected by the user. For Pose Estimation, we used OpenPOSE [27]. This is a real-time multi-person system that detects the human body, hand, face, and foot keypoints on single images. We used OpenPOSE's BODY_25 (Fig. 3b) model for speed and accuracy. To detect objects, we used the YOLOV3-tiny [28], which provides sufficient speed and accuracy to be run on an integrated development kit in the future.

This approach starts by acquiring an RGB image from a UAV. Each monocular image has dimensions represented by a width w and height h. The BODY_25 model provides P_n points where $n \in \mathbb{N}$, $0 \leq n \leq 24$. We only use eight points: P_1 until P_8 (Fig. 3c, left-side). Each point $P_n \in \mathbb{R}^{w \times h}$ has 2D coordinates P_n^x and P_n^y, the pointing gesture is defined by the arm keypoints: P_3, P_4 and P_6, P_7. Notice that P_5, P_6 and P_7 are analogous to P_2, P_3 and P_4, respectively. Two points from the same arm are used to calculate a 2D line equation, which can take the form

$$f(x) = \frac{(P_3^y - P_4^y)}{(P_3^x - P_4^x)} \times (x - P_3^x) + P_3^y \quad (1)$$

for points P_3, P_4. For each arm, a line segment is defined starting from an elbow point like P_3 in the direction of a hand point like P_4 up to the border of the image. The borders of the image can be defined as $f(y) = w$, $f(y) = 0$, $f(x) = h$, $f(x) = 0$, for every $(x, y) \in \mathbb{R}^{w \times h}$. The choice of which border to use is guided by the direction of the pointing gesture.

We use YOLO [28] to detect objects. It produces bounding boxes for the objects it was trained to recognize. The bounding boxes are simply two points $B_{n1} \in \mathbb{R}^{w \times h}$ and $B_{n2} \in \mathbb{R}^{w \times h}$ used to define a rectangle that contains the object on the image.

An object is selected if the line segment crosses the object's bounding box. More details in [7]. In the case that the line segment crosses more than one bounding box, we chose the bounding box that has the center closest to the line segment.

This system works in outdoor and indoor environments. If the user points in the general location of a target object (e.g., window), the system will select it, and there is no need to point directly towards the object. But there are shortcomings: a line crossing multiple windows is ambiguous. If there are multiple windows around the correct one, it is not guaranteed that the pointing line will cross the correct one closest to the center (see Fig. 13a), and it is not guaranteed that the user will point to the center of the window. Therefore, the lack of depth information to make a more informed decision limits this system. In order to overcome that limitation, we propose the 3D pointing gesture interface approach.

3D pointing gesture interface approach

To estimate the correct pointing area, or target object (window), 3D information is required. We can capture the 3D information of a scene in the format of point clouds. Point clouds can be obtained from diverse sources, we chose a monocular simultaneous localization and mapping (SLAM), named ORB-SLAM, that is relatively low cost and that produced a sparse point cloud with enough information for our purposes: obtaining the distances from drone to user and to target.

ORB-SLAM [29] is a feature-based monocular system that operates in real-time in indoor and outdoor environments. We were interested in the sparse point cloud it provides, and we modified its original code to output the point cloud of the feature points detected on the current image frame. By combining object detection and pose estimation algorithms, in our case YOLO and openPOSE, we can then estimate the depth of objects of interest on the current RGB image. This monocular SLAM provides an unscaled point cloud. To scale it, we perform a one-time calibration step as explained below. We also provide a general model, using average user values, in case this calibration process is not feasible. With depth estimated, we can then calculate the pointing gesture direction up to a specific point in the monocular image. A flowchart of this approach is shown in Fig. 4.

This approach starts by acquiring an RGB image from a UAV. Each image frame has a width w and height h. The X axis represents the width, the Y axis the height, with origin on the upper left of the image. Using the sparse point cloud and calibration files we infer an estimation of the Z axis pointing towards the inside of the image.

Calibration

The 3 planes of body motion are the sagittal, frontal, and transverse. The calibration step requires the user to be standing with open arms at the side of the body. The arms should be in the user's frontal plane. We placed the UAV at a fixed distance D_f of 3 m to the user, but other distances could also be used. We also took measurements of the user's height and torso size. A few frames were recorded in this condition. OpenPOSE's BODY_25 model (Fig. 3b) provides 25 keypoints $P_n \in \mathbb{R}^{w \times h}$ where $n \in \mathbb{N}$, $0 \leq n \leq 24$, for each image frame. Each point has 2D coordinates P_{nx} and P_{ny}.

The calibration step is the creation of a file to store the values of 8 of the user's keypoints (in pixel values) at a fixed distance D_f to the UAV. And also store the real measurements of the user's height and torso size (in metric units). The calibration step takes about a minute and

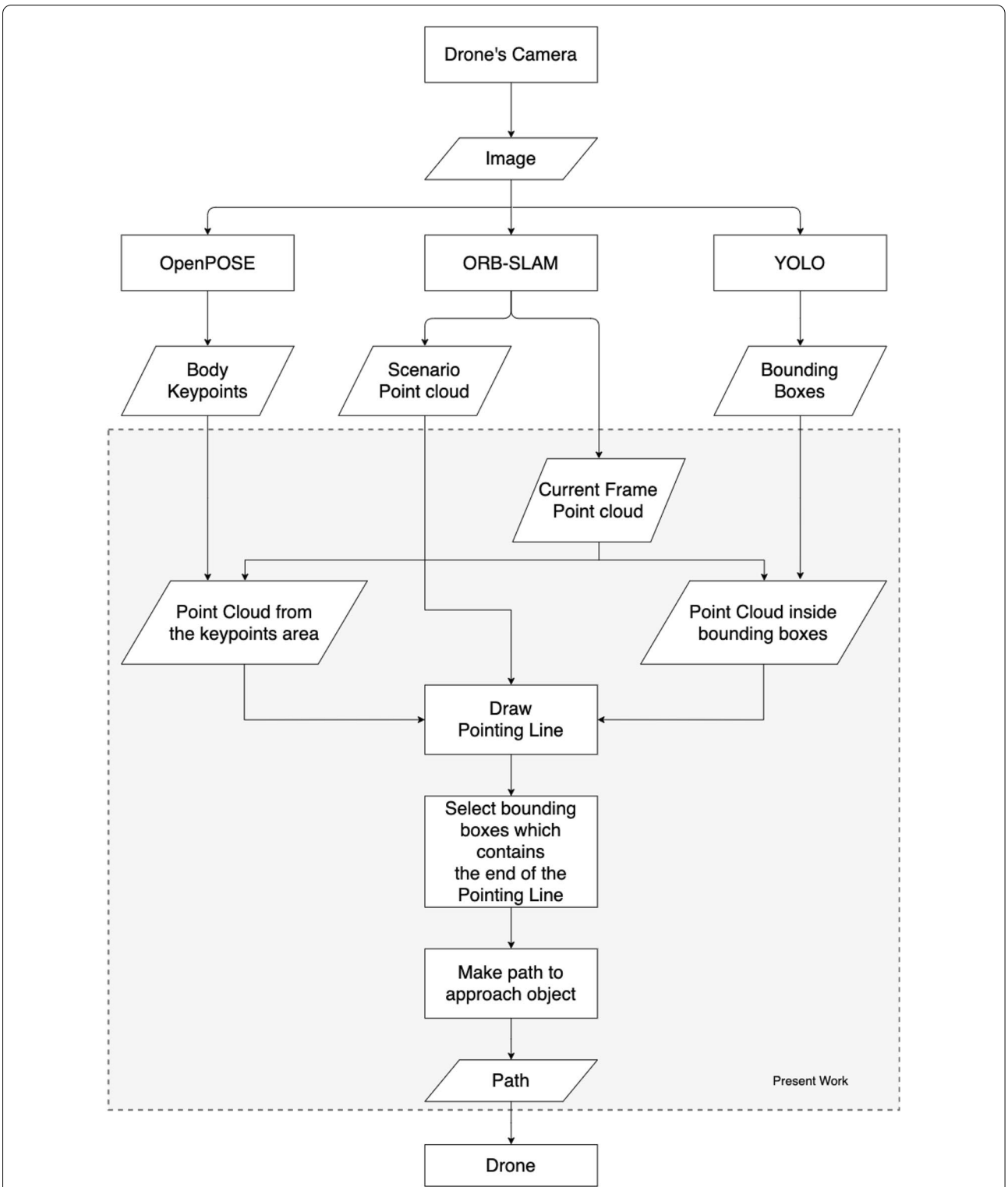

Fig. 4 3D pointing gesture interface application flowchart. ORB-SLAM integrated with Object Detection and Pose Estimation towards pointing gesture interaction

is a one-time step for each user. These 8 keypoints are points P_1, P_2, P_3, P_4, P_5, P_6, P_7 and P_8 in Fig. 3c.

Depth estimation using calibration

In the calibration step we store 8 keypoints from user at a fixed distance D_f from the camera (UAV). More specifically, calibration gives us the size of the user's torso, in pixels, given by the euclidean distance between points P_1 and P_8 (see Fig. 3c):

$$d_{f18}(P_1, P_8) = \sqrt{\left(P_1^x - P_8^x\right)^2 + (P_1^y - P_8^y)^2}, \quad \text{at distance } D_f \tag{2}$$

at a fixed distance D_f to the UAV. Therefore D_f will refer to the distance to the user in calibration time. In a real situation, the user moves away or towards the UAV, therefore the current distance D_c from the user to the UAV's camera (UAV) changes. To estimate D_c we use the current distance d_{c18} between points P_1 and P_8, and the inverse linear relationship between size and distance:

$$D_c = \frac{D_f \times d_{f18}}{d_{c18}} \tag{3}$$

$$d_{c18}(P_1, P_8) = \sqrt{\left(P_1^x - P_8^x\right)^2 + (P_1^y - P_8^y)^2}, \quad \text{at distance } D_c \tag{4}$$

where d_{c18} is the current euclidean distance between points P_1 and P_8 (Fig. 3c), and d_{f18} is the fixed distance between points P_1 and P_8 (Fig. 3c). That is, d_{f18} is the distance between points P_1 and P_8 (Fig. 3c) at calibration time.

When the user moves, the 8 detected keypoints also change. We assume that the user is always standing, so all changes in the image size of the user's torso (d_c) will be affected by the change in the user distance to the UAV (D_c). As can be observed in Fig. 3c, the torso is represented by points P_1 and P_8, the top of the torso to one of the shoulders are represented by points P_1 and P_2, respectively. On that same side, the elbow is P_3, and finally the hand is P_4.

When the user distance to the UAV increases, we observe that the original distance d_{f18} from P_1 to P_8 is decreased by $\frac{D_f}{D_c}$, see Fig. 5 and the rearranged Eq. 3: $d_{c18} = d_{f18} \times \frac{D_f}{D_c}$. If the user has his or her arms in the frontal plane, the current distance from P_1 to P_2 (labeled d_{c12}) is also decreased by $\frac{D_f}{D_c}$. Using Eq. 3 $\frac{D_f}{D_c}$ can be rewritten as $\frac{d_{c18}}{d_{f18}}$. Therefore, the expected value for the current distance d_{c12} can be defined as:

$$d_{c12} = \left(d_{f12} \times \frac{d_{c18}}{d_{f18}}\right), \text{when the user has his or her arms at the frontal plane} \tag{5}$$

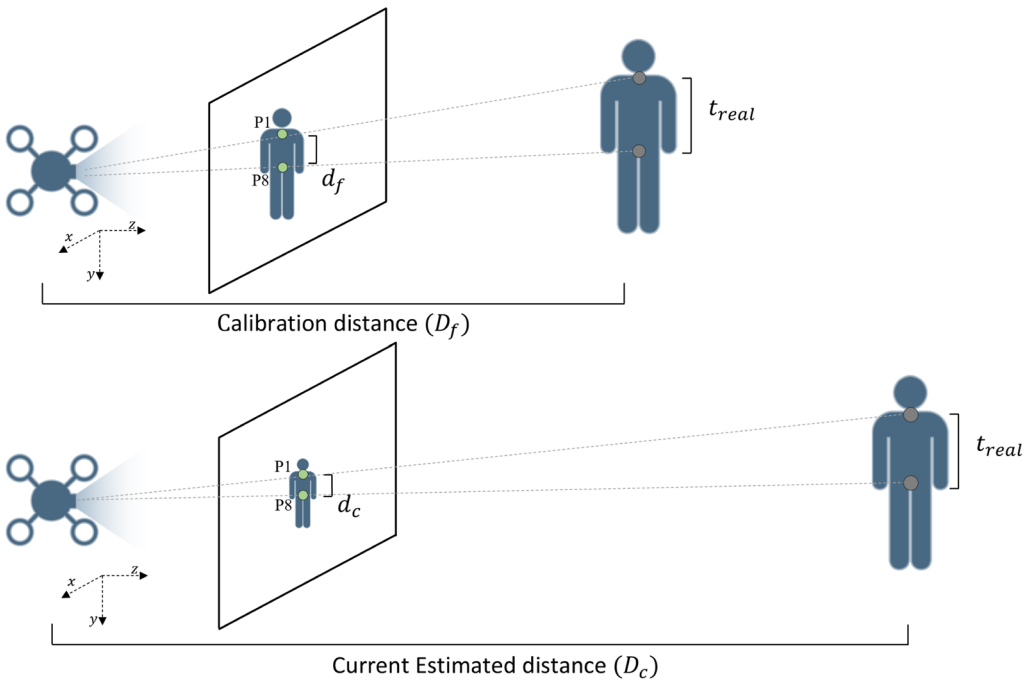

Fig. 5 Relation between distance in the real world and size in the image When the user distance to the UAV increases, the original distance d_{f18} is decreased by $\frac{D_f}{D_c}$

Similarly for d_{c23}, d_{c34}:

$$d_{c23} = \left(d_{f23} \times \frac{d_{c18}}{d_{f18}} \right), \text{when the user has his or her arms at the frontal plane} \tag{6}$$

$$d_{c34} = \left(d_{f34} \times \frac{d_{c18}}{d_{f18}} \right), \text{ when the user has his or her arms at the frontal plane} \tag{7}$$

During the calibration, the user was standing with open arms at the side of the body. The arms were in the user's frontal plane. When the user points towards an object behind his or her body, the size of the user's arm on the image will be shorter than when the user was standing with open arms at the side of the body. This is the cue we use to determine the depth of each point on the arm.

For example, consider the distance from P_3 to P_4, which corresponds to one of the forearms of the user. The distance from P_3 to P_4 at calibration time is d_{f34}, and the current distance is d_{c34}. When the user has open arms at the side of the body, the relationship between d_{f34} and d_{c34} is expressed by Eq. 7. However, when the user points backwards d_{c34} will be less than the expected value of $d_{f34} \times \frac{d_{c18}}{d_{f18}}$.

$$d_{c12} < \left(d_{f12} \times \frac{d_{c18}}{d_{f18}} \right), \text{ when the user is pointing backwards} \tag{8}$$

$$d_{c23} < \left(d_{f23} \times \frac{d_{c18}}{d_{f18}} \right), \text{ when the user is pointing backwards} \tag{9}$$

$$d_{c34} < \left(d_{f34} \times \frac{d_{c18}}{d_{f18}} \right), \text{ when the user is pointing backwards} \tag{10}$$

Therefore, by checking how the current distances (for example d_{c12}, d_{c23}, d_{c34} for one of the arms) behave, we can quantify the change in depth of each point in the user's hands, elbows and shoulders. If the user's arm stays in the frontal plane, the distances in between his arm's keypoints decrease by $\frac{d_c 18}{d_f 18}$. However, when the user points to something behind himself or herself, moving the arm in the transverse and sagittal planes, the projection of the arm on the image plane will be smaller than expected (Fig. 3c).

To quantify the depth of each point, we assume that the depth P_1^z will be the current distance D_c to the UAV. To express that depth D_c in pixel units, we use the proportion from the torso pixel size in calibration time d_{f18} and real size t_{real} in metric units:

$$P_1^z = \frac{D_c \times d_{f18}}{t_{real}} \tag{11}$$

Using a right-angle triangle we calculate the estimated depth for subsequent points P_2, P_3 and P_4. So the estimated depth P_2^z of point P_2 would be the depth from the previous joint point (P_1 in this case) plus the calculation of the right-angle triangle (Fig. 3c), as shown in Eq. 12.

The estimated depth P_2^z for P_2 uses the current distance d_{c12} from P_1 to P_2, the saved distance d_{f12} from P_1 to P_2 at calibration time, the current distance d_{c18} from P_1 to P_8, and the saved distance d_{f18} from P_1 to P_8 at calibration time:

$$P_2^z = P_1^z + \sqrt{\left(d_{f12} \times \frac{d_{c18}}{d_{f18}} \right)^2 - (d_{c12})^2} \tag{12}$$

Similarly for P_3^z, it consider the the preceding points' depths (P_2^z):

$$P_3^z = P_2^z + \sqrt{\left(d_{f23} \times \frac{d_{c18}}{d_{f18}} \right)^2 - (d_{c23})^2} \tag{13}$$

The estimated depth P_4^z for the hand considers the preceding points' depths (P_3^z):

$$P_4^z = P_3^z + \sqrt{\left(d_{f34} \times \frac{d_{c18}}{d_{f18}} \right)^2 - (d_{c34})^2} \tag{14}$$

Analogously, we obtain P_5^z, P_6^z and P_7^z. We now have (x, y, z) coordinates, in pixels, for every point on the arms. At this point, we just need the depth of the building to finish the pointing line, from the user to the building direction.

3D target estimation

Knowing the estimated user distance in the current frame, we can estimate the distance to the building with ORB-SLAM. It is an ORB-feature based SLAM that provides a point cloud in the form of map points and its own definition of keyframes. Map points are the structure for the 3D reconstruction of the scene. Each map point corresponds to a textured planar patch in the world whose position has been triangulated from different views [30]. Points correspond to ORB features in the images, so map points are triangulation of FAST corners. The usual output is the map points for the whole world. We modified the original code to also output all map points corresponding to the ORB features in just the current image. OpenPOSE gives us the position of the user on the current image, and YOLO, the windows. Because we can then get the map points associated with each one, it gives

us the unscaled relationship between the UAV distance to the user and the UAV distance to the building (windows).

Because ORB-SLAM provides unscaled point cloud we need to equate the previously known distance D_c (from user to UAV, in metric units) to ORB-SLAM's user point cloud's average distance. D_c will give a dimension scale to the values of the point cloud. After scaling the point cloud, we can isolate the building point cloud using YOLO's recognition of the windows and estimate the building distance B_c in metric units.

With the building distance B_c known, the product $b_c = \frac{B_c \times d_{f18}}{t_{real}}$ will be the conversion from the distance B_c, in metric system units, to our system Z axis, in pixel units. Given two keypoints from the arm, we calculate the 3D line equation, and find (x_c, y_c) values given b_c as a z value, so for the keypoints P_3 and P_4:

$$\alpha\left(\frac{b_c - P_3^z}{P_4^z - P_3^z}\right) = \frac{\boldsymbol{x_c} - P_3^x}{P_4^x - P_3^x} = \frac{\boldsymbol{y_c} - P_3^y}{P_4^y - P_3^y} \tag{15}$$

α is needed because the 3D line equation belongs in an orthogonal projection, and this system deals with perspective projection. This is a big challenge in a monocular-based long-scaled interaction scenario, where the distance between the user and target of interaction (building's windows) exceeds 3 m. The calculation of α is part of the innovation presented in this work, as it was obtained after numerous tests in simulation. α was observed as:

$$\alpha = \frac{a}{(B_c - D_c) \times b + a} \tag{16}$$

In Eq. 16, a is the maximum width, in metric units, in the image's field-of-view FOV_x at distance D_f of the camera:

$$a = 2 \times D_c \times tan\left(\frac{FOV_x}{2}\right) \tag{17}$$

In Eq. 16, b is the change in the maximum width seen in the image at every change in the distance to the camera:

$$b = 2 \times tan\left(\frac{FOV_x}{2}\right) \tag{18}$$

The field-of-view FOV_x of the image is calculated with respect to the focal length F_x of the camera's Intrinsic Matrix:

$$FOV_x = 2 \times arctan\left(\frac{w}{2 \times F_x}\right) \tag{19}$$

where w is the width of the image frame. Simplifying Eq. 16, α is a scaling factor associated with B_c as follows:

$$\alpha = \frac{D_c}{B_c} \tag{20}$$

So, α can be defined as the proportion between the distance to the camera (D_c) and the building distance (B_c), as demonstrated in Fig. 6. The result of Eq. 15 is then a target point $T_P \in \mathbb{R}^{w \times h}$ where

$$T_P = (x_c, y_c, b_c) \tag{21}$$

If T_P, obtained by Eq. 15, is located inside a window's bounding box, that window is selected. Each bounding box is defined by two points $B_{n1} \in \mathbb{R}^{w \times h}$ and $B_{n2} \in \mathbb{R}^{w \times h}$, a rectangle that contains the object on the image.

In the following section, we describe the experiments performed with the proposed systems. We tested both systems in a simulation environment and real-world scenario.

Experiments and results

We ran experiments in a simulated and real-world environment on both applications: 2D pointing gesture interface and 3D pointing gesture interface. The results are reported in this section.

Simulation setup

We used Gazebo [31] 7.0 to create a simulated environment with a building model, a human model, and a UAV model. We modified all models from Gazebo's originals (Fig. 7). To ensure a variability of positions, the experiment had eight different positions for the human (Fig. 8). In some experiments, the human model executed three fixed pointing angles (Fig. 9), for each one of the eight positions. In this case we wanted to have a test scenario with various "true negative" in the ground truth. In other experiments, the human model would point to four windows for each position. In this other case, the objective was to obtain a ground truth with various "true positives". The UAV stood at a distance of 10 or 20 m from the building, depending on the experiment. We used those two specific distances because they were considered a medium distance from the building, and a long distance from the building (larger-scale interaction scenario). The UAV stood at a fixed height of 2.5 m from the ground.

For each pointing gesture in the Experiments, there is a ground truth associated. This ground truth is either a specific window or that no windows were selected in that case (e.g., pointing to the wall). We considered a true positive when the system correctly identifies the specific window, true negative means that the system correctly identifies when the pointing gesture is aimed at a not-window object or something outside the UAV's view. False positive happens

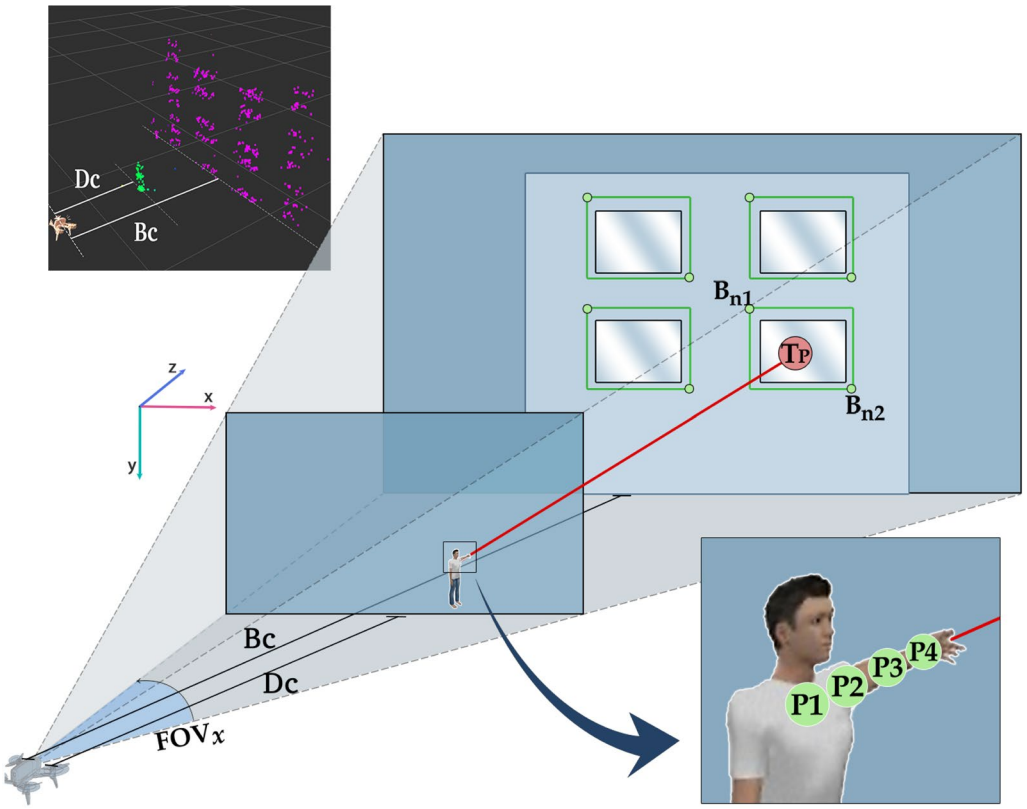

Fig. 6 Slice of perspective projection at current user distance D_c and current building distance B_c. The 3D pointing gesture system outputs a target point T_P that, if inside a detected window area, selects that window

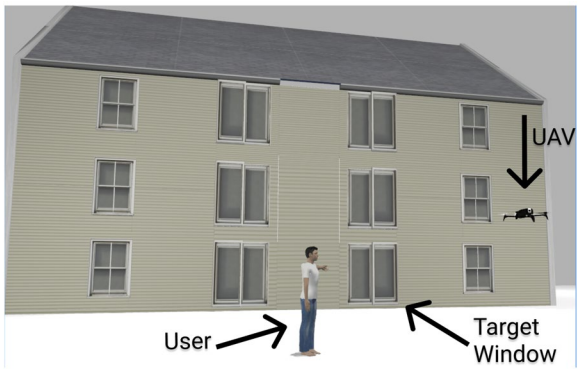

Fig. 7 Simulation environment We used Gazebo 7.0, all models used were modified from Gazebo's originals

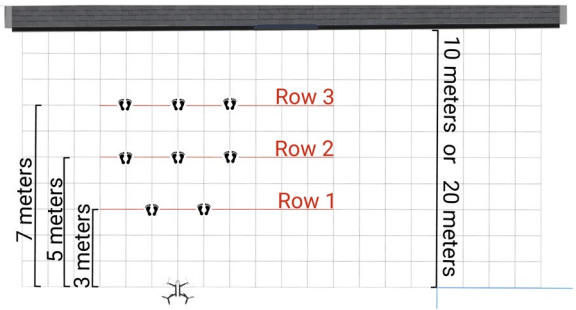

Fig. 8 User's positioning on the experiments. We tested the human model on eight different positions, for each Experiment

when the wrong window is identified, false negative means that there is a window being pointed at but the system wrongly concluded that the user is not pointing to any windows. The results from the technologies used here (OpenPOSE, YOLO, ORB-SLAM) are non-deterministic, so when deciding the outcome of the pointing gesture, we consider a 4 s-window to identify the Mode and make a decision.

2D pointing gesture interface approach result

We tested the 2D approach using the eight positions shown in Fig. 8, the building stood at 10 m from the UAV. This Experiment is labeled "Exp0". For each position, the human model tested all three fixed angles, as shown in

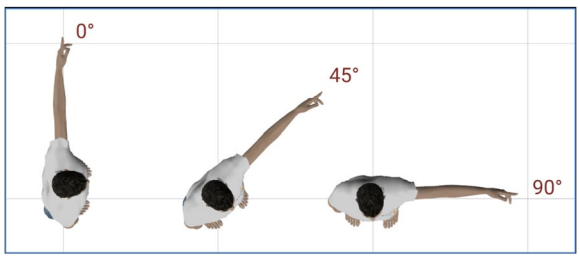

Fig. 9 Fixed angles used. We experimented with fixed angles, that meant that in some positions the human model would be pointing to the wall, or to outside the UAV's view

Fig. 9. This approach could not identify when the human model pointed to not-windows objects; it always tries to select a window. If at least one window bounding box crosses the pointing line, this approach will select it even if the user is pointing at the wall. Results showed 79% of false positives and 21% of true positives. The 2D approach did not produce any false negatives or true negatives. The resulting accuracy was 20%, precision 20%, recall 100%, and F1 Score of 0.34. The Matthew correlation coefficient could not be calculated due to zeroed false negatives and true negatives. The results are summarised on Table 2.

3D pointing gesture interface approach result

We tested the 3D pointing gesture interface approach in three experiments. The first experiment was labeled "Exp1". Exp1 used the eight user positioning shown in Fig. 8, with fixed pointing angles (Fig. 9). The building stood at 10 m of distance from the UAV. Results showed approximately 25% of true positives, 58% of true negatives, 12% of false positives, and 4% of false negatives. The resulting accuracy was 83%, precision was 66%, recall was 85%, F1 Score was 0.75, and Matthew correlation coefficient was 0.63. This result is comparably better than the previous one, so we decided to perform other experiments with the 3D pointing gesture interface approach.

The second experiment was labeled "Exp2". In Exp2, we used eight positions with fixed angles, but the building stood 20 m away from the UAV. We wanted to see how it behaved with targets (windows) further away. Results showed approximately 25% true positives, 58% true negatives, 0% false positives, and 16% of false negatives. The resulting accuracy was 83%, precision was 100%, recall was 66%, F1 Score was 0.75, and the Matthew correlation coefficient was 0.68. The results were slightly better, given the increased distance to the building, but still similar. We decided to perform another experiment to test pointing at the windows with varied angles instead of using fixed angles.

The third experiment was labeled "Exp3". For each one of the eight positions in Fig. 8, the human model would point to all four bottom windows of the building, using angles that best suited each window. Results showed approximately 87% true positives, 0% true negatives, 0% false positives, and 12% false negatives. The resulting accuracy was 87%, precision was 100%, recall was 87%, and F1 score was 0.93. The Matthew correlation coefficient was not applicable due to the lack of negatives in the ground truth. The results were slightly better than the previous experiments. We noticed that when the human model stood at Row 3 (Fig. 8), OpenPOSE would output less than reliable poses. We decided to analyze the data from the previous three Experiments by Row. The results (Table 2) show that indeed the first row had better results than row 2, which had better results than row 3.

When considering all three experiments, results showed approximately 45% of true positives, 38% of true negatives, 4% of false positives, and 11% of false

Table 2 Summary of experiments on simulation environment

	2D Exp0	3D Exp1	3D Exp2	3D Exp3	3D Row1	3D Row2	3D Row3	3D Exp123
UAV-building distance	10 m	10 m	20 m	10 m	N/A	N/A	N/A	N/A
UAV-user distance	N/A	N/A	N/A	N/A	3 m	5 m	7 m	N/A
MCC	N/A	0.639	0.683	N/A	0.891	0.631	0.500	0.701
F1 score	0.344	0.75	0.75	0.933	0.952	0.814	0.740	0.857
Accuracy	0.20	0.83	0.83	0.875	0.94	0.81	0.74	0.84
Precision	0.20	0.66	1	1	1	0.84	0.83	0.91
Recall	1	0.85	0.66	0.87	0.90	0.78	0.66	0.80
True positives	20.8%	25%	25%	87.5%	55.5%	40.7%	37%	45.8%
True negatives	0%	58.3%	58%	0%	38.8%	40.7%	37%	38.8%
False positives	79.1%	12.5%	0%	0%	0%	7%	7%	4.1%
False negatives	0%	4%	16.6%	12.5%	5%	11.1%	18.5%	11.1%

negatives. The resulting accuracy was 84%, precision 91%, recall 80%, F1 Score of 0.85, and Matthew correlation coefficient of 0.70. A sample of successful and failed cases can be seen in Fig. 10.

Real-world experiment

An experiment was conducted at the Kobe Fire Academy, Japan. The experiment's purpose was the identification of the user's pointed window by the UAV. We used the Parrot Bebop 2 [32] for the experiments; This is a quadcopter that weighs around 500 g, offers 25 min of autonomous flight time, and can film in 1080p full HD with its wide-angle 14-megapixel lens. For the sake of performance, the image size was limited to 856 x 480 pixels. We used an available SDK called bebop_autonomy for the ROS-supported system.

The Bebop 2 interfaced via wifi with a laptop. We used an Alienware 13 R3, with an NVIDIA GTX 1060 graphics card, 16 GB RAM, and an i7-7700hq processing unit. Other than the Bebop 2 and Laptop, no other electronic devices were used. They interfaced using the Robot Operating System (ROS) [33].

We asked permission from firefighters and received permission from the Kobe Fire Academy to perform the experiment. Regarding the use of the aerial robot, there are legal restrictions. This research has got permission from the Ministry of Land, Infrastructure, Transport and Tourism, and Osaka Aviation Bureau under license number: P180900923 for performing drone experiments. In subject experiments, the ethics committee for safety subjects it to an ethical review established by Cyber Media Center at Osaka University. From the Ethics Committee, we obtained approval for data use in this subject. Appropriate care was taken to avoid any psychological or physiological distress on the subject during the experiment.

It is challenging to get the exact result of an experiment conducted outdoors as we can not have ground truth like we have with experiments that use motion capture indoors. Therefore, we focus on a classification problem in the present work where the correct pointed window, among an array of windows of a building, should be identified by the proposed 3D interface approach.

During the experiment, a user pointed to multiple windows on a building, and the UAV camera's image was recorded. A total of 6 individuals were able to participate in this setting. Participant age ranged from 23 to 31 years ($M = 28.66$, $SD = 2.94$), 1 participant was female, 50% of the participants had previous experience with manually flying a UAV, and 50% had previously used some form of gesture interface.

Each participant was positioned in 6 different spots from 12 possible spots (see Fig. 11). The 12 spots were chosen to have enough variability of human positions in relation to the building. These 12 spots were distributed in 3 columns and 4 rows in front of the building. The columns were aligned with the left, center, and right sides of the building. Starting from a medium distance, the closest row was 10 m from the building, increasing a distance of 5 m for every other row until the last row. The last row stood at 25 m from the building, providing test cases at a large-scale scenario. For each spot, the user first opened both arms and pointed sideways, at no window, ensuring "true negative" test cases where the user wasn't pointing at anything. Then the user pointed at ten windows, sequentially, 5 s for each window. A total of 432 combinations of windows, users and positions were tested.

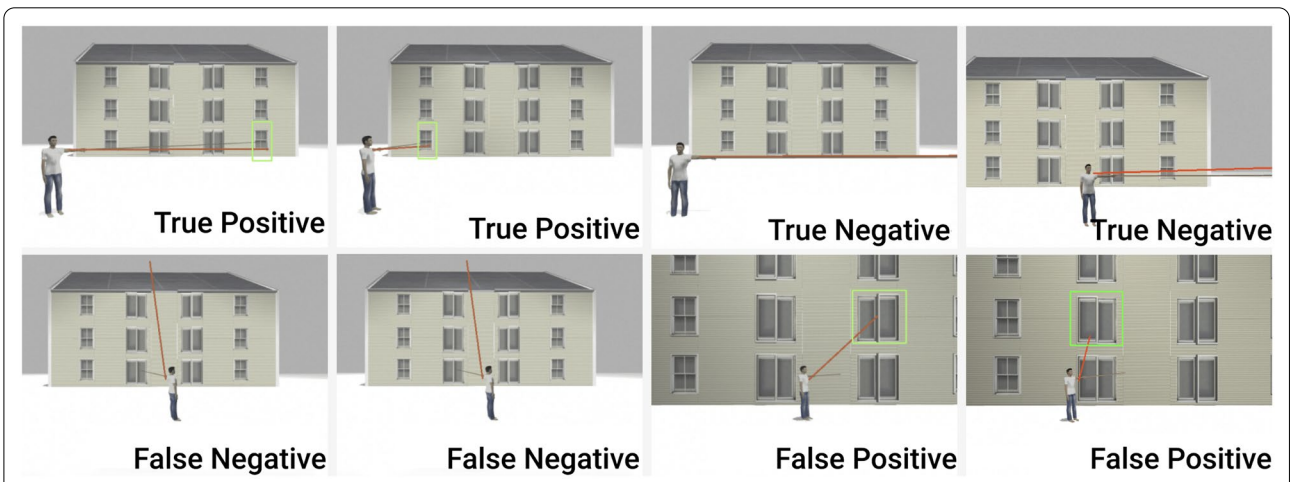

Fig. 10 Sample of simulation experiments' results

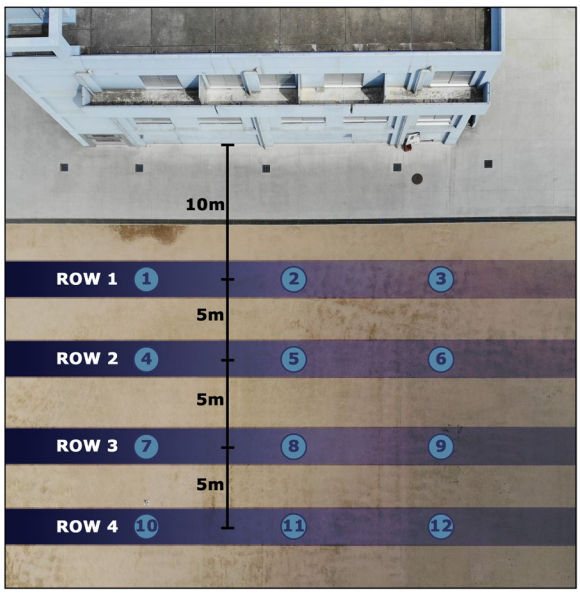

Fig. 11 Top view of real-world experiment

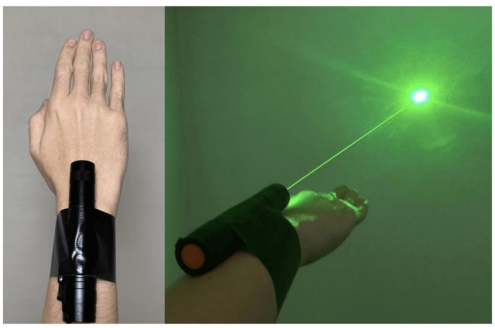

Fig. 12 A Laser pointer was used in the experiments. As ground-truth, each user used a laser pointer to ensure they were pointing to the intended window

To ensure the correct aim, we attached a laser pointer (Fig. 12) to the user's right arm. The laser (3000 mW with a wavelength of 532 nm) allowed the user to point at long-range objects outdoors with precision. In preliminary experiments, we found substantial human error when trying to point without any visual feedback. So we wanted to investigate integrating visual feedback from the UAV's point of view (POV) so that the user can point more precisely and understand the output of the UAV's used system.

The UAV faced the building, with the user inside its view. Sometimes not all windows were present inside the UAV's image view. The UAV stood from 2 to 5 m behind the user and 1.7 m to 3 m high. We used this short distance because at longer distances, the 856 × 480 pixel image would lose details and cause OpenPOSE to output unsatisfactory results. By using a higher image resolution, this distance can be increased accordingly.

We classified each pointed window as true positive if it was the correct window, as false positive if it wasn't the correct window, and as false negative if no window was selected. We used true negative in two cases; the pointed window was outside the image, and the used approach determined that no window was selected. The second case of true negative was when the user was not pointing to any window.

A total of 432 combinations of windows, users, and positions were recorded and later processed using the two proposed approaches.

2D pointing gesture interface approach result

This approach chooses the window with the center closest to the pointing line when crossing more than one window.

The results (Table 3) of the 432 combinations using the 2D approach showed an F1-Score of 0.45 and a Matthews Correlation Coefficient (MCC) of 0.1, which shows that this approach was only slightly better than the neutral points of both metrics. The experiment used an array of windows on the same plane, and this approach worked better when the targets were in different planes, due to the limitation of using only 2D data.

Analyzing the results by row: The first row shows an F1-Score of 0.65 and an MCC of 0.39, the second row an F1-Score of 0.48 and an MCC of 0.23, the third row an F1-Score of 0.39 and an MCC of 0.09, and the fourth row an F1-Score of 0.28 and an MCC of -0.26. So the results worsen the more distant the row from the building,

Table 3 Summary of experiments on real-world environment

	2D R1	2D R2	2D R3	2D R4	2D AllRows	3D R1	3D R2	3D R3	3D R4	3D AllRows
Rows-build. dist.	10 m	15 m	20 m	25 m	10–25 m	10 m	15 m	20 m	25 m	10–25 m
MCC	0.39	0.23	0.09	− 0.26	0.1	0.53	0.19	0.1	− 0.003	0.26
F1 score	0.65	0.48	0.39	0.28	0.45	0.81	0.78	0.70	0.58	0.73
Accuracy	0.49	0.33	0.25	0.16	0.3	0.73	0.65	0.55	0.41	0.59

which is expected since the target area decreases with distance.

A sample result of this approach is in Fig. 13a, we can see the outputs from OpenPOSE and YOLO. The result shown in the picture is a false positive. The user was pointing to the window above the selected one at that moment. This ambiguity caused by the lack of tree dimensional information in the decision process is the most significant cause of errors in the 2D approach.

3D pointing gesture interface approach result

This approach uses ORB-SLAM to infer the real-worlds thirds dimension, from monocular images, to predict the pointed window better. The calibrated file used for each participant was previously recorded.

The results (Table 3) of the 432 combinations using the 3D pointing gesture interface approach showed an F1-Score of 0.73 and a Matthews Correlation Coefficient (MCC) of 0.26, which offers a significant improvement from the previous result of the 2D Approach.

Analyzing the results by row: The first row shows an F1-Score of 0.81 and an MCC of 0.53, the second row an F1-Score of 0.78 and an MCC of 0.19, the third row an F1-Score of 0.70 and an MCC of 0.1, and the fourth row an F1-Score of 0.58 and an MCC of −0.003. So the results worsen the more distant the row from the building, which is expected since the target area decreases with distance.

A sample result of this approach is in Fig. 13b, we can see the outputs from ORB-SLAM, OpenPOSE and YOLO. The result shown in the picture is a true positive.

We classified each detected window as true positive if it was the correct window target (Fig. 14a–c). We classified each detected window as false positive if it wasn't the correct target window (Fig. 14d–f). We classified these as false negatives if no window was selected when the user was pointing to a window in the scene (Fig. 14g–i). We classified these as true negatives when either the correct window target was outside the image or the user was not pointing to a window (Fig. 14j–l).

Discussion

Regarding the effectiveness of this method for firefighter applications, there are a number of factors that should be considered. First, the types of building structures natural to the region of application. The standardization of building structures in Japan facilitates the use of visual solutions like our proposed 3D interface. Second is the angular resolution necessary to visualize the scene. For example, in case of buildings with more than four stories, the distance between user and target window can extrapolate the technical limits of low-cost UAV-mounted cameras. In that case it would be necessary a device with increased angular resolution, which in turn would enable greater image resolution, implicating a higher computational cost.

In the current state of the proposed 3D pointing gesture interface approach, the effective range for practical use regarding the number of floors of a building would be up to the 3rd floors of buildings. The effective range regarding the distance user-building would be 20 m. The third row of the real-world experiment stood at a 20 m distance from the building and provided a F1-score of 0.7 (Table 3). The fourth row showed a decay in the F1-score of 0.12 points, scoring 0.58. The results of the fourth row, although promising, felt cumbersome. Regarding accuracy, the 3D interface presented overall had a total of 59% accurate

Fig. 13 **a** This figure shows the outputs of OpenPOSE, YOLO, and the 2D pointing gesture interface approach. This sample is a false positive; the correct window is above the selected one. This error is due to a line crossing multiple windows is ambiguous. This example is the most significant cause of error given the lack of 3D information to make a more informed decision in this approach. **b** This figure shows the outputs of OpenPOSE, YOLO, ORB-SLAM, and the 3D pointing gesture interface approach. Here, the calculation considers the estimated depth from projective transformations and monocular SLAM's point cloud, resulting in selecting the correct window. This sample is a true positive

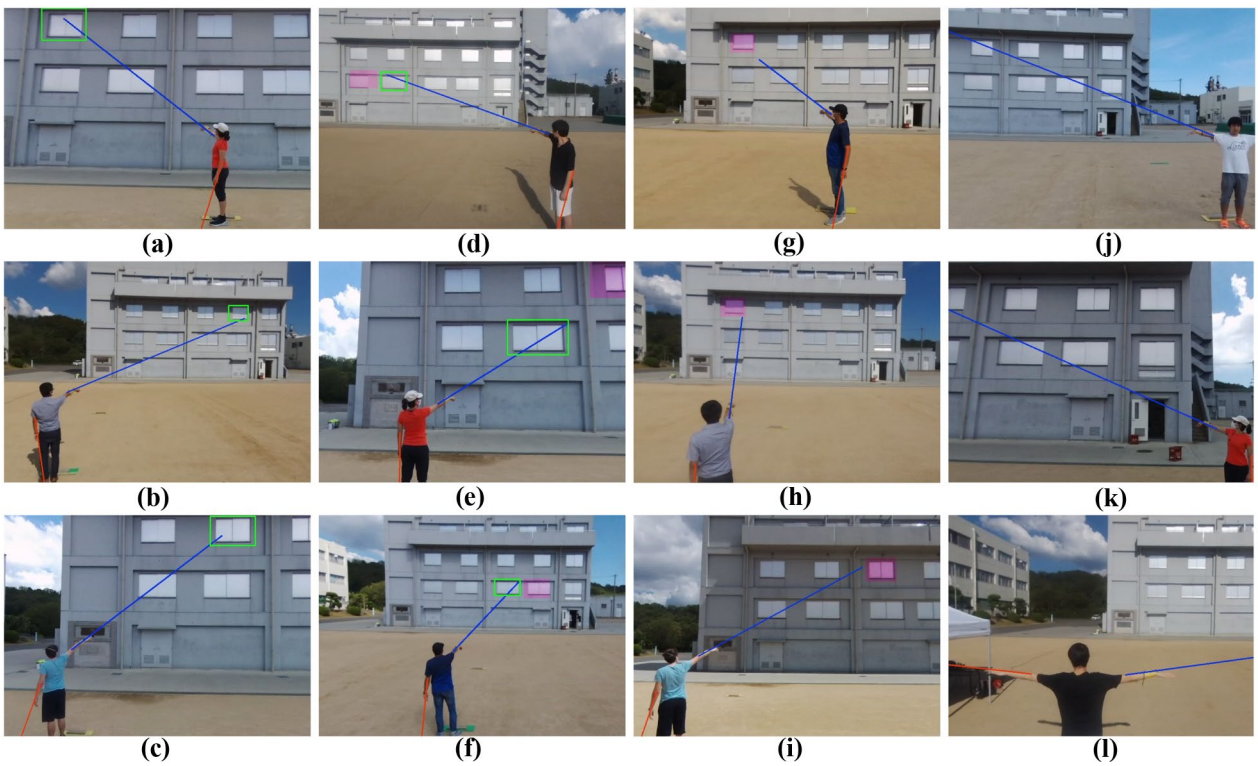

Fig. 14 Samples from the 3D pointing gesture interface approach. **a**–**c** True positive samples. **d**–**f** False positive samples. **g**–**i** False negative samples. **j**–**l** True negative samples

guesses, which is a significant improvement from the 30% accurate guesses from the previous 2D proposed solution (Table 3). The first row, 10 m from the building, was the most accurate, presenting 73% accurate guesses on the 3D gesture interface approach. The expected accuracy for flawless use in a real emergency situation is higher, but when we increase the image size we add computational cost that could hinder the use of low-cost equipment. So the 3D gesture interface approach could benefit from accuracy improvement that doesn't impact the current computational cost.

Other factors like weather conditions and the presence of smoke coming out the windows could also hinder the application of the proposed visual-based 3D interface solution.

Overall, the results from the 3D pointing gesture interface approach showed significant improvement in the limited setting in which it operated. Figure 15 shows the most significant improvement from the 2D approach to the 3D pointing gesture interface approach. With the 3D pointing gesture interface approach it is possible to distinguish between elements aligned on the same plane, or when there aren't any object being pointed inside the scene.

In the real-world experiment, the UAV was facing the building as statically as possible, although a considerable amount of drift was present due to wind conditions at the time. The calibration file for each person was previously recorded in another environment to simulate the real-world setting where one calibration file was used multiple times for the same person.

The UAV distance to the user was primarily determined by acceptable output from openPOSE given the resolution. We understand that by increasing the resolution of the UAV image, it is possible also to increase the distance between the UAV and user, but this will also increase the computational cost of the system. When running the experiment, the simultaneous localization and mapping (SLAM) based system ran at 16 fps on average, and the 2D based system ran at 29 fps.

The use of ORB-SLAM required at least a small amount of translational movement to keep updating the generating point cloud. To obtain an initial point cloud, the UAV can be pre-programmed to translate along the horizontal direction at a certain height and distance which can cover the whole surface of the building. In our experiment, the height is 3.5 m from the ground and the translate distance is around 1 m which can cover a building of around 3 stories, standing at 30 m distance from the UAV.

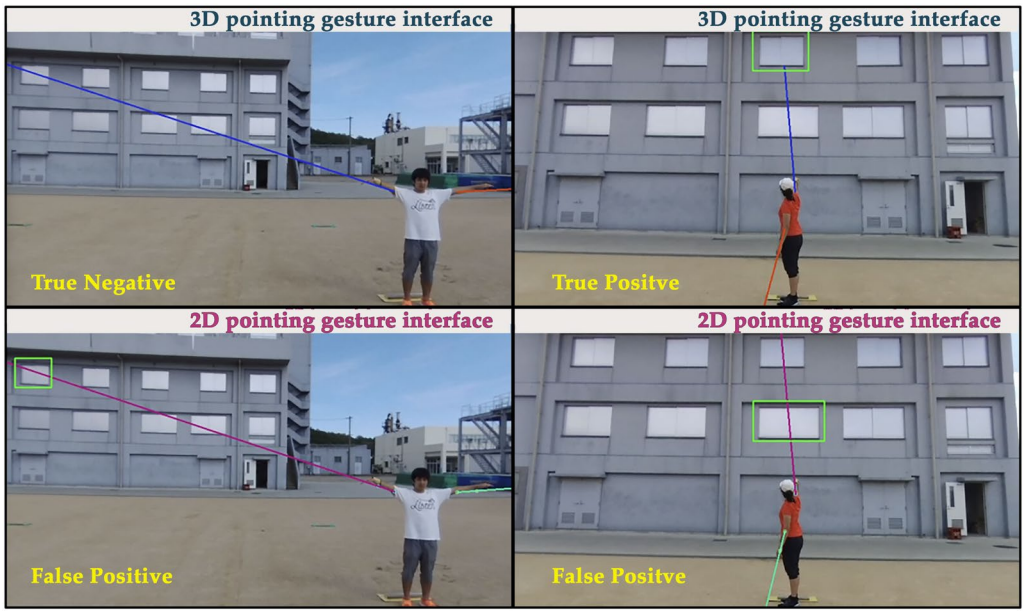

Fig. 15 Samples of the 3D pointing gesture interface approach outperforming the 2D pointing gesture interface approach, given the same scene

We also showed that the sparse point cloud from ORB-SLAM could detect the user position, and the translational movements ensured that the point cloud updated that position after a few moments.

The output of ORB-SLAM was also incremented to provide the point cloud associated with the current image frame, and that is a modification to the original ORB-SLAM code, which we will make available for further research.

Occlusion occurred when the user's arm was not visible because other parts of the user body overlaid the pointing arm (Fig. 16). Often this body part was the head. Occlusion occurred in less than 4% of the 432 combinations tested. It happened mostly in the middle column, that is, positions 2, 5, 8, and 11 of the experiment setting (Fig. 11). This result suggests that pointing from the side columns resulted in better detection from the UAV's point of view (POV).

We filtered the OpenPOSE output so that it would use the skeleton of only the user, even if other persons appeared in the background of the image. We did that by filtering the biggest OpenPOSE skeleton fist, then tracking that skeleton using pixel distance from the previous frame. The result was that after the initial movement, it could keep track of the correct skeleton regardless of its size. For smoothness, we used the averaged value of the last ten keypoints, for each OpenPOSE keypoint on the user's skeleton.

The current 3D pointing gesture interface approach could be used for other scenarios besides firefighting, it has a variety of applications because it's based only on a monocular camera, so the only constraint is the need for a monocular camera, a computational unit and a

Fig. 16 Sample of OpenPOSE's output where occlusion occurred. On both images, the head overlaid the pointing arm. On the top image, the hand's keypoint could not be "guessed." On the bottom image, even with occlusion, the hand keypoint was "guessed" close to the correct position

wireless signal. The "select" gesture is very general, it's for example a good cue for intent, and there are many lines of researches that try to understand human intent [34–36]. For example, this can be used to understand human intent though camera security on airports, where seeing where people are pointing helps raise security concerns, depending on other factors. Another application example could be a companion UAV that can interact with ordinary objects selected by the user. It could also be used in an educational environment, in outdoor classes when the professor points towards the surroundings, the class camera can identify the target and show a specific content for the class.

One limitation of the system presented here is that in the real-world experiment, the UAV was facing the building as statically as possible, although a considerable amount of drift was present due to wind conditions at the time. Another limitation is that the user must always be inside the field of view of the UAV, to give commands to the UAV. So there must be a confirmation before the drone moves to the desired target, possibly losing sight of the user.

Conclusion

Given the wide variety of applications and multiple degrees of freedom, it is challenging to design a natural interaction, namely, Human–Drone interaction, to control UAVs effectively. In past work, we analyzed which gesture fits best for a particular scenario, used our Gesture Development Process [5], validated it in general tasks and with a diverse group of users [6], and conducted an elicitation study with the targeted users [7].

Here we presented a system that enables UAVs equipped with a monocular camera to determine which window of a building was selected by a human user, on a large-scale, indoors or outdoors. We developed two applications: one with monocular SLAM and another without SLAM (no depth information). Experimental evaluation showed that the 3D pointing gesture interface obtained, on average, a 0.85 F1-Score in simulation and a 0.73 F1-Score in a real-world setting. The 3D pointing gesture interface obtained a 0.58 F1-Score when considering only the results obtained at the maximum distance of 25 m between drone and building.

Our contributions include a Human–Drone interaction investigation on firefighting's first response needs, and the large-scale monocular-based 3D pointing gesture interface approach. We also contributed to the modification of ORB-SLAM's output to include the current frame's respective point cloud, made available for further research. Finally, the verification and validation of both approaches on the simulated environment and real-world scenario contributed by producing artifacts such as a dataset of humans pointing to windows outdoor.

Future work

We see various ways to improve the current 3D pointing gesture interface system. By moving the UAV to get a better point of view (POV) on the pointing gesture, we could avoid occlusion and increase accuracy. This POV would be based on the results presented here.

The visual feedback makes a difference in the accuracy of the user's pointing task and UAV's correct results. Therefore, we could improve the User Experience and statistical results by integrating visual feedback to the current 3D pointing gesture interface system, possibly abdicating laser pointer use.

Finally, the deployment of the 3D pointing gesture interface proposed approach on an embedded system should also be considered.

Abbreviations

HDI: Human–Drone interaction; UAV(s): Unmanned aerial vehicle(s); FoV: Field-of-view; DAQ: Data acquisition; ROS: Robot operating system; MCC: Matthews correlation coefficient; SLAM: Simultaneous localization and mapping; ORB: Oriented FAST and Rotated BRIEF; FAST: Features from accelerated segment test; BRIEF: Binary robust independent elementary features.

Acknowledgements

We would like to thank Kobe Fire Academy for their cooperation and collaboration in the course of this work.

Authors' contributions

ACSM devised the system's basic concept, technically constructed the system, and conducted the research and experiments. PR led the research progress, assisted with the implementation, secured funding for the research, and revised and refined the manuscript. JO assisted the research, assisted with the implementation, secured funding for the research, and revised and refined the manuscript. YU, MH, and HT assisted the research and revised the manuscript. All authors read and approved the final manuscript.

Funding

This research was supported in part by the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) and Khalifa University of Science, Technology and Research Grant, by ONR grant #N62909-18-1-2036, and by the Japanese Government's Monbukagakusho (MEXT) Scholarship.

Data availability statement

The datasets generated during the current study are available in the Next-Cloud repository. https://bit.ly/33FldHN

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, Japan. [2] Cybermedia Center, Osaka University, Toyonaka, Osaka, Japan.

Received: 18 October 2020 Accepted: 5 March 2021
Published online: 16 April 2021

References

1. Al-Eidan RM, Al-Khalifa H, Al-Salman AM (2018) A review of wrist-worn wearable: sensors, models, and challenges. J Sens Hindawi. https://doi.org/10.1155/2018/5853917
2. Blum, R., 2008. Linux command line and shell scripting bible (Vol. 481). John Wiley & Sons
3. Bacim F, Nabiyouni M, Bowman DA (2014) Slice-n-Swipe: a free-hand gesture user interface for 3D point cloud annotation. In: IEEE symposium on 3D user interfaces (3DUI), pp 185–186
4. Jeong S, Jin J, Song T, Kwon K, Jeon JW (2012) Single-camera dedicated television control system using gesture drawing. IEEE Trans Consum Electr 58(4):1129–1137
5. Medeiros AC, Tavares TA, da Fonseca IE (2018) How to design an user interface based on gestures? In: International conference of design, user experience, and usability. Springer, Cham, pp 63–74
6. Medeiros ACS, Ratsamee P, Uranishi Y, Mashita T, Takemura H, Tavares TA (2020) 3D gesture interface: Japan-Brazil perceptions. In: International conference on human-computer interaction. Springer, Berlin, pp 266–279
7. Medeiros AC, Ratsamee P, Uranishi Y, Mashita T, Takemura H (2020) Human–Drone interaction: using pointing gesture to define a target object. In: International conference on human-computer interaction. Springer, Berlin, pp 688–705
8. Funk M (2018) Human–Drone interaction: let's get ready for flying user interfaces! Interactions 25(3):78–81
9. Mitra S, Acharya T (2007) Gesture recognition: a survey. IEEE Trans Syst Man Cybernetics Part C Appl Rev 37(3):311–324. https://doi.org/10.1109/TSMCC.2007.893280
10. Van den Bergh M, Van Gool L (2011) Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In: IEEE workshop on applications of computer vision (WACV), pp 66–72: https://doi.org/10.1109/WACV.2011.5711485
11. Nandakumar R, Kellogg B, Gollakota S (2018) Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information. J Sens. https://doi.org/10.1155/2018/5809769
12. Choi JW, Ryu SJ, Kim JH (2019) Short-range radar based real-time hand gesture recognition using LSTM encoder. IEEE Access 7:33610–33618
13. Dankovich LJ, Bergbreiter S (2019) Gesture recognition via flexible capacitive touch electrodes. In: International conference on robotics and automation (ICRA), pp 9028–9034
14. Chossat JB, Tao Y, Duchaine V, Park YL (2015) Wearable soft artificial skin for hand motion detection with embedded microfluidic strain sensing. In: IEEE international conference on robotics and automation (ICRA), pp 2568–2573. https://doi.org/10.1109/ICRA.2015.7139544
15. DelPreto J, Rus D (2019) Sharing the load: human-robot team lifting using muscle activity. In: International conference on robotics and automation (ICRA), pp 7906–7912:
16. DelPreto J, Rus D (2020) Plug-and-play gesture control using muscle and motion sensors. In: ACM/IEEE international conference on human-robot interaction, pp 439–448
17. Kim J, Mastnik S, Andre E (2008) EMG-based hand gesture recognition for realtime biosignal interfacing. In: International conference on intelligent user interfaces, pp 30–39
18. Samadani AA, Kulic D (2014) Hand gesture recognition based on surface electromyography. In: International conference of the IEEE engineering in medicine and biology society, pp 4196–4199
19. Nandakumar R, Kellogg B, Gollakota S (2014) Wi-fi gesture recognition on existing devices. arXiv preprint arXiv:1411.5394
20. Tolgyessy M, Dekan M, Duchon F, Rodina J, Hubinsky P, Chovanec LU (2017) Foundations of visual linear human-robot interaction via pointing gesture navigation. Int J Soc Robot 9(4):509–523
21. Liu T, Chen Z, Wang X (2019) Automatic instructional pointing gesture recognition by machine learning in the intelligent learning environment. In: International conference on distance education and learning, pp 153–157
22. Gromov B, Guzzi J, Gambardella LM, Giusti A (2020) Intuitive 3D control of a quadrotor in user proximity with pointing gestures. Sensors 8(9):10
23. Chen YA, Wu TY, Chang T, Liu JY, Hsieh YC, Hsu LY, Hsu MW, Taele P, Yu NH, Chen MY (2018) ARPilot: designing and investigating AR shooting interfaces on mobile devices for drone videography. In: International conference on human-computer interaction with mobile devices and services (MobileHCI'18), pp 1–8
24. Chen L, Ebi A, Takashima K, Fujita K, Kitamura Y (2019) PinpointFly: an egocentric position-pointing drone interface using mobile AR. In: SIGGRAPH Asia 2019 emerging technologies (SA'19), pp 34–35
25. Obaid M, Kistler F, Kasparaviciute G, Yantac AE, Fjeld M (2016) How would you gesture navigate a drone? a user-centered approach to control a drone. In: International academic Mindtrek conference, pp 113–121
26. Lidar sensors for robotic applications. https://www.sentekeurope.com/robotics-lidar
27. Cao Z, Hidalgo G, Simon T, Wei SE, Sheikh Y (2019) Openpose: realtime multi-person 2d pose estimation using part affinity fields. In: IEEE transactions on pattern analysis and machine intelligence
28. Redmon J, Farhadi A (2018) Yolov3: An incremental improvement. arXiv preprint. arXiv:1804.02767
29. Mur-Artal R, Tardos JD (2017) Orb-slam2: an open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans Robot 33(5):1255–1262
30. Mur-Artal R, Tardos JD (2014) ORB-SLAM: tracking and mapping recognizable features. In: Workshop on multi view Geometry in robotics (MVIGRO), pp 2
31. Koenig N, Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE/RSJ international conference on intelligent robots and systems (IROS) 3:2149–2154
32. Parrot Bebop 2 drone. https://www.parrot.com/us/drones/parrot-bebop-2
33. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY (2009) ROS: an open-source Robot Operating System. In: ICRA workshop on open source software 3(3.2), pp 5
34. Jin Z, Pagilla PR (2020) Human-robot teaming with human intent prediction and shared control. Artif Intell Mach Learn Multi Domain Oper Appl. https://doi.org/10.1117/12.2559296
35. Holtzen S, Zhao Y, Gao T, Tenenbaum JB, Zhu SC (2016) Inferring human intent from video by sampling hierarchical plans. In: IEEE international conference on intelligent robots and systems (IROS), pp 1489–1496
36. Erden MS, Tomiyama T (2010) Human-intent detection and physically interactive control of a robot without force sensors. IEEE Trans Robot 26(2):370–382

Publisher's Note