RESEARCH

3D Pointing Gestures as Target Selection Tools for Monocular UAVs

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

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

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 the firefighters' desire 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 improves a 2D pointing gesture interface by integrating depth information with SLAM, solving multiple objects aligned on the same plane ambiguity, in a large-scale outdoor environment. Experimental results showed that our 3D pointing gesture interface obtained a 0.85 and 0.73 F1-Score on average in simulation and real-world experiments and 0.58 F1-Score at the maximum distance of 25 meters between 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 arrive 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 firefighters of 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, that 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 could go inside and perform some 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 SLAM system). Experimental evaluation is made available in simulation and also in a real-world setting. The list of contributions include:

- Gesture Development Process and Human-Drone Interaction elicitation study on firefighters' experience in a first-response situation to discover a suitable gesture.
- Development of a monocular-based 2D and 3D pointing gesture interface proposed applications. 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 outdoor.

1 Research Background

1.1 Gesture Development Process

We first went through a Gesture Development Process that led to the choice of the pointing gesture for selecting the windows of a building. We also validated this pointing gesture through an elicitation study with participants who had experience in the system's application environment, mainly Firefighters. This section will explain the background that led to using a "Pointing Gesture" in our scenario.

The process (Figure 2) we presented in [5] defines gestures for a particular function or action of a system. This Gesture Development Process has three stages, and its

final product is a Template depicting the final and best gestures for the given system functions.

In [6], we applied this process to 5 system functions: "Select", "Rotate", "Scale", "Translate" and "Stop" interactions with a 3D Cube (virtual object). The Vocabulary of Finalist Gestures can be seen on Figure 3. We implemented it using a LeapMotion controller (hand and finger tracker) and validated this against a counterpart application that used a mouse and keyboard. Participants from Brazil and Japan tested both applications, and answered a 7-point Likert scale questionnaire with 28 items about their User Experience. The mouse and keyboard application was considered more Conventional and Technical. The Gestural Interface application was deemed more Inventive and Human.

In the present work, we use the gesture for "Select" 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 (Figure 4) showed that all firefighters used a pointing gesture and voice commands to indicate a specific window within an average of 3 seconds.

1.2 Related Work

Gesture recognition[9] has been researched using various devices such as RGB cameras, depth cameras [10, 11], radar-based sensors [12], capacitive sensors [13], sensorized gloves [14], electromyography sensors [15, 16, 17, 18], Wi-Fi [19] and others. Each device has its own limitations, and in this section we will summarise the use of these devices on detecting pointing gestures and/or robot control.

The work of Tölgyessy[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 meters 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 meters), 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 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.

There are multiple studies that research the use of gestures to navigate a UAV [23, 20, 16, 22]. 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 on an outdoor environment, where long distances make the use of depth cameras difficult due to hardware limitations. Moreover, lidars like Hokuyo [24] 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, as this could also impair dexterity or task performance in a firefighting environment. 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.

2 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 Figure 5(a).

2.1 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[25]. 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 (Figure 5(b)) model for speed and accuracy. To detect objects, we used the YOLOV3-tiny [26], 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 (Figure 5(c), 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

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

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[26] 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 Figure 14(a)), 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.

2.2 3D Pointing Gesture Interface Approach

To estimate the correct pointing area, or pointed 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 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 [27] 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 Figure 6.

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. We integrate an estimation of the Z axis pointing towards the inside of the image.

The BODY_25 model (Figure 5(b)) provides 25 keypoints $P_n \in \mathbb{R}^{w \times h}$ where $n \in \mathbb{N}$, $0 \leq n \leq 24$. Each point has 2D coordinates P_{nx} and P_{ny}. We first perform a calibration step; this is simply to create a file to store the values of all the user's keypoints at a fixed distance D_f to the UAV. We also store the real measurements of the user's height and torso size. The three planes of body motion are the sagittal, frontal, and transverse. In the calibration step, the user is standing with arms relaxed at the side of the body, frontal plane. The calibration step takes about a minute and is a one-time step for each user.

2.2.1 Calibration

The calibration step stores the size of the user's torso, in pixels, given by the euclidean distance between points P_1 and P_8:

$$d_f\left(P_1, P_8\right) = \sqrt{\left(P_1^x - P_8^x\right)^2 + \left(P_1^y - P_8^y\right)^2} \tag{2}$$

at a fixed distance D_f to the UAV. When the user moves away or towards the UAV, those distances changes. To estimate the current distance D_c from user to the UAV we use the current distance d_{c18} between points P_1 and P_8, and the inverse linear relationship between size and distance:

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

where d_{c18} is analogous to equation 2. This approximation holds for distances where the angular distance to user is smaller than 10 degrees, which is the case in our scenario.

When moving, if the user's arm stays in the frontal plane, the distances in between his arm's keypoints is decreased by $\frac{D_c}{D_f}$. 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. We assume that the depth P_1 will be the current distance D_c to the UAV. To express that

depth in pixel units, we use the proportion from the torso pixel size in calibration time d_{f18} and real size tx (metric units):

$$P_{1z} = \frac{D_c \times d_{f18}}{tx} \tag{4}$$

Using a right-angle triangle we calculate the estimated depth on an arm keypoint. This intuition is better visualized on Figure 5(c).

2.2.2 Depth Estimation

The estimated depth P_4^z for the hand considers the preceding keypoints' depths P_3^z, P_2^z and P_1^z. 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:

$$P_2^z = P_1^z + \sqrt{\left(\frac{d_{c18} \times d_{f12}}{d_{f12}}\right)^2 - (d_{c12})^2} \tag{5}$$

$$P_3^z = P_2^z + \sqrt{\left(\frac{d_{c18} \times d_{f23}}{d_{f23}}\right)^2 - (d_{c23})^2} \tag{6}$$

$$P_4^z = P_3^z + \sqrt{\left(\frac{d_{c18} \times d_{f34}}{d_{f34}}\right)^2 - (d_{c34})^2} \tag{7}$$

knowing P_2^z and P_3^z, we can estimate P_4^z. 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.

2.2.3 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 [28]. 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 user distance to the UAV and the building (windows). We use this data and the estimated user distance D_c to estimate the building distance B_c.

With the building distance B_c known, the product $b_c = \frac{B_c \times d_{f18}}{tx}$ 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:

$$\alpha \left(\frac{b_c - P_3^z}{P_4^z - P_3^z} \right) = \frac{\boldsymbol{x_c} - P_3^x}{P_4^x - P_3^x} = \frac{\boldsymbol{y_c} - P_3^y}{P_4^y - P_3^y} \tag{8}$$

where α is a scaling factor associated with B_c in the following way:

$$FOV_x = 2 \times arctan \left(\frac{w}{2 \times F_x} \right) \tag{9}$$

$$a = 2 \times D_c \times tan \left(\frac{FOV_x}{2} \right) \tag{10}$$

$$b = 2 \times tan \left(\frac{FOV_x}{2} \right) \tag{11}$$

Finally, α can be obtained by

$$\begin{aligned} \alpha &= \frac{a}{(B_c - D_c) \times b + a} \\ \alpha &= \frac{D_c}{B_c} \end{aligned} \tag{12}$$

in equation 9, the Field-of-View of the Image is calculated with respect to the Focal Length F_x of the camera's Intrinsic Matrix. In equation 10 a is the maximum width, in metric units, in the Field-of-View at distance D_f of the camera. In equation 11, b is the change in the maximum width at every change in the distance to the camera. Lastly, α can be defined as the proportion between the distance to the camera (D_c) and the building distance (B_c), as seen in Figure 7. The result is a target point $T_P \in \mathbb{R}^{w \times h}$ where

$$T_P = (x_c, y_c, b_c) \tag{13}$$

If T_P 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.

3 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.

3.1 Simulation Setup

We used Gazebo[29] 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 (Figure 8). The experiment had eight different positions for the human (Figure 9). In some Experiments, the human model executed three fixed pointing angles (Figure 10), for each one of the eight positions. In other experiments, the human model would point to four windows for each position. The UAV stood at a fixed distance from the building, and a fixed height of 2.5 meters 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 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 four second-window to identify the Mode and make a decision.

3.1.1 2D Pointing Gesture Interface Approach Result

We tested the 2D approach using the eight positions shown in Figure 9, the building stood at 10 meters from the UAV. This Experiment is labeled "Exp0". For each position, the human model tested all three fixed angles, as shown in Figure 10. 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.

3.1.2 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 Figure 9, with fixed pointing angles (Figure 10). The building stood at 10 meters 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 meters away from the UAV. We wanted to see how it behaved with targets (windows) further away. Results showed approximately 58% true positives, 25% true negatives, 16% false positives, and 0% of false negatives. The resulting accuracy was 83%, precision was 77%, recall was 100%, F1 Score was 0.87, and the Matthew Correlation Coefficient was 0.68. The results were slightly better 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 Figure 9, 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, 12% false positives, and 0% 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 (Figure 9), 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 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 Figure 11.

3.2 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 [30] for the experiments; it is a quadcopter that weighs around 500 grams, offers 25 minutes of autonomous flight time, can film in 1080p full HD with its wide-angle 14-megapixel lens. But for the sake of performance, the image size was limited to 856 x 480 pixels. We used an available SDK by the name of 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, 16GB 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)[31].

We have asked permission from firefighters and received permission from 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.

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 Figure 12). 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. The closest row was 10 meters from the building, increasing a distance of 5 meters for every other row until the last row (25 meters from the building). For each spot, the user first opened both arms and pointed sideways, at no window, then the user pointed at ten windows, sequentially, 5 seconds for each window. A total of 432 combinations of windows, users and positions were tested.

To ensure the correct aim, we attached a laser pointer (Figure 13) to the user's right arm. The laser was of 3000mW and 532 wavelengths so the user could point with precision, outdoors, in a long-range. 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 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 meters behind the user and 1.7 meters to 3 meters high. We used this short distance because at longer distances, the 856 x 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.

3.2.1 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, which is expected since the target area decreases with distance.

A sample result of this approach is in Figure 14(a), 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.

3.2.2 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 Figure 14(b), 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 pointed window as true positive if it was the correct pointed window (Figure 15(a) - 15(c)). We classified a detected pointed window as false positive if it wasn't the correct pointed window (Figure 15(d) - 15(f)). We classified as false negative if no window was selected when the user was pointing to a window on the scene (Figure 15(g) - 15(i)). We classified as true negative when either the correctly pointed window was outside the image, or the user was not pointing to a window (Figure 15(j) - 15(l)).

4 Discussion

The results from the 3D pointing gesture interface approach showed significant improvement in the limited setting in which it operated. Figure 16 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 used 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 it will also increase the computational cost of the system. When running the experiment, the SLAM based system ran at 16fps on average, and the 2D based system ran at 29fps.

The use of ORB-SLAM required at least a small amount of translational movement to keep updating the generating point cloud. Sometimes, the residual drift from the UAV could accomplish that; other times, we had to translate the UAV by manual commands. We noticed this because we were also running the ORB-SLAM when we were recording the UAV's camera topic.

We also show 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 (Figure 17). 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 (Figure 12). This result suggests that pointing from the side columns resulted in better detection from the UAV's 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, such as a companion UAV that can interact with ordinary objects selected by the user.

5 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 works, 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 meters between drone and building.

Our contributions include a Human-Drone Interaction investigation on firefighting's first response needs, and the large-scale monocular-based 2D and 3D pointing gesture interface approaches. 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.

6 Future Work

We see various ways to improve the current 3D pointing gesture interface system. By moving the UAV to get a better 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

Availability of data and materials

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

Competing interests

The authors declare that they have no competing interests.

Funding

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

Author 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.

Acknowledgements

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

Author details

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

References

1. Al-Eidan, R. M., Al-Khalifa, H., and Al-Salman, A. M. : A Review of Wrist-Worn Wearable: Sensors, Models, and Challenges. Journal of Sensors, Hindawi (2018)
2. The Linux Command Line for Beginner. https://ubuntu.com/tutorials/command-line-for-beginners1-overview

3. Bacim, F., Nabiyouni, M. and Bowman, D.A.: Slice-n-Swipe: A free-hand gesture user interface for 3D point cloud annotation. IEEE Symposium on 3D User Interfaces (3DUI), 185–186 (2014)
4. Jeong, S., Jin, J., Song, T., Kwon, K. and Jeon, J.W.: Single-camera dedicated television control system using gesture drawing. IEEE Transactions on Consumer Electronics **58**(4), 1129–1137 (2012)
5. Medeiros, A. C., Tavares, T. A., and da Fonseca, I. E.: How to Design an User Interface Based on Gestures? International Conference of Design, User Experience, and Usability (Springer, Cham.), 63–74 (2018)
6. Medeiros, A. C. S., Ratsamee, P., Uranishi, Y., Mashita, T., Takemura, H., and Tavares, T. A.: 3D Gesture Interface: Japan-Brazil Perceptions. International Conference on Human-Computer Interaction (Springer), 266–279 (2020)
7. Medeiros, A. C., Ratsamee, P., Uranishi, Y., Mashita, T., and Takemura, H.: Human-Drone Interaction: Using Pointing Gesture to Define a Target Object. International Conference on Human-Computer Interaction (Springer), 688–705 (2020)
8. Funk, M.: Human-drone interaction: let's get ready for flying user interfaces! Interactions **25**(3), 78–81 (2018)
9. Mitra, S., and Acharya, T.: Gesture Recognition: A Survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **37**(3), 311–324 (2007). https://doi.org/10.1109/TSMCC.2007.893280
10. Van den Bergh, M., and Van Gool, L. : Combining RGB and ToF Cameras for Real-Time 3D Hand Gesture Interaction. IEEE Workshop on Applications of Computer Vision (WACV), 66–72 (2011). https://doi.org/10.1109/ WACV.2011.5711485
11. Nandakumar, R., Kellogg, B., and Gollakota, S.: Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information. Journal of Sensors (2018)
12. Choi, J. W., Ryu, S. J., and Kim, J. H.: Short-range radar based real-time hand gesture recognition using LSTM encoder. IEEE Access **7**, 33610–33618 (2019)
13. Dankovich, L. J., and Bergbreiter, S.: Gesture Recognition via Flexible Capacitive Touch Electrodes. International Conference on Robotics and Automation (ICRA), 9028–9034 (2019)
14. Chossat, J. B., Tao, Y., Duchaine, V., and Park, Y. L.: Wearable soft artificial skin for hand motion detection with embedded microfluidic strain sensing. IEEE international conference on robotics and automation (ICRA), 2568–2573 (2015). https://doi.org/10.1109/ICRA.2015.7139544
15. DelPreto, J., and Rus, D.: Sharing the load: human-robot team lifting using muscle activity. International Conference on Robotics and Automation (ICRA), 7906–7912 (2019)
16. DelPreto, J., and Rus, D. : Plug-and-Play Gesture Control Using Muscle and Motion Sensors. ACM/IEEE International Conference on Human-Robot Interaction, 439–448 (2020)
17. Kim, J., Mastnik, S., and André, E. : EMG-based hand gesture recognition for realtime biosignal interfacing. International Conference on Intelligent User Interfaces, 30–39 (2008)
18. Samadani, A. A., and Kulic, D.: Hand Gesture Recognition Based on Surface Electromyography. International Conference of the IEEE Engineering in Medicine and Biology Society, 4196–4199 (2014)
19. Nandakumar, R., Kellogg, B., and Gollakota, S.: Wi-fi gesture recognition on existing devices. arXiv preprint arXiv:1411.5394. (2014)
20. Tölgyessy, M., Dekan, M., Duchoň, F., Rodina, J., Hubinský, P., and Chovanec, L. U.: Foundations of visual linear human–robot interaction via pointing gesture navigation. International Journal of Social Robotics **9**(4), 509–523 (2017)
21. Liu, T., Chen, Z., and Wang, X.: Automatic Instructional Pointing Gesture Recognition by Machine Learning in the Intelligent Learning Environment. International Conference on Distance Education and Learning, 153–157 (2019)
22. Gromov, B., Guzzi, J., Gambardella, L.M. and Giusti, A.: Intuitive 3D Control of a Quadrotor in User Proximity with Pointing Gestures. sensors **8**(9), 10 (2020)
23. Obaid, M., Kistler, F., Kasparavičiūtė, G., Yantaç, A.E. and Fjeld, M. : How would you gesture navigate a drone? a user-centered approach to control a drone. International Academic Mindtrek Conference, 113–121 (2016)
24. Lidar Sensors for Robotic Applications. https://www.sentekeurope.com/robotics-lidar
25. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E. and Sheikh, Y.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
26. Redmon J, Farhadi A.: Yolov3: An incremental improvement. arXiv preprint (2018). arXiv:1804.02767
27. Mur-Artal, R. and Tardós, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics **33**(5), 1255–1262 (2017)
28. Mur-Artal, R. and Tardós, J.D. : ORB-SLAM: tracking and mapping recognizable features. Workshop on Multi View Geometry in Robotics (MVIGRO) **2014**, 2 (2014)
29. Koenig, N. and Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) **3**, 2149–2154 (2004)
30. Parrot Bebop 2 Drone. https://www.parrot.com/us/drones/parrot-bebop-2
31. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y.: ROS: an open-source Robot Operating System. ICRA workshop on open source software **3**(3.2), 5 (2009)

Figures

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

Figure 2 Gesture Development Process Pipeline

Figure 3 Vocabulary of Gestures The "Select" gesture was applied to a specific scenario: collaboration between UAV and Firefighters.

Figure 4 An elicitation study with firefighters showed that pointing gesture and voice commands were unanimously used to specify windows to a UAV.

Figure 5 (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 projected arm on the picture will be smaller than pointing sideways (frontal plane). Using a right-angle triangle intuition, the calibration data gives us an estimation of the depth value in an arm keypoint.

Figure 6 3D pointing gesture interface application flowchart. ORB-SLAM integrated with Object Detection and Pose Estimation towards pointing gesture interaction.

Figure 7 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.

Figure 8 Simulation Environment We used Gazebo 7.0, all models used were modified from Gazebo's originals.

Figure 9 User's positioning on the experiments. We tested the human model on eight different positions, for each Experiment.

Figure 10 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.

Figure 11 Sample of Simulation Experiments' Results

Figure 12 Top view of real-world experiment

Figure 13 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.

Figure 14 (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.

Figure 15 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.

Figure 16 Samples of the 3D pointing gesture interface approach outperforming the 2D pointing gesture interface approach, given the same scene.

Figure 17 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.

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]	✓	-	✓	-	✓	-	-	✓	-
Present	✓	✓	✓	✓	-	-	✓	-	✓

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	10m	10m	20m	10m	N/A	N/A	N/A	N/A
UAV-User Distance	N/A	N/A	N/A	N/A	3m	5m	7m	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%

Table 3 Summary of Experiments on Real-World Environment

	2D Row1	2D Row2	2D Row3	2D Row4	2D AllRows	3D Row1	3D Row2	3D Row3	3D Row4	3D AllRows
Rows-Building Distance	10m	15m	20m	25m	10-25m	10m	15m	20m	25m	10-25m
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**