The Visual Computer
https://doi.org/10.1007/s00371-023-03250-w

RESEARCH

The spheroidal trackball: generalising the fixed trackball for virtual camera navigation

Daniel González-Toledo[1] · María Cuevas-Rodríguez[1] · Luis Molina-Tanco[1] · Arcadio Reyes-Lecuona[1]

Accepted: 24 December 2023

Abstract
Virtual trackball techniques have become a standard in 3D applications, particularly for interfaces with limited degrees of freedom such as touchscreens or mice. The fact that we are used to them does not mean that they cannot be improved upon. Recent research has highlighted the significance of considering users' mental models of a preferred rotation axis, as it can improve performance, perceived usability and perceived workload. Building upon these findings, this paper introduces the spheroidal trackball framework—a novel method for orbiting the virtual camera around elongated objects. The paper presents the mathematical formulation and the evaluation of the technique. The formulation offers enough information to implement the approach. The evaluation shows the advantages of this approach over the fixed spherical trackball for this class of objects, in terms of task performance, usability and perceived workload. This research constitutes an advancement in the refinement of 3D user interaction techniques, opening new avenues of innovation in this still evolving field.

Keywords Usability · 3D interaction · 3D user interfaces · Virtual trackball · Virtual camera navigation · Ellipsoid · Spheroid

1 Introduction

The inspection of a virtual object is common in many 3D applications, which require a strategy to orbit the virtual camera around the object. This is, in general, not a trivial problem [18]. One of the main challenges is how to map the available set of degrees of freedom (DoF) of the interaction devices onto the DoF of the virtual camera. In desktop or web-based applications, we are constrained to use reduced-DoF interaction devices such as touchscreens or mice. Thus, certain assumptions have to be made about what users *mean* when they perform gestures or actions via these, still ubiquitous, interaction devices.

A family of algorithms assume that the priority is close examination of the object's surface, such as, for example, when editing small details of an object. The challenge here is defining smooth surfaces adapted to the object's geometry. A relatively early example is HoverCam [16], which implements a layered set of surfaces around an object via a sphere tree. More recent work such as the generalised trackball [17] builds on HoverCam with a multi-resolution mesh-less representation of the smooth surface around the object. These strategies have not been, in general, adopted by commercial applications involving 3D inspection of virtual objects. The risk is the excessive dependence on details of the object surface, which might result, for example, in shakiness of the virtual camera motion.

Most applications choose the opposite strategy, which is to constraint the camera to a virtual sphere centred at the object. The camera is always perpendicular to the surface of the sphere, and the radius of the sphere depends on a *zoom* level. Over the years, researchers have reported a number of variations to the ubiquitous *Virtual Sphere* controller, a mechanism proposed by Chen et al. in an early study [8]. The basics of the technique was to rotate the 3D scene by encasing it in a virtual sphere. To orbit the camera, the user had to roll the virtual sphere by dragging its surface in a desired direction: up–down and left–right. This created and accumulated

✉ Arcadio Reyes-Lecuona
areyes@uma.es

Daniel González-Toledo
dgonzalezt@uma.es

María Cuevas-Rodríguez
mariacuevas@uma.es

Luis Molina-Tanco
lmtanco@uma.es

[1] Telecommunication Research Institute (TELMA), Universidad de Málaga , Málaga, Spain

Published online: 27 February 2024

rotations around axes perpendicular to the dragging gesture, axes that were arbitrary with respect to the object, but always parallel to the screen.

The virtual sphere and its subsequent variations, such as Shoemake's *Arcball* [24, 25], are ideal to give the user the impression that the object is suspended in an empty space. In fact, they suggest that the user actions rotate the object, rather than suggesting that the user is orbiting the camera. However, if the scene includes a ground reference plane, the approach to map user gestures to camera motion is different. In fact, in the *family tree* of virtual trackballs, these approaches are in a different branch which stems from an earlier contribution, the *Number Wheel* [27], a two-axis valuator in which the user gestures in two dimensions were mapped into rotations around two specific rotation axes. A variation of this technique, the two-axis valuator (TAV) with fixed up-vector or *Fixed Trackball* [2, 20] is widely used in 2022 by commercial 3D modelling tools and repository viewers [1, 6, 14, 19, 26]. Here, horizontal gestures are mapped onto rotations around a fixed global vertical axis. This is consistent with scenes in which there is a ground plane which serves as reference.

Recent work [13] has shown that there can be advantages to generalising the fixed trackball to fixing other axes of rotation different to a global up-vector, such as, for example, a horizontal global rotation axis, when this better fits a *mental model* that the user has about the object. For example, a wheel or a turbine may have a horizontal rotation axis. If the object is placed with this axis parallel to the reference ground plane, Gonzalez-Toledo et al. suggested that fixing a horizontal global axis may be the best possible approach in terms of usability, performance and fatigue reduction, when navigating around these objects. Other axes might be preferred in other contexts, such as, for example, a diagonal rotation axis representing the tilting of the Earth's rotation axis. This improvement of the ubiquitous TAV with fixed up-vector requires assuming a mental model that users share about an object, or, alternatively, interactive tools that allow selecting which axis to fix in navigation.

The different members of the *trackball family* work on the implicit assumption that the objects inspected have high sphericity, i.e. their bounding box is similar to a cube. However, when the object is less spherical in proportions, all trackball approaches share a drawback: multiple *zoom* and *pan*[1] operations are required to inspect different areas of the object. Since the camera orbits around the centre of the object, if the camera is too close, the sides of the object will be inaccessible—thus the need for the pan operation. If the camera is further, so as to access the sides, then it might be too far from other areas of the object surface (see Fig. 1). The accumulation of orbit, zoom and pan operations can be inefficient and unsatisfactory for the user. On the other hand, trying to adapt the camera motion to the object's exact shape introduces the aforementioned issues related to excessive detail in camera motion [17].

What we ask ourselves is if we can take the best of all worlds: keep the simplicity and universality of the trackball, acknowledge the mental model of the users about the rotation axes of the object they are inspecting, but increase the interaction efficiency for objects with low sphericity, by automatically deforming its surface to adapt the camera orbit to the object's bounding box. In the remaining of this paper, we introduce the formulation of the spheroidal trackball, and evaluate if it improves efficiency, perceived usability and perceived workload in the inspection of objects where sphericity is violated.

2 Formulation of the spheroidal trackball

The fixed trackball with generalised fixed vector [13] defines an enclosing spherical surface which constraints the camera orbit. Our goal is to generalise this strategy for objects that depart from the sphericity constraint. In this paper, we consider objects that extend or reduce their bounding box in only one of the three axis, i.e. two of the dimensions of the bounding box have the same length, which results in a trackball in the shape of an ellipsoid of revolution, or a *Spheroidal Trackball*. The axes which is different in size coincides with the preferred axis of rotation in the mental model that the user has about the object, the object *intrinsic* axis of rotation [13].

Let us assume, without loss of generality, that this dimension is aligned with a global X-axis[2], and that the bounding box is centred at $(0, 0, 0)$. The bounding box extends e_x, e_y and e_z units in each axis, so that the dimensions of the bounding box are $2e_x \times 2e_y \times 2e_z$, and $e_y = e_z$.

An ellipsoid is, in general, an affine transformation of a sphere defined as a quadric surface. If we define a Cartesian coordinate system with origin at the centre of the ellipsoid, then for any point (x, y, z) on the ellipsoid Eq. 1 holds:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{1}$$

where a, b and c are half the length of the principal axes of the ellipsoid. An ellipsoid of revolution or simply a spheroid is an ellipsoid where two of the principal axes have the same

[1] A *pan* operation is a translation in global coordinates of the trackball centre.

[2] In the rest of cases, it is possible to compute a transformation of the coordinate system so that the intrinsic axis is aligned with the X-axis, before proceeding with the formulation.

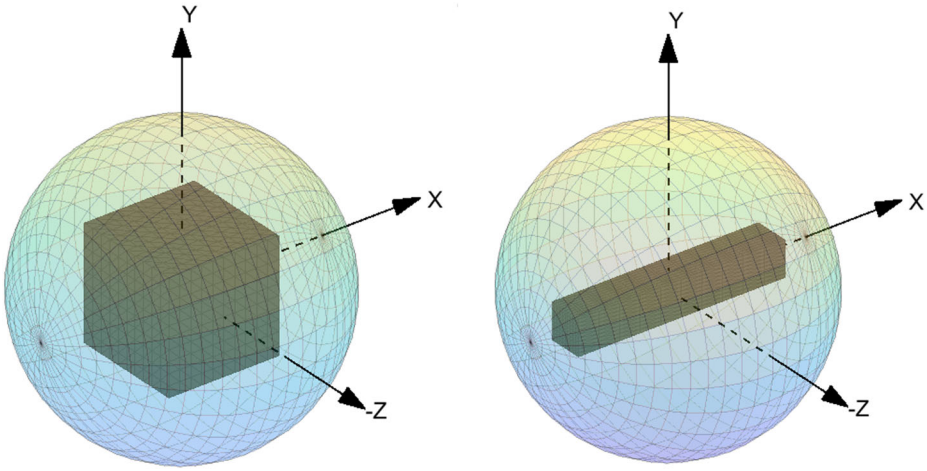


Fig. 1 Trackballs work on the assumption that the objects inspected have high sphericity, i.e. their bounding box is similar to a cube (left). But objects with little sphericity (right) may require multiple zoom and/or pan operations for the camera, if the object is to be examined closely

length, which simplifies Eq. 1 to Eq. 2:

$$\frac{x^2}{a^2} + \frac{y^2 + z^2}{b^2} = 1 \tag{2}$$

This will be the equation that restricts how the camera orbits around the object. In particular, in our proposal:

1. the camera position will be defined by a longitude and a latitude on the spheroid.
2. the camera orientation will be always perpendicular to the surface of the spheroid, as in the standard spherical trackball.

Thus, the problem can be formulated as follows: given the bounding box of the object (as previously defined), and a desired position and orientation in global coordinates of a virtual camera, we must solve for:

- the parameters a, b in Eq. 2, which define the spheroid that bounds the trajectory of the camera,
- a strategy to map two-dimensional user gestures to variations in latitude and longitude of the camera position across the spheroid.
- a strategy to map *zoom level* user gestures to variations in the size of the spheroid, i.e. to changes in a, b.

Let us suppose an initial position and orientation of the camera that, without loss of generality, will simplify the mathematical formulation of the spheroidal trackball. The camera is at $(0, 0, -Z_0)$, and the bounding box of the object is centred at $(0, 0, 0)$ (Fig. 2, left). The orientation of the camera is such that it is pointing towards $(0, 0, 0)$, and its up-vector is towards Y, and remember that there is a preferred axis of rotation towards X.

Any two-dimensional gesture of the user should result in a smooth motion of the virtual camera, such that the camera keeps looking at the object at all times. Let us further simplify the problem and imagine that the user is going to perform only a horizontal gesture to inspect the object towards X, i.e. the dimension along which the object departs from sphericity. The camera should describe a trajectory on the XZ plane, from $(0, 0, -Z_0)$ to $(a, 0, 0)$. The initial and final orientation of the camera are both clear, towards $(0, 0, 0)$. We have established that the camera shall always be perpendicular to the spheroidal surface, but how to choose a? (see Fig. 2 right).

We start by remembering that the direction in which the camera is looking is always perpendicular to the surface of the spheroid. Since we simplified our problem, this means that, for a given position of the camera on its elliptical trajectory, the camera is always looking in a direction normal to the ellipse on that point. A fundamental difference with the spherical trackball is that these normals do not meet at $(0, 0, 0)$. Instead, the camera travels almost in an automatic pan of the object, and the normals cross the major axis of the ellipse at points $(X_c, 0, 0)$ which advance along the X-axis more and more slowly until they reach a limit, a turning point that eventually makes the camera face the bounding box from $(a, 0, 0)$. It so happens that the turning point is a perfectly known point, the cusp of the ellipse's evolute [29]. The evolute of an ellipse is a stretched astroid, sometimes known as the Lamé curve [28] (Fig. 3). The cusp of the evolute $(E_c, 0, 0)$ can be obtained from the dimensions of the semi-axes of the ellipse, as:

$$E_c = a - \frac{b^2}{a} \tag{3}$$

This expression allows us to choose a value for a (Fig. 3). For the spherical trackball, $E_c = 0$. If we start to make a larger, we will start to get the panning effect. The value of a has to grow sufficiently so as to look at the bounding box of the object at all times, i.e. the turning point E_c should not exceed the dimensions of the bounding box e_x. Moreover, if

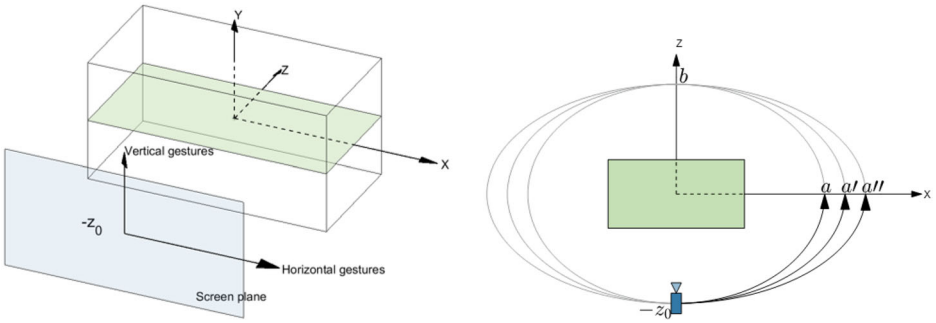


Fig. 2 The virtual camera is at $(0, 0, Z_0)$, and the bounding box of the object is centred at $(0, 0, 0)$. The user performs a horizontal gesture, taking the camera from $(0, 0, -Z_0)$ to $(a, 0, 0)$. What does the user *mean*? What should be the orientation of the camera on every point of the trajectory? And, how to choose between a, a' or a''?

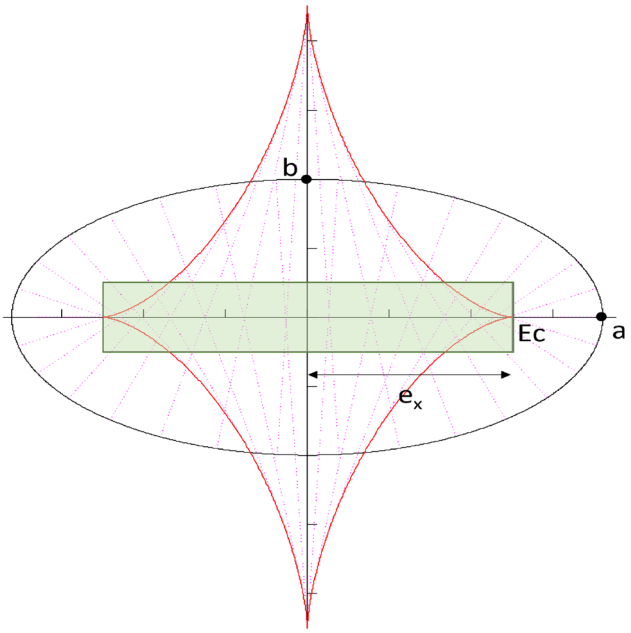


Fig. 3 The ellipse represents the trajectory of the camera. The normals to the ellipse are the directions in which the camera points. The normals cross the X-axis at points that do not go beyond E_c, the cusp of the evolute of the ellipse. We choose this point to be on the bounding box of the object to select the size of the ellipse-shaped curve that constraints the horizontal displacement of the camera

we make $E_c = e_x$ (see Fig. 3), the camera will focus at the edge of the object while turning around it without letting the object go out of frame at any moment. Notice that choosing a larger value for E_c may make the object go out of frame in that situation. Thus, we finally solve for a by making $E_c = e_x$ in Eq. 3. The other parameter b in Eq. 2 is $b = Z_0$ since we only allowed the bounding box to stretch in one dimension.

$$\begin{aligned} b &= Z_0 \\ a &= \tfrac{1}{2}(e_x + \sqrt{e_x^2 + 4b^2}) \end{aligned} \tag{4}$$

We have solved for a and b in 2, which define the spheroid that bounds the orbiting of the camera. Thus, to summarise, the spheroid shapes depend both on the shape of the object and on the distance of the camera from it. To achieve this, we match the cusp of the evolute of the ellipse with the edge of the object on the axis of rotation. As the camera orientation is always normal to the spheroid surface, this makes the camera look at the edge of the object when turning around it, which guarantees a correct visualisation of the object at all points. However, we still need strategies to map two-dimensional user gestures to variations of the camera position across the spheroid, and zoom-level gestures to variations in size of the spheroid.

At this point, it is convenient to define the movements of the camera across the spheroid in geodesic (i.e. longitude and latitude) coordinates. Remember that there is a preferred axis of rotation, which we have chosen to align with the X-axis, without loss of generality. This means that vertical gestures of the user result in rotations of the camera around a global X-axis, while horizontal gestures of the user result in camera displacements on the elliptic trajectory defined by a and b, a trajectory defined on a plane which may have been previously rotated by a vertical gesture of the user[3]. Thus, gestures which are parallel to the preferred axis of rotation are always mapped to changes in latitude, while gestures perpendicular to the preferred axis of rotation are always mapped to changes in longitude (see Fig. 4).

2.1 Latitude displacements of the virtual camera

Let us start by mapping changes in latitude. A change in latitude was precisely what allowed us to find values for a and b. In order to map a horizontal gesture of a user to a movement of the camera on the ellipse defined by a and b, we consider an arbitrary initial position of the camera on the ellipse $p(x, z)$:

$$p(x, z) = (a\ \cos(t_0), b\ \sin(t_0))\ \ t_0 \in [0, 2\pi] \tag{5}$$

A gesture of the user of $\Delta\tilde{x}$ screen pixels is thus mapped to a change in the parameter Δt, which takes the camera to a new position on the ellipse $(a\ \cos(t_0 + \Delta t), b\ \sin(t_0 + \Delta t)$

[3] We make the camera orbit around the object, while the virtual trackball metaphor suggests a virtual rotation of the object. In both cases, the effect on the rendered scene is the same, but, in our case, the movements of the camera are in the opposite direction of that of the mouse cursor.

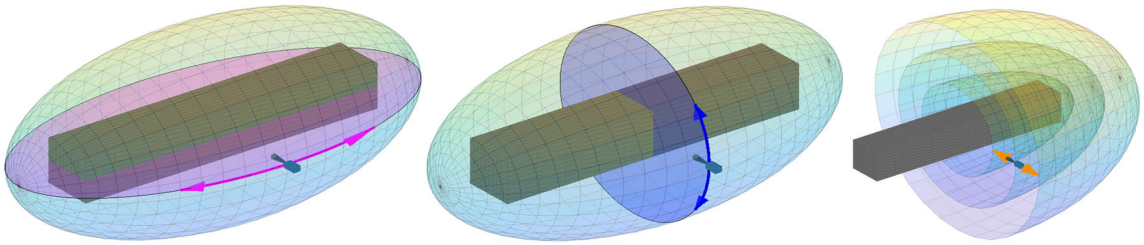


Fig. 4 Device-horizontal gestures are mapped to *latitude* displacements of the virtual camera on the spheroidal trackball, when the preferred axis of rotation is horizontal (left). Device-vertical gestures are mapped to *longitude* displacements of the virtual camera (centre), and zoom-level changes vary the size of the spheroid (right). We formulate all expressions in this section using this case, in which the preferred axis of rotation coincides with a dimension of the bounding box which is larger than the other two, resulting in a *prolate* spheroidal trackball. All expressions are given at the end of the section for the opposite case, i.e. *oblate* spheroidal trackballs (not depicted here)

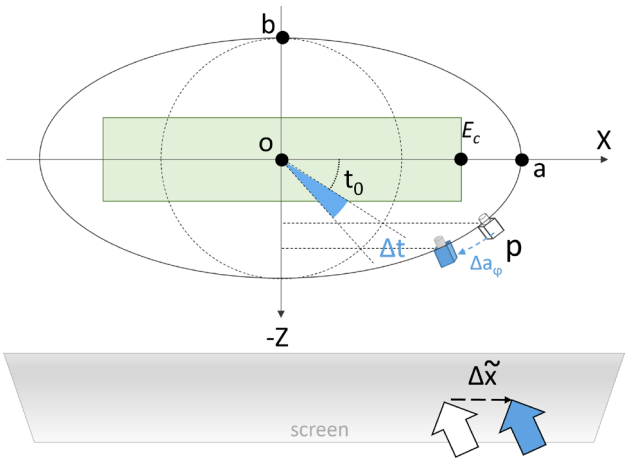


Fig. 5 Incremental user gestures $\Delta\tilde{x}$ parallel to the preferred axes of rotation (X in the example) result in incremental displacements Δa_φ that take the camera from $p(x, z) = (a\ \cos(t_0), b\ \sin(t_0))$ to $(a\ \cos(t_0 + \Delta t), b\ \sin(t_0 + \Delta t)$. In the text, we compute Δt from $\Delta\tilde{x}$, subject to usability requirements

(Fig. 5). Our problem is to compute Δt as a function of $\Delta\tilde{x}$ subject to the following requirements:

- user gestures should result in camera displacements which are independent of the screen resolution.
- user gestures should result in camera speeds which are perceived as independent of the distance between the camera and the object.

Making gestures independent of the screen resolution is straightforward. For a screen with W pixels, we compute a normalised gesture Δr simply as:

$$\Delta r = \frac{\Delta\tilde{x}}{W} \tag{6}$$

This normalised gesture r has an important property: a gesture which spans the whole screen results in a normalised gesture $r = 1$.

Let us imagine that we are inspecting a *generic* object with the shape of an spheroid that is perfectly inscribed in the bounding box. The intersection of such object with the horizontal plane is an ellipse with semi-axes e_z and e_x (Fig. 6).

The camera at p is looking in the direction of p', a point on the ellipse inscribed in the object, i.e. p' is the point on the object that we see at the centre of the screen. If the camera is close enough to the object, so that the object spans the whole screen and beyond, what we see is an arc of the inscribed ellipse which can be approximated by k_φ:

$$k_\varphi = 2 \cdot \sqrt{(x - x')^2 + (z - z')^2} \cdot \tan(\alpha) \tag{7}$$

where α is the camera's horizontal field of view. Since we know $p(x, z)$, the position of the camera, a and b, the semi-axes of the external ellipse on which the camera is moving, and e_x, e_z, the semi-axes of the inscribed ellipse, we can solve for x', z' and thus compute k_φ from 7:

$$x' = \frac{e_x^2 \cdot m_{t_0} \cdot D \pm e_z \cdot e_x \cdot \sqrt{e_z^2 - D^2 + e_x^2 \cdot m_{t_0}^2}}{e_z^2 + e_x^2 \cdot m_{t_0}^2}$$
$$z' = \pm\, b \cdot \sqrt{1 - \frac{x'^2}{e_x^2}} \tag{8}$$

where $D = (a^2 - b^2)/b \cdot \sin(t_0)$ and $m_{t_0} = a/b \cdot \tan(t_0)$ are the parameters in the equation $z = m_{t_0} \cdot x - D$ of the line passing through p and p'.

We now make use of the previous property of the normalised gestures of the user Δr. For a gesture that spanned the full width of the screen, Δr would be equal to 1. The displacement of the camera projection p' that the user expects would be equal to exactly the fragment of the object that the user sees on the screen. and that we approximated by k_φ (Eqs. 7, 8). Smaller displacements of the camera Δa_φ are thus proportional to normalised gestures exactly by k_φ:

$$\Delta a_\varphi = k_\varphi \cdot \Delta r \tag{9}$$

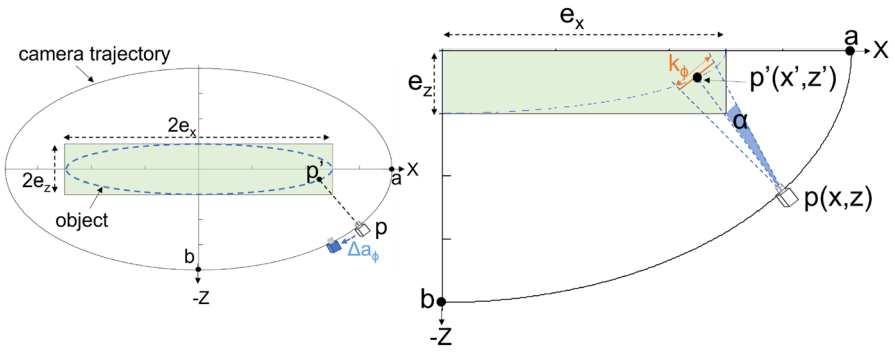


Fig. 6 An object with the shape of a spheroid is inspected by the camera. If the camera is close enough to the object, what we see is a fraction of the intersection of the object with the camera plane that can be approximated by k_φ (see text)

Let us suppose that the camera is very close to the object, and let us consider a very small camera displacement. This allows us to approximate Δt. by:

$$\Delta t \approx \frac{\Delta a_\varphi}{\sqrt{(e_x \cdot \sin t_0)^2 + (e_z \cdot \cos t_0)^2}} \tag{10}$$

Combining all of the above results in:

$$\Delta t = \Delta \tilde{x} \cdot \frac{2 \tan \alpha}{W} \sqrt{\frac{(x - x')^2 + (z - z')^2}{(e_x \cdot \sin t_0)^2 + (e_z \cdot \cos t_0)^2}} \tag{11}$$

where x', y' can be computed using Eq. 8, α is the horizontal field of view of the camera, and W is the width of the screen in pixels.

2.2 Longitude displacements of the virtual camera

We now follow a similar reasoning to compute how vertical gestures of the user should be mapped to longitude displacements of the virtual camera (Fig. 7). Longitude displacements are simpler to compute since the trajectory of the camera around the preferred rotation axis is circular, rather than ellipsoidal. Our goal is now to compute the motion of the camera $\Delta\theta$ on its circular trajectory from a vertical gesture of the user of $\Delta\tilde{y}$ screen pixels, subject to identical usability requirements, i.e. independence of screen resolution and independence of camera speed with distance between camera and object.

In a similar procedure to the one we followed to map latitude displacements, we imagine the camera looking closely at an object with the shape of the spheroid, inscribed in the bounding box. The intersection with the vertical plane is this time a circumference of radius e_z (Fig. 7).

k_λ is the arc of this circumference visible on the screen through the camera, with a simpler expression than the previously computed k_φ for latitude displacements, as the distance between p, the position of the camera, and p', the point on the object, is simply the difference in radius of the camera trajectory r_{cam} and the radius of the object e_z:

$$k_\lambda = 2(r_{\text{cam}} - e_z) \cdot tag(\beta) \tag{12}$$

where β is the vertical field of view of the camera.

The radius of the camera trajectory depends on the latitude of the camera. It can be obtained from the parametric equations of the ellipse, giving as a result: $r_{\text{cam}} = |b \cdot \sin(t)|$.

For a vertical gesture of the user of $\Delta\tilde{y}$ pixels, we follow the same reasoning as in latitude displacements to find that incremental arc displacements of the camera in longitude are proportional exactly by k_λ to the vertical displacements on the screen. Thus, the parameter $\Delta\theta$ which moves the camera on its circular trajectory can be computed as:

$$\Delta\theta = \Delta\tilde{y} \cdot \frac{2 \tan \beta}{H} \cdot \frac{|b \cdot \sin(t)| - e_z}{e_z} \tag{13}$$

where H is the height in pixels of the camera.

This mechanism adapts the longitude movements to any latitude in the ellipsoid. At the edges (maximum and minimum latitude), a longitude movement is translated into rotations of the camera around its view axis, resulting in an apparent rotation of the object in the screen which will be clockwise or counterclockwise depending on the direction of the mouse movement, but independent of the position on the screen where the mouse cursor is being moved. This leads to a somewhat undesired effect whereby the object may not follow the mouse cursor.

2.3 Zoom-level changes of the virtual camera

We have previously defined mappings between two-dimensional user gestures to variations in latitude and longitude across the spheroid. The only requirement left is the strategy to map zoom level gestures to variations in the size of the spheroidal trackball, i.e. changes to a, b in Eq. 2. The strategy to map zoom-level user gestures is straightforward: gestures $\Delta\tilde{z}$ of the user are mapped to changes Δb in one of the semi-axes of the spheroid. Let the size of the spheroid before the gesture the spheroid be defined by semi-axes a_0, b_0 in Eq. 2.

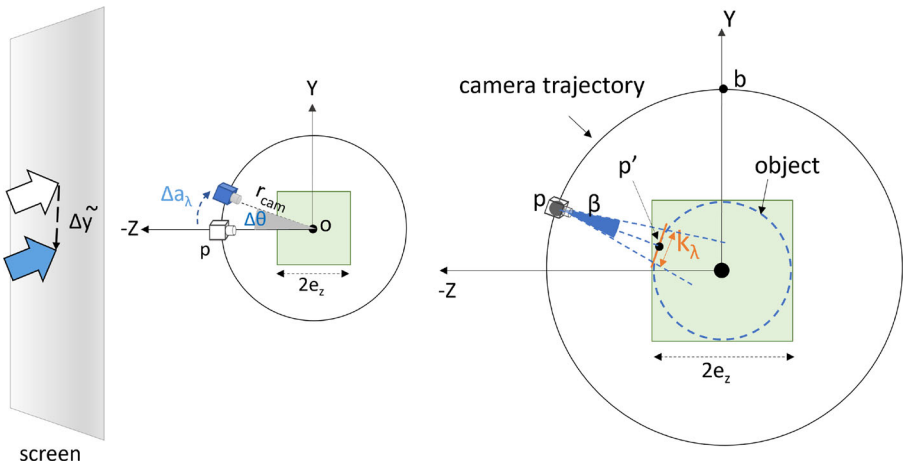


Fig. 7 The object with the shape of a spheroid is inspected by the user making a small vertical displacement. If the camera is close enough to the object, what we see a fraction of the intersection of the object with the vertical camera plane that can be approximated by k_λ (see text). To compute $\Delta\theta$, the circular motion of the camera from the vertical gesture of the user $\Delta\tilde{y}$ we use the same concept: a gesture that spans the whole screen vertically should result in a movement of p' of the object which can be approximated by k_λ

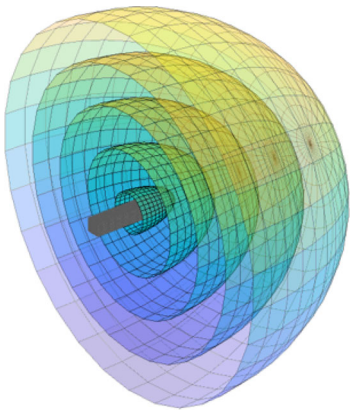


Fig. 8 Effect of the distance between the camera and the object on the shape of the ellipsoid. The furthest from the object, the more spherical the spheroid becomes. From a long distance, the spheroidal trackball defaults to the traditional spherical trackball

A gesture $\Delta\tilde{z}$ of the user expressing a zoom-level change is mapped to a change Δb so that the new semi-axis a, b are simply:

$$\begin{aligned} b &= b_0 + \Delta b \\ a &= \tfrac{1}{2}(e_x + \sqrt{e_x^2 + 4b^2}) \end{aligned} \tag{14}$$

It is interesting to note that when $b \gg e_z$, i.e. when the camera is far from the object bounding box, the spheroidal trackball becomes more and more spherical:

$$\begin{aligned} \lim_{b\to\infty} a &= \lim_{b\to\infty} \frac{1}{2}(e_x + \sqrt{e_x^2 + 4b^2}) \\ &= \lim_{b\to\infty} \frac{1}{2}(e_x + 2b) = b \end{aligned} \tag{15}$$

This is a convenient result by which from a long distance, the spheroidal trackball defaults to the classical spherical trackball (see Fig. 8).

What is left to complete the spheroidal trackball a strategy to map $\Delta\tilde{z}$ zoom-level gestures of the user to Δb changes to the spheroid. At this point, we need to find what is the *minimum spheroid* that can be defined around the object before the camera trajectory intersects the bounding box.

The minimum spheroid that just intersects the bounding box can be computed by writing the equation of the circle that goes through p and p' on Fig. 9, as well as that of the ellipse on the horizontal plane going through p'. If we impose the relationship between the spheroids semi-axis a, b that we are imposing for the spheroid trackball (Eq. 3), we are left with a cubic equation on one of the semi-axes of the spheroid with coefficients depending only on the dimensions of the bounding box of the object:

$$a^3 - E_c\, a^2 - (E_c{}^2 + r^2)\, a + E_c{}^3 = 0 \tag{16}$$

This cubic equation can be solved using the Cardano–Ferrari method. The analysis is not trivial, but given the geometric characteristics of the solution we are looking for, it is possible to consistently choose one of the solutions to the cubic equation to obtain the following expression to compute the values of the semi-axes $a_{\min}$, $b_{\min}$ of the minimum spheroid:

$$\begin{aligned} a_{\min} &= \tfrac{2}{3} \cdot \sqrt{4E_c{}^2 + 3r^2} \cdot cos(\tfrac{\Psi}{3}) + \tfrac{E_c}{3} \\ b_{\min} &= \sqrt{a_{\min}^2 - E_c\, a_{\min}} \\ \Psi &= \arcos\left(\frac{-1}{2} \frac{16E_c{}^3 - 9E_c\, r^2}{\sqrt{64E_c{}^6 + 144E_c{}^4\, r^2 + 108E_c{}^2\, r^4 + 27r^6}} \right) \end{aligned} \tag{17}$$

where E_c and r can be directly computed from the bounding box of the object. This is an elaborate expression, but only has to be computed once, for instance, when the geometry of the object is loaded by the application.

The minimum spheroid is useful to map because it provides a lower bound to the changes in the semi-axes Δb that we set out to compute at the beginning of the section. For a

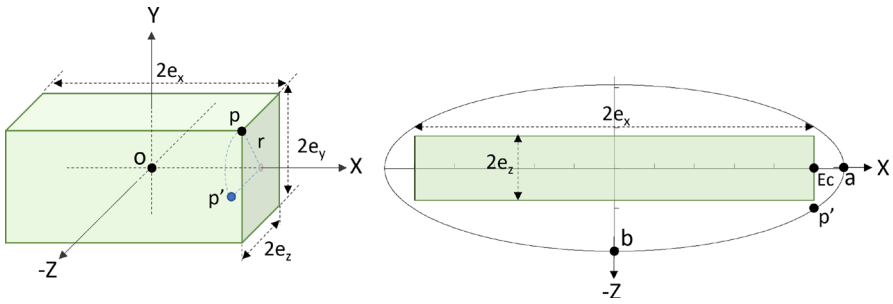


Fig. 9 The minimum spheroid trackball will just touch the bounding box in point p, which is at a radius r of the horizontal axis, which can be computed from the size of the bounding box (left). Also at this radius but on the horizontal plane is p', a point on the intersection of the spheroid with the horizontal plane (right)

given initial position of the camera on a spheroid defined by a_0, b_0, one can compute the distance between that position and the minimum spheroid defined by $a_{\min}, b_{\min}$, then map the variations $\Delta\tilde{z}$ produced the interaction device to percentages of that distance. For the kind of spheroids that we are considering in all the formulation, the expression is simply:

$$\Delta b = k_{\text{scale}} \cdot (b_0 - b_{\min}) \cdot \Delta\tilde{z} \tag{18}$$

where k_{scale} is a scale factor that will depend on the application[4]. This and the expressions in Eq. 14 allow us to compute the spheroidal trackball at the new zoom level.

2.4 Generalisation to oblate spheroidal trackballs

The formulation above is general to scaling of the classical spheroidal trackball in any of the axis. However, we have given expressions in some cases that assume that the semi-axis a is greater than b (aligned, respectively, with global X- and Z-axes for convenience). This is to say that the formulation above in some cases assumes that the axis that is scaled is greater than the other two, which are equal but smaller. These are called prolate spheroids, having the shape of a rugby ball. We generalise all expressions in this section to oblate spheroids, where the semi-axis which is scaled is smaller to the other two, and the shape is that of a Pilates ball (when someone sits on it). Table 1 has a twofold purpose: to generalise the formulation to oblate spheroids, and to have all the practical expressions summarised together and separated from other, secondary expressions used in the formulation.

The formulation of the spheroidal trackball is an alternative framework which to the traditional approach to orbiting a virtual camera around an object. In the remaining of the paper, we seek to evaluate this framework in an experiment which test our hypotheses.

[4] The exact value for k_{scale} depends also on whether the interaction device is co-located with the screen, as in touchscreens, or not, as in mouse wheels, where the mapping is arbitrary: the mouse-wheel *scroll delta* is itself an integer value returned by the mouse driver software without meaningful units.

3 Evaluation of the spheroidal trackball

Once the spheroidal trackball is formulated, our goal is now to evaluate if this framework will improve interaction, in terms of higher performance, increased perceived usability and reduction of perceived workload.

3.1 Overview

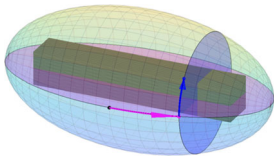
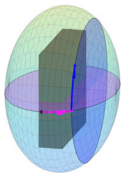
We designed an experiment in which participants had to use a software application to complete a series of inspection and docking tasks. The object inspected was a 3D model of a gas turbine, a horizontally elongated object which strongly suggests a horizontal axis of rotation, which coincides with its longest dimension. All subjects had to perform a search-and-dock task of a small target on the surface of the object. First they have to find it, then align it with a viewfinder. The task was performed several times using different navigation conditions. In one condition the navigation used a spherical trackball, and in the other a spheroidal trackball adapted to the surface of the object. Our hypothesis was that using the spheroidal trackball will benefit the usability of the application, in terms of increased performance (less time to complete the task), higher perceived usability and lower perceived workload.

3.2 Inspection and docking task

Each participant had to complete a combined inspection and docking task a number of times on the same virtual object: a gas turbine, an elongated object with enough three-dimensional detail so as to hide from the viewer a small yellow sphere (target) depending on the orientation of the camera. The goal for the subjects was thus to (1) find the yellow target sphere and (2) dock the target inside the viewfinder. The viewfinder has two concentric circles. Docking is completed when the projection on the camera view of the yellow target sphere is within the ring-shaped area between the circles (Fig. 10)

Across iterations of the task, the yellow target sphere appeared on manually pre-defined locations on the object. When docking is completed, the viewfinder changes colour

Table 1 Summary of expressions for prolate and oblate spheroidal trackballs

Trackball	Prolate spheroid	Oblate spheroid
		
Equation of camera orbit	$\frac{x^2}{a^2} + \frac{y^2+z^2}{b^2} = 1\ ,\ a > b$	$\frac{x^2}{b^2} + \frac{y^2+z^2}{a^2} = 1\ ,\ a > b$
Bounding box ($2e_x \times 2e_y \times 2e_z$)	$e_y = e_z$	also $e_y = e_z$
	$e_x > e_y$	but $e_x < e_y$
Solving for a, b	$b = Z_0$	$a = Z_0$
	$a = \frac{1}{2}(e_x + \sqrt{e_x^2 + 4b^2})$ $(*)$	$b = \sqrt{a^2 - a \cdot e_z}$ $(**)$
Latitude displacements (Δt)	$\Delta\tilde{x} \cdot \frac{2\tan\alpha}{W}\sqrt{\frac{(x-x')^2+(z-z')^2}{(e_x \cdot \sin t_0)^2+(e_z \cdot \cos t_0)^2}}$	same expression, but
		swap $e_x \longleftrightarrow e_z$
Longitude displacements ($\Delta\theta$)	$\Delta\tilde{y} \cdot \frac{2\tan\beta}{H} \cdot \frac{r_{\text{cam}}-e_z}{e_z}$	no change
Zoom-level changes (Δb or Δa)	$\Delta b = k_{\text{scale}} \cdot (b_0 - b_{\text{min}}) \cdot \Delta\tilde{z}$	$\Delta a = k_{\text{scale}} \cdot (a_0 - a_{\text{min}}) \cdot \Delta\tilde{z}$
	$b = b_0 + \Delta b$, then solve for a	$a = a_0 + \Delta a$, then solve for b
	using expression above $(*)$	using expression above $(**)$

The quantities $\Delta\tilde{x}$, $\Delta\tilde{y}$ and $\Delta\tilde{z}$ are horizontal, vertical or zoom-level gestures. W and H are the display's width and height in the same units as the gestures. Other quantities used in expressions (a_{min}, b_{min}, α, β, x' or z') are explained in the text of this section

for one second, and a sound suggesting success is emitted by the application. To proceed to the next iteration, the subject had to press the space bar. A decreasing counter helps the user understand how many repetitions are left for a given trial. More detail is given in Sect. 3.4.

3.3 Experimental design

We designed an experiment with two independent variables. The first variable has two within-subject conditions. Condition 1 employs the spheroidal trackball (introduced in this paper) to orbit the virtual camera, while Condition 2 corresponds to the traditional spherical trackball. In spheroidal navigation, the subjects were allowed horizontal rotation, vertical rotation and zoom-level gestures. In spherical navigation, the users were also allowed to make pan gestures. Each of the participants performed the study twice with each condition in a counterbalanced order. The participants did not receive demonstrations or explanations about the navigation techniques. They were told that they were going to try four variants of a technique to manipulate three-dimensional objects on the computer, and they were explained the task that they had to perform.

The second variable has three levels: the targets were set out to uniformly occupy three types of locations on the object surface. From an initially centred, neutral position of the virtual camera, we classified each area of the object surface into three categories: (1) Central: targets are visible and centred on the object (2) Lateral-visible: targets are visible but towards the sides of the object, and (3) Lateral-hidden, towards the sides of the object and hidden by the blades of the object. The object has enough detail in this area to hide the target, unless the camera orbits or pans towards it). Figure 10 attempts to illustrate this classification with examples. The targets were arranged so that, at each trial, the target was hidden and in a different area according to the previous classification. This strategy makes sure that the participants always had to navigate the camera around the object to find the following target. Four sequences were manually designed to meet these constraints. Across participants, the mapping between block and sequence was also counterbalanced using Latin squares.

Figure 11 shows the structure of the full participation of one subject in the evaluation. Once the demographic questionnaire and the informed consent are filled in, subjects completed four blocks, two for each condition. The order in which each condition appeared was counterbalanced using Latin squares. Each block was preceded by a training phase that allowed the users to practise the inspection and docking task for as long as they wished. The word 'Training' appeared during this stage, instead of the counter showing how many targets where left to find. Once the participants felt they were ready, they clicked on a button on the corner of the screen to signal the end of the training. The scenario illustrated in Fig. 10a was loaded and the block started. Each block includes 18 trials of the inspection and docking task. Each trial starts when the user presses the space bar on the

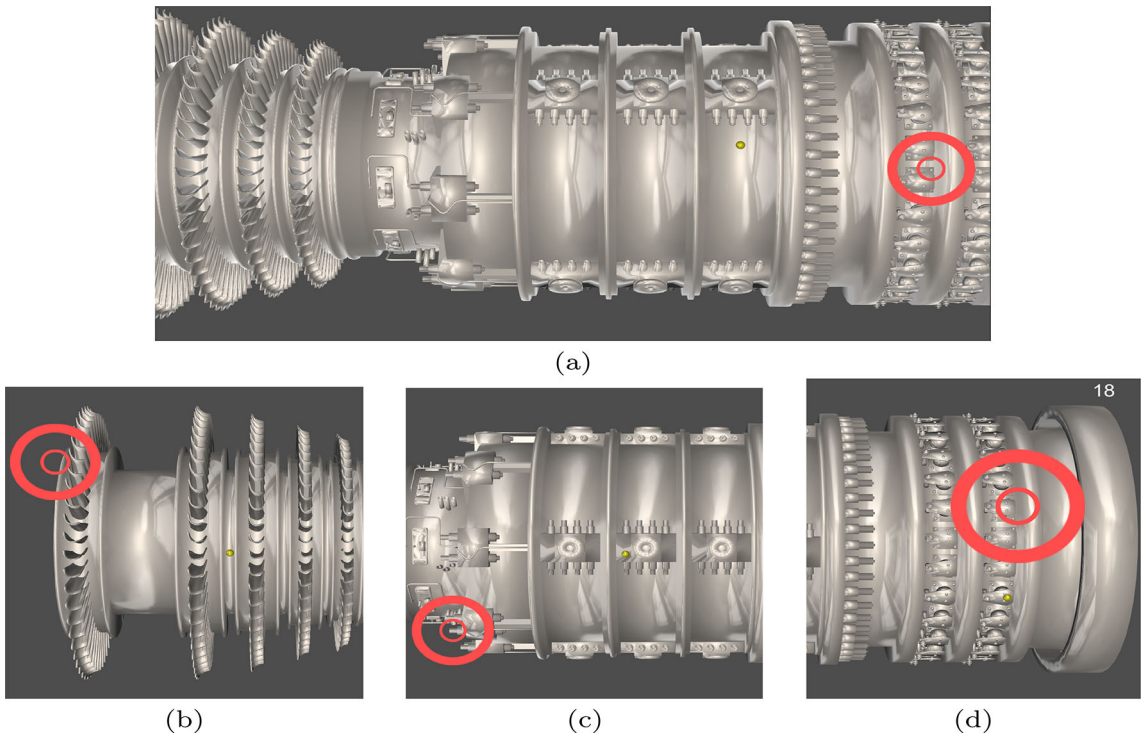


Fig. 10 **a** shows the 3D model of the untextured gas turbine with a yellow sphere (target) and the ring-shaped viewfinder. **b–d** shows examples of the three zones in which target can appear. From left to right, lateral-hidden, central and lateral-visible. The figure also illustrates how the viewfinder changes position and size between trials

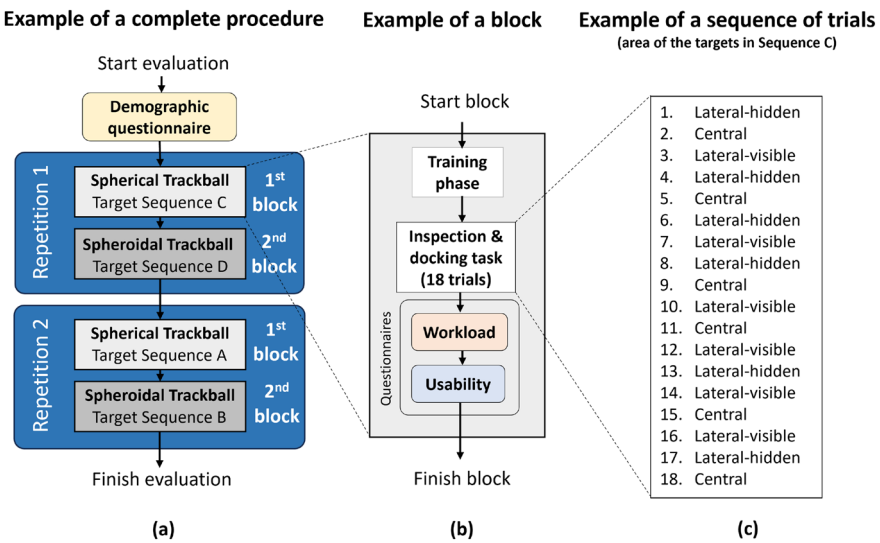


Fig. 11 Experimental design. **a** Shows the complete procedure of the experiment. **b** Shows the procedure of a block. **c** Shows an example of a sequence of the 18 trials performed in one block, it lists the areas where the target is placed in the Sequence C. The conditions and sequence order for participant 3 are shown as an example

keyboard. The position of the target for each trial was selected by the software using a strategy that forced subjects to always use the navigation technique of the block to look for the target. The strategy is detailed in Sect. 3.4. The participants were asked to complete each trial as fast as possible. At the end of each block, they filled in a usability questionnaire, followed by a perceived-workload questionnaire.

3.4 Experimental apparatus

The evaluation took place in a closed room. The participants used a Lenovo Legion Y530-15ICH laptop with an Intel i5 CPU and 16GB of RAM, running Windows 10. The software application was developed using Unity3D. The core of the software is HOM3R, a 3D viewer for complex hierarchical product models [12]. The application run at full screen with

a resolution of 1920×1080 pixels. The interaction was done with the mouse, which rested on the same table as the laptop. The space bar of the laptop keyboard was used to start each trial. There were three gestures that the user could perform with the mouse: (1) Zoom in or out on the object using the mouse wheel. Certain minimum and a maximum zoom levels were set to avoid the user to completely lose the object. (2) Orbit around global horizontal and vertical axis through dragging gestures with the left button of the mouse. (3) Pan horizontally across the object through dragging gestures of the middle mouse button (the wheel), only for the spherical trackball condition. At each trial, the viewfinder could appear anywhere on the screen, with different sizes (Fig. 10), so as to force participants to use all the available gestures of the interaction device.

3.5 Participants

A total of 32 individuals participated in the evaluation, of whom 20 identified as men, 11 as women and one as nonbinary. Ages were between 18 and 49 ($M = 24.16$, SD $= 3.65$). Thirty-one participants were 18–29 years old and one was 40–49. None of the participants reported any uncorrected visual acuities or impairments. All subjects were offered a small 3d-printed gift in return of their participation. No experience with 3D modelling or similar software was required. However, the demographic questionnaire showed that 68.75% of participants had experience with 3D modelling software, while 78% of participants had used video games. The participants signed their informed consent, and the whole evaluation procedure was reviewed and approved by the Experimentation Ethics Committee of the University of Malaga..

3.6 Data collection and analysis

Our goal was to evaluate whether there were differences between using the traditional spherical trackball and the spheroid trackball introduced in this paper. For this we measured performance, perceived usability and perceived workload.

We measured performance in terms of the time that it took participants to find and dock the target in the viewfinder. We averaged the time for each participant and each condition across the 18 trials. We also computed average times for the six trials within each object area (central, lateral-visible and lateral-hidden). We computed geometric time averages to compensate for the excessive time that some users take to complete trials [22]. Perceived usability was measured with a ten item SUS questionnaire [7]., translated to Spanish by Devin [11]. The participants scored each question with a 5-point Likert scale which ranged from 'Strongly Disagree' to 'Strongly Agree'. The answers were processed to obtain a SUS score [21] of how usability is perceived by the participant. Perceived workload was measured with the Raw TLX [15], in its Spanish translation by [10]. The questionnaire has six questions that are answered in a 21 point scale—from very low to very high. The RTLX score was the average for each participant and condition.

We computed a within-subject, two-factor ANOVA where the two factors were the navigation technique (spherical or spheroid) and the position of the target (central, lateral-visible and lateral-hidden) to assess if there was an effect in performance of the navigation techniques depending on whether the target was more difficult to search and dock. In addition, we computed a paired-samples T test to assess if there were statistically significant differences in usability and workload perceived by participants.

3.7 Results

3.7.1 Task performance

To estimate performance, we measured the time that participants took to complete the search-and-dock task detailed in previous sections, and thus a smaller value corresponds to a better performance. We found a significant effect in task performance of the navigation technique (spheroid vs. spherical). For spheroidal navigation, the mean was $8.91^{+0.47}_{-0.44}$ s,[5] while for spherical navigation the mean was $14.25^{+0.923}_{-0.85}$ s. The estimated difference between both techniques was $5.34^{+0.78}_{-0.72}$ s. (Fig. 13, All areas).

We computed a two-factor ANOVA to find that there was an effect of the navigation technique ($F(1, 31) = 292.586$, $p < 0.001$). We also found that there was an effect of position ($F(2, 62) = 97.408$, $p < 0.001$), and an interaction effect between position and navigation technique ($F(2, 62) = 11.721$, $p < 0.001$).

We performed separated T tests to assess the effect of performance of the navigation technique in each of the areas of the object. We found an effect in all areas:

- Lateral-hidden: The t value was $t(31) = -14.263$, $p < 0.001$. The means were $9.54^{+0.66}_{-1.32}$ for spheroidal navigation, and $16.92^{+1.31}_{-1.22}$ for spherical navigation.
- Central: The t value was -10.251, $p < 0.001$ and the means were $7.8^{+0.46}_{-0.43}$ and $11.34^{+0.88}_{-0.81}$, respectively, for spheroidal and spherical navigation.
- Lateral-visible: The t value was -14.590, $p < 0.001$ and the means were $9.5^{+0.52}_{-0.48}$ for the spheroidal, and $15.07^{+1.07}_{-1.01}$ for the spherical.

[5] Note that it is conventional to use the superscript–subscript notation for plus or minus one standard deviation. Instead, we use it to report 95% confidence intervals on the measured times [9].

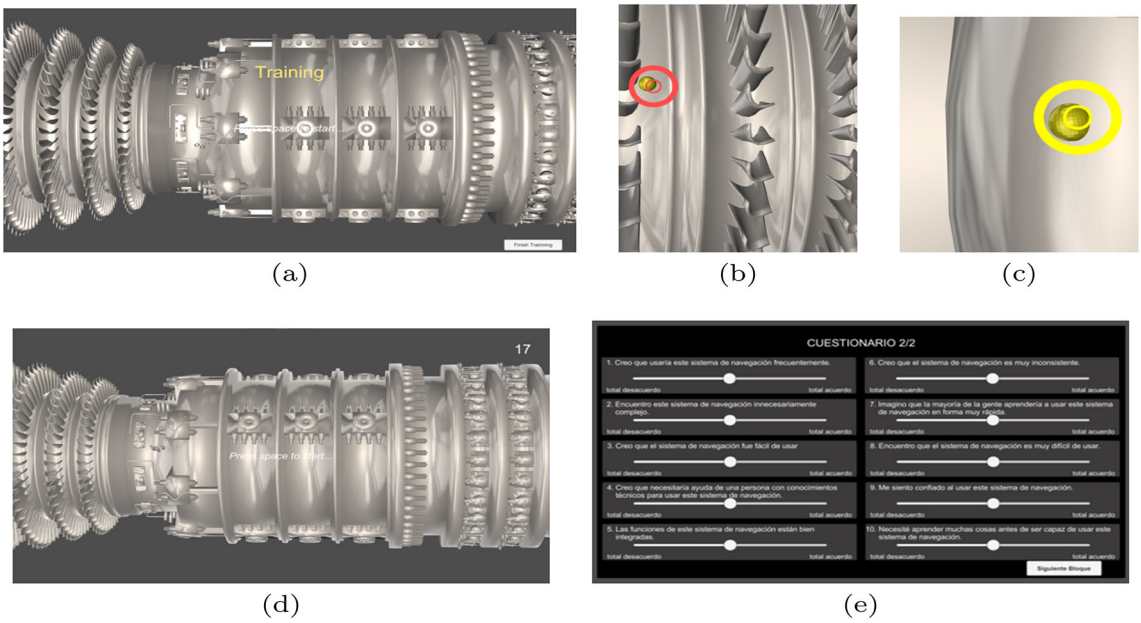


Fig. 12 Software used for the evaluation. From the top left image, clockwise: The users are allowed to train with the application for as long as required. At each trial, the viewfinder can appear anywhere on the screen, with different sizes. Once each block is finished, the software presents the user with the usability and workload questionnaires

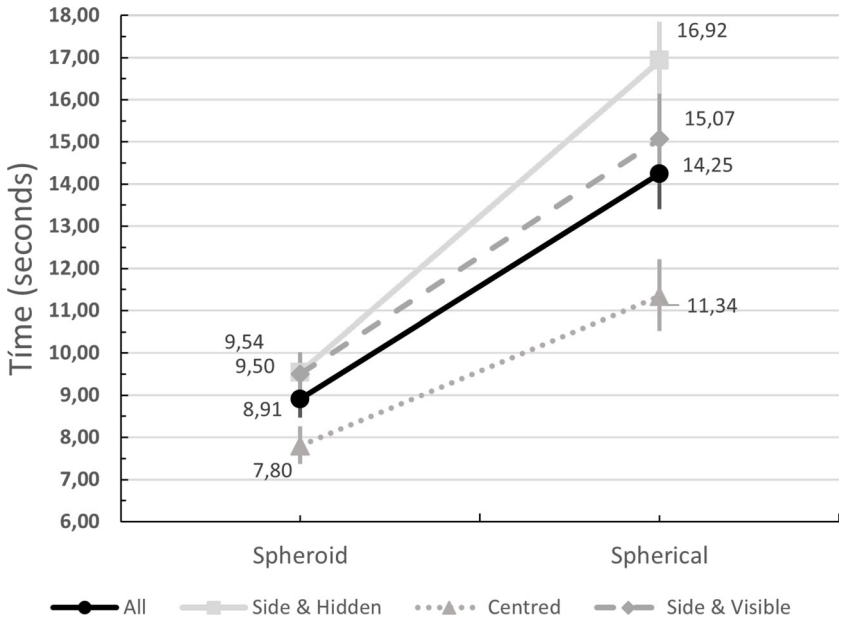


Fig. 13 Estimated means and CIs for the time it took participants to complete the inspection and docking task. The horizontal axis represents the navigation technique, and there are different plots for each of the object areas. All error bars are 95% CIs

Figure 13 depicts the means and CIs separately for each object area.

3.7.2 Perceived usability and workload

We also found a significant effect in the perceived usability as measured by the SUS score $t(31) = 8.486$, $p < 0.001$. The mean score for spheroidal navigation was $83.79^{+3.68}_{-3.68}$, while for spherical navigation it was $69.49^{+4.3}_{-4.3}$ (Fig. 14a). Note that this time, a larger value in the SUS score is better.

Finally, we also measured a significant effect of the navigation technique in the workload reported by participants via the Raw TLX questionnaire. $t(31) = -6.631$, $p < 0.001$. When participants used the spheroidal navigation, the mean in reported workload was $29.56^{+4.3}_{-4.3}$ points. When they used the spherical navigation, the mean reported workload was $41.56^{+5.66}_{-5.66}$ (Fig. 14b). This time, less is better.

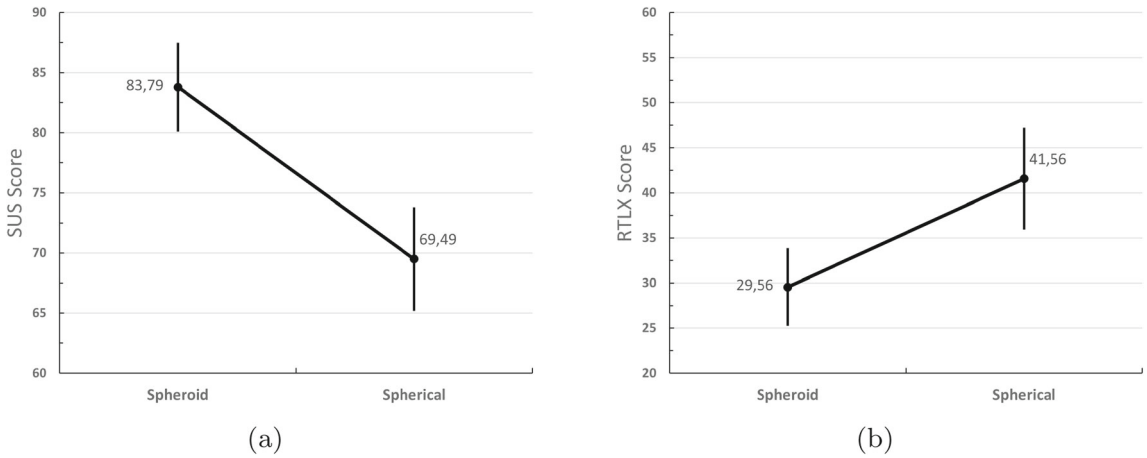


Fig. 14 **a** Means and CIs of the usability scores for each navigation technique. A larger score in the SUS test indicates better perceived usability. **b** Perceived workload in mean and CIs of the points obtained in the Raw TLX questionnaire (less is better). All error bars are 95% CIs

4 Discussion

4.1 On some of the choices we made to evaluate the spheroidal trackball

We selected a virtual object that represents a power turbine. The object has two features that make it suitable for evaluating spheroidal navigation: it is elongated, and has an intrinsic rotation axis. Recent work has shown that objects having a clear rotation axis in the user mental model allow fixed-axis navigation techniques to provide a more natural and effective interaction [13]. The rotation axis of the turbine is also horizontal, making the prolate spheroidal trackball the adequate choice for evaluation.

Spheroidal navigation allows reaching the different areas of the object to be reached by orbiting around the object, while maintaining a more constant distance to the object than spherical orbiting. In contrast, in the traditional way of interaction, when we need to *zoom* the camera near the object, we do so towards its centre, leaving the sides of the object unreachable. To reach the extremes we need to perform a translation of the centre of the object, what is known as a *pan* gesture. The spheroidal navigation saves the user from having to combine orbit and pan gestures to inspect the object. This is why we chose to separately compare both techniques for targets at the centre and at the sides of the object.

4.2 On the results in task performance in the different areas of the elongated object

Targets at the centre of the object are accessible by both techniques without panning. This explains a smaller difference in performance between spheroidal and spherical navigation, which was expected. However, the difference is still significant. Subjects seem to be using panning gestures at the centre areas of the object even though they do not strictly need it, and paying in performance. Firstly, panning is available: if the subjects can use it, they will. And it is not just a matter of availability: panning can simplify the last part of the task, docking, as the correspondence between screen gestures and camera motion is more direct. On the other hand, the closer the camera to the surface of the object, the more similar is spheroidal orbiting to panning, which makes docking also simpler, without incurring the performance loss of switching between gestures.

At the sides of the object, the difference is even stronger, as panning is indispensable in spherical navigation to reach this areas. For targets that are at the lateral area but not hidden, panning is only required for docking, but it is not imperative to locate them, as a zoom out gesture can be sufficient to reveal their location. However, for targets hidden in the turbine blades panning can be required just to find where the targets are. This can explain the different results in performance: A maximum advantage for spheroid navigation for targets hidden in the lateral blades of the turbine; a slightly smaller, but still large advantage for spheroid in lateral, non-hidden targets; a smaller advantage for targets in the centre of the turbine. Still, in all areas the advantage of spheroidal navigation is significant, showing a consistent improvement in performance over the traditional approach based in spherical navigation.

4.3 On the results in perceived usability and workload and the degrees of freedom of each navigation technique

The reduced degrees of freedom (DoF) of spheroidal navigation can also explain its advantage in perceived usability and perceived workload. These results can be discussed in the context of the classic theory of Berstein for motor skill

learning [3]. In his work, Berstein formulates the motor problem as that of choosing among a redundant set of DoF. Even for the apparently simplest tasks, such as throwing a ball, not only an enormous number of DoF of the human biomechanichal system have to be coordinated, but also, somehow, selected, as the set of DoF is redundant, i.e. there are multiple combinations that may produce the same motor result. How is then even possible that we can solve problems with so many numbers of variables, and become dexterous at performing tasks by training? There are many theories as to how this is accomplished, but Berstein suggests that at the early stages of learning, we employ different mechanisms to *freeze* as many DoF as possible in order to solve the motor problem. This hypothesis has extended by others in human and robotic motor skill learning [4]. Spheroidal navigation constitutes a similar strategy by which some of the navigation DoFs are frozen to simplify the problem of accessing the periphery of objects that depart from sphericity.

Let us think of an example of classical navigation based on a spherical trackball. We are looking from a small distance at the central area of an elongated object such as the turbine employed in the evaluation, and from here we need to travel to the left side of the object. In principle, we have a choice in the gesture that we can make: we can either orbit or pan the camera towards our target. Indeed these two actions have to be mapped to different buttons or the mouse, or require us to use an additional key on the computer keyboard. There are redundant DoFs to solve the problem, as in Berstein's formulation. If we are close to the object and our target is far in the edge of the object, orbit might altogether the wrong choice, as we will not put the target in reach of the virtual camera. On the other hand, with spheroidal navigation panning is not required, as spheroidal orbiting will maintain the camera at a close, almost constant distance to the surface of the object, resulting in an efficient interaction that uses a reduced set of DoFs.

We can also frame our proposal in the classic Schema Theory [23]. According to this model, we retrieve a general motor programme from memory in order to solve a new motor problem. To retrieve the programme we use as inputs the initial and the final, desired situation, then we adapt the general motor programme to the specific conditions of the problem. In our example, in the case of spherical navigation, the retrieved schema involves orbiting and panning. In spheroidal navigation, the retrieved programme will involve only orbiting, in a schema which is simpler and thus easier to acquire.

Other strategies mentioned in the introduction, such as HoverCam [16], are designed to maintain a constant distance between the orbiting camera and the virtual object surface. Precisely because they are based on the distance to the actual object surface, they are vulnerable to traversing every feature and result in a jittering effect on the camera image during movement. Our approach encloses the object within a smooth surface—the spheroid—providing an adaptation, that, while not perfect, consistently ensures smooth camera movements, eliminating any potential shakiness. Finally, our proposal requires a non-uniform motion of the camera. This was specifically designed to approximate a one-to-one movement of the object in the screen with the mouse (this approximation would be exact if the object is an spheroid). Our evaluation shows that, despite this non-uniform motion, this approach yields to positive results on the user experience in terms of usability.

5 Towards the ellipsoidal trackball

In this paper, we have extended recent work to generalise the fixed trackball by relaxing the implicit assumption that the objects inspected have high sphericity. We have started by considering objects with a bounding box in which only one of the three dimensions is larger or smaller than the other two. The resulting trackball is spheroidal. A spheroid is a surface constructed by rotating an ellipse around one of its axis. If we rotate it around the longer axes, we obtain a prolate spheroid, but if we rotate it around the shorter axes, we obtain an oblate spheroid. For both cases, we have formulated equations that map device gestures performed by the user to orbital motion on the spheroid (latitude and longitude displacements), and re-computations of the spheroidal surface, which map zoom-level gestures of the user.

An evaluation performed with 32 subjects in a search-and-dock task strongly suggests that the spheroidal trackball outperforms the classical approach based on the spherical trackball, in terms of performance (time to complete the task), perceived usability and perceived workload.

We have previously discussed [13] practical implications for applications willing to take profit of the improvements in performance, usability and fatigue perception of using a fixed rotation axis that is consistent with a mental model of a user about the object. This can be generalised to the framework presented in this work. One promising option is the use of visualisation widgets [5] to interactively fix an axis for navigation around objects that are arbitrarily oriented in space. Once this axis is fixed, it is possible to compute a transformation of the coordinate system in which the bounding box of the object would fit either the prolate or the oblate spheroid formulations in Table 1.

The next step is to allow for objects that have a bounding box with different sizes in all dimensions. Such objects will require an *Ellipsoidal Trackball*. An ellipsoid is, in general, a surface that is not obtained by rotating an ellipse around one of its axis, as opposed to an spheroid or ellipsoid of revolution. One can obtain such shape from a sphere by performing scale deformations in all three dimensions. Mapping

user device gestures while constraining the virtual camera to an ellipsoid requires extending the formulations in this paper, and performing a new evaluation on objects that depart from sphericity in all dimensions. There are multiple examples of such objects: vehicles, pieces or furniture, or buildings. In future work, we aim to formulate the ellipsoidal trackball and evaluate its performance against both the traditional approaches based on spherical trackballs, and on the strategy introduced in this paper, the spheroidal trackball.

Funding Funding for open access publishing: Universidad Málaga/CBUA. This work was partially supported by the PLUGGY project (https://www.pluggy-project.eu/), European Union's Horizon 2020 research and innovation programme under grant agreement No 726765, and the Spanish National Project SAVLab, under grant No. PID2019-107854GB-I00, funded by MCIN/AEI/ 10.13039/ 501100011033/ FEDER UE. The data and the application used to conduct the experiment are available upon request.

References

1. Autodesk: 3ds Max—3d Modelling, Animation and Rendering Software (2012). https://www.autodesk.com/products/3ds-max/overview, http://usa.autodesk.com/3ds-max/
2. Bade, R., Ritter, F., Preim, B.: Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In: Smart Graphics, vol. 3638, pp. 138–150. Springer, Berlin (2005). https://doi.org/10.1007/11536482
3. Bernstein, N.A.: The Coordination and Regulation of Movement. Pergamon Press, New York (1967)
4. Berthouze, L., Lungarella, M.: Motor skill acquisition under environmental perturbations: on the necessity of alternate freezing and freeing of degrees of freedom. Adapt. Behav. **12**(1), 47–64 (2004)
5. Besançon, L., Ynnerman, A., Keefe, D.F., Yu, L., Isenberg, T.: The state of the art of spatial interfaces for 3D visualization. Comput. Graph. Forum **40**(1), 293–326 (2021). https://doi.org/10.1111/CGF.14189
6. Blender Foundation: blender.org—Home of the Blender project—Free and Open 3D Creation Software (2015). https://www.blender.org/
7. Brooke, J.: SUS: a "quick and dirty" usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, A.L. (eds.) Usability evaluation in industry. Taylor and Francis, London (1996)
8. Chen, M., Mountford, S.J., Sellen, A., Chen, M., Mountford, S.J., Sellen, A.: A study in interactive 3-D rotation using 2-D control devices. ACM SIGGRAPH Comput. Graph. **22**(4), 121–129 (1988)
9. Cowan, G.: Statistical Data Analysis, 1st edn. Clarendon Press, Oxford (1998)
10. de Arquer, I., Nogareda, C.: Estimación de la carga mental de trabajo: el método NASA TLX. Notas técnicas de prevención. Instituto Nacional de Seguridad y Salud en el Trabajo. Gobierno de España **NTP 544** (2001)
11. Devin, F.: Sistema de Escalas de Usabilidad: ¿qué es y para qué sirve? | UXpañol. http://uxpanol.com/teoria/sistema-de-escalas-de-usabilidad-que-es-y-para-que-sirve/
12. Gonzalez-Toledo, D., Cuevas-Rodriguez, M., Flores-Holgado, S.: Collaborative management of inspection results in power plant turbines. In: Grösser, S.N., Reyes-Lecuona, A., Granholm, G. (eds.) Dynamics of Long-Life Assets: From Technology Adaptation to Upgrading the Business Model, pp. 193–208. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-45438-2_11
13. Gonzalez-Toledo, D., Cuevas-Rodriguez, M., Molina-Tanco, L., Reyes-Lecuona, A.: Still room for improvement in traditional 3D interaction: selecting the fixed axis in the virtual trackball. Vis. Comput. (2022). https://doi.org/10.1007/S00371-021-02394-X
14. Google SketchUp: 3D Design Software 3D Modeling on the Web SketchUp (2019). https://www.sketchup.com/, https://www.sketchup.com/page/homepage
15. Hart, S.G.: Nasa-task load index (NASA-TLX); 20 years later. Proc. Hum. Factors Ergon. Soc. Ann. Meet. **50**(9), 904–908 (2006). https://doi.org/10.1177/154193120605000909
16. Khan, A., Komalo, B., Stam, J., Fitzmaurice, G., Kurtenbach, G.: HoverCam: interactive 3D navigation for proximal object inspection. In: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, Vol. 1, No. 212, pp. 73–80 (2005). https://doi.org/10.1145/1053427.1053439
17. Malomo, L., Cignoni, P., Scopigno, R.: Generalized trackball for surfing over surfaces. In: Proceedings of the Conference on Smart Tools and Applications in Computer Graphics, pp. 89–97 (2016). https://doi.org/10.2312/stag.20161368
18. Marchand, E., Courty, N.: Controlling a camera in a virtual environment. Vis. Comput. **18**(1), 1–19 (2002). https://doi.org/10.1007/S003710100122
19. Official Google, B.: 3D Warehouse (2016). https://3dwarehouse.sketchup.com/
20. Rybicki, S., DeRenzi, B., Gain, J.: Usability and performance of mouse-based rotation controllers. In: Proceedings—Graphics Interface, pp. 93–100. ACM, Victoria, British Columbia, Canada (2016). https://people.cs.uct.ac.za/~jgain/publications/rotationcontrollers.pdf
21. Sauro, J.: MeasuringU: Measuring Usability with the System Usability Scale (SUS) (2011). https://measuringu.com/sus/
22. Sauro, J., Lewis, J.R.: Average task times in usability tests. In: Proceedings of the 28th International Conference on Human factors in Computing Systems—CHI '10, p. 2347. ACM Press, New York (2010). https://doi.org/10.1145/1753326.1753679
23. Schmidt, R.A.: A schema theory of discrete motor skill learning. Psychol. Rev. **82**(4), 225–260 (1975). https://doi.org/10.1037/h0076770
24. Shoemake, K.: ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In: Proceedings of the conference on Graphics interface '92, pp. 151–156. Canadian Information Processing Society, Vancuver (1992). https://dl.acm.org/citation.cfm?id=155312
25. Shoemake, K.: Arcball rotation control. In: P.S. Heckbert (ed.) Graphics Gems IV, Chapter III.1, pp. 175–192. AP Professional, San Diego (1994). https://dl.acm.org/citation.cfm?id=180910
26. Sketchfab: Sketchfab - Publish & find 3D models online (2019). https://sketchfab.com/
27. Thornton, R.W.: The Number Wheel. In: Proceedings of the 6th Annual Conference on Computer Graphics and Interactive

Techniques—SIGGRAPH '79, vol. 13, pp. 102–107. ACM Press, New York (1979). https://doi.org/10.1145/800249.807430
28. Weisstein, E.W.: Ellipse Evolute—From Wolfram MathWorld (2020). https://mathworld.wolfram.com/EllipseEvolute.html
29. Weisstein, E.W.: Evolute—From Wolfram MathWorld (2020). https://mathworld.wolfram.com/Evolute.html

Daniel González-Toledo earned his M.Sc. degree in Telecommunications engineering in 2013 at the University of Malaga (Spain). He started his career as an engineer at IBM Global Services in 2012, before that he was working in different companies as IT staff to pay for his university studies. At the end of 2013, he started his academic career in the department of Electronic Technology at the University of Malaga. Since then, he has worked as a researcher on four different European projects. He is currently working on a European Project (SONICOM) focused on 3D audio spatialization while pursuing his Ph.D. in the field of 3D interaction techniques. During his Pd.D. studies he has done research stays at Imperial College London (United Kingdom) and at microwave and fiber optics laboratory of the National Technical University of Athens.

María Cuevas-Rodríguez earned her M.Sc. degree in Telecommunications engineering in 2011 and M.Sc. degree in Mechatronic in 2013 at the University of Malaga (Spain). She moved for a year to Tatung University (Taiwan) in 2009/2010. In 2011, she started her career in the Department of Electronic Technology at the University of Malaga. Since then, she works as researcher in different European Projects, on topics related to 3D interaction with reduced DoF and 3D binaural audio. She obtained her PhD degree in Telecommunication Engineering in 2022. During her Ph.D. she has been working in the Imperial College London (UK) for 3 months as visiting student and in Facebook Reality Laboratory (Seattle, EEUU) for 6 months with an internship. She is currently working on a European Project (SONICOM - https://www.sonicom.eu/) focused on 3D audio spatialization.

Luis Molina-Tanco is a Lecturer at the University of Malaga (Spain) and a researcher at the DIANA group. He studied Telecommunication Engineering (BSc+MSc) at Universidad Politécnica de Madrid (Spain), worked in the telecommunications industry and then went back to academia to undertake a PhD in Computer Science at the University of Surrey (UK), supervised by Professor Adrian Hilton. In 2003 he joined the Department of Electronic Technology (DTE) at the University of Malaga, as Researcher and Lecturing Assistant, and as a Lecturer since 2007. Since 1996 he has worked in national and EU funded projects in the fields of HCI, pattern recognition, robotics, and virtual reality, spanning applications in telemedicine, performing arts, manufacturing, and cultural heritage.

Arcadio Reyes-Lecuona is an Associate Professor and the head of DIANA research group at the University of Malaga (Spain). He obtained his PhD degree in Telecommunication Engineering in 2001. He is a Telecommunication Engineer since 1995 (BSc+MSc) and Psychologist since 2017 (BSc+MSc). During the last 20 years, he has been working in VR, haptics, 3D interaction and 3D audio. He has participated in several EU and national projects and has been PI in three national projects and five European projects. His research interests are in HCI in VR, including 3D interaction with reduced DoF and 3D binaural audio, being one of the coordinators of the 3D Tune-In audio toolkit, an open-source library for 3D audio rendering. He is Vice-President for Communication of EuroXR (European association of Extended Reality) and member of the steering committee of AIPO (Spanish-speaking association of HCI).