Project MATVJII: Visualizing 3D data

MATVJII

November 23, 2017

1 Introduction

This project is about how to develop a tool to control the orientation of 3D shapes by using a mouse as input. This feature is implemented in many 3D-design software to help users to visualize 3D shapes in several spatial orientations. Warranting a proper and natural user experience using a 2D device as input for controlling the orientation of a 3D body that can rotate arbitrarily about 3 independent axis is a hard task.

What the `arcball` tool does is to define rotations based on the user input, apply the rotation to the 3D model (rotate its frame) and finally redraw the object refreshing the image on the screen. When this process is ran at high rates, it is possible to create the effect of interaction between the user and the 3D model.

You will use MatLab and GUIDE in this project to design a user interface that allows to rotate a cube by means of using the `arcball` technique. Moreover, you will work with different attitude parametrizations and the relations between them to provide to the user useful data about the body orientation.

2 `Arcball` basics

The `arcball` method consists in the implementation of a *virtual trackball*. You could imagine a ball enclosing the 3D shape that is desired to be rotated (see Fig. 1). By clicking on the figure or its surroundings, and move the cursor the `arcball` method will simulate the rotation of the sphere about its center and as consequence the rotation of the body. By selecting two points over the sphere it is possible to calculate the quaternion that rotates the initial point to the final point. Therefore, we can use this fact to continuously generate rotations, apply them over the points that define our figure and simulate a change in the body's orientation.

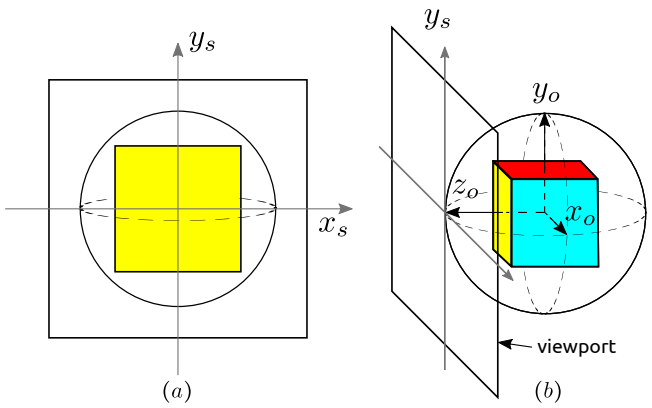

Figure 1: Cube embedded on a sphere. **(a)** Projection over the image plane. **(b)** 3D view of the image.

2.1 Map 2D points to 3D

2.1.1 Shoemake's arcball

By using the sphere equation, we can map points from the plane to the z-positive semi-sphere. If we consider that the origin of the image plane coincides with the origin of the sphere $x - y$ plane (see Fig. 2), any 2D point will have an image over the sphere with coordinates:

$$\boldsymbol{m}(x,y) = \begin{cases} \left(x,\, y,\, +\sqrt{r^2 - x^2 - y^2}\right)^\intercal & \text{if } \; x^2 + y^2 < r^2 \\ \frac{r}{\sqrt{x^2+y^2}}\,(x,\, y,\, 0)^\intercal & \text{if } \; x^2 + y^2 \geq r^2 \end{cases}$$

where r is the sphere radii.

This formulation is sufficient for extracting points over the sphere surface, however it presents abrupt changes in angle for mouse trajectories that conect external points of the sphere as shown in Fig. 3. You would not notice this effect in your implementation if you take points continuously, implying that two consecutive points will be near enough to rotate smoothly. Even though, the Shoemake's `arcball` presents the problem that the object rotates faster when moving near the boundary of the sphere and slowly when the pointer is near the origin. It is because the sphere shape. A solution is not just using the sphere, but combining it with an additional smooth surface. One example of this improvement is presented in the next section.

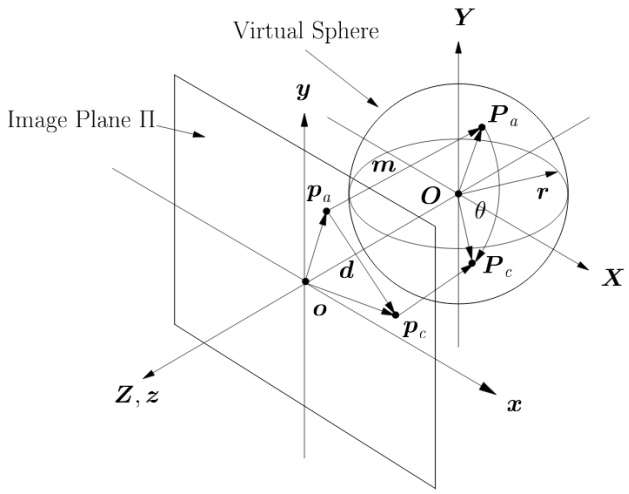

Figure 2: Extracted from (1). Relations between the sphere and viewport points.

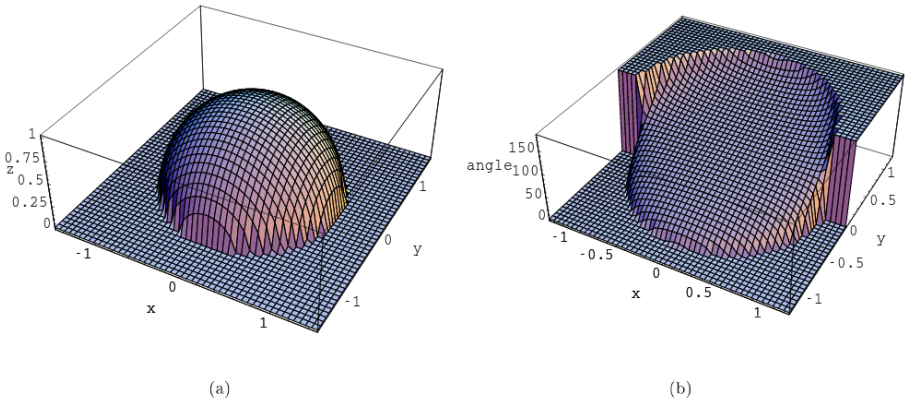

Figure 3: Extracted from (1). **(a)** z coordinate in function of pointer position and $r = 1$ using Shoemake's `arcball`. **(b)** angle between the axis of rotation and the z axis for $\boldsymbol{p_0} = (1.5, 0, 0)$ and $\boldsymbol{p_1} = \boldsymbol{m}(x, y)$ using Shoemake's `arcball`.

2.2 Holroyd's `arcball`

The Holroyd's `arcball`, fuse the sphere presented above with an hyperboloid (see Fig. 4).

The hyperboloid surface can be described by the next equation

$$z = \frac{r^2}{2\sqrt{x^2 + y^2}}$$

This function has a singularity as $z \to \infty$ for $x, y \to 0$, and extends to $z \to 0$

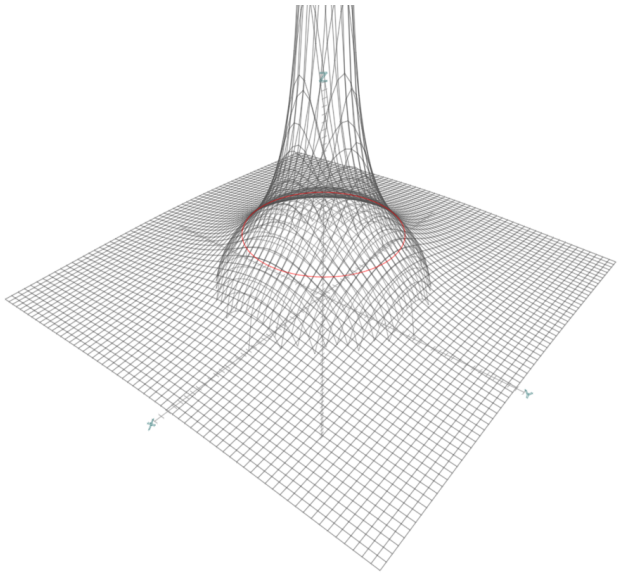

Figure 4: Extracted from (3). Sphere and hyperboloid, in red the circle $x^2+y^2 = r^2/2$.

as $x, y \to \infty$.

It is possible to combine both shapes equating the z coordinate:

$$\frac{r^2}{2\sqrt{x^2+y^2}} = \sqrt{r^2-x^2-y^2} \to x^2+y^2 = \frac{1}{2}r^2$$

Hence,

$$\sqrt{x^2+y^2} = z = \frac{\sqrt{2}}{2}r$$

the intersection of both functions is a circle at constant height. Since both curves are continuous it is possible to make a smooth transition. As consequence vectors on the 3D surface can be obtained from the 2D mouse coordinates by:

$$\boldsymbol{m}(x,y) = \begin{cases} \left(x,\, y,\, +\sqrt{r^2-x^2-y^2}\right)^\intercal & x^2+y^2 < \frac{1}{2}r^2 \\ \dfrac{r\left(x,\, y,\, \frac{r^2}{2\sqrt{x^2+y^2}}\right)^\intercal}{\left\|\left(x,\, y,\, \frac{r^2}{2\sqrt{x^2+y^2}}\right)\right\|} & x^2+y^2 \geq \frac{1}{2}r^2 \end{cases}$$

On Fig. 5, can be observed that the abrupt changes associated to the Shoemake implementation has been relaxed.

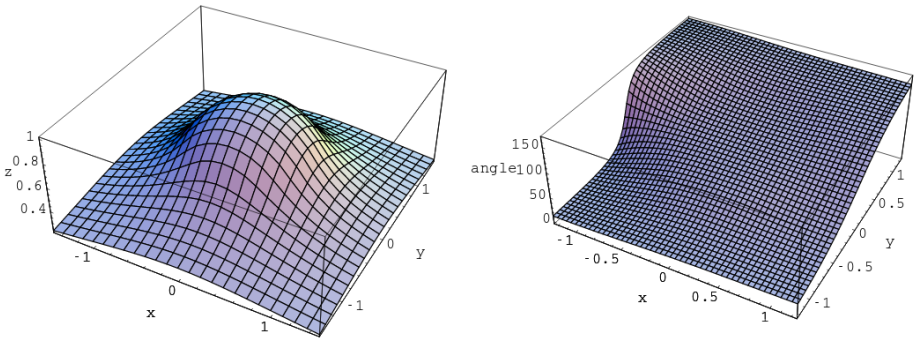

Figure 5: Extracted from (1). **(a)** z cordinate in function of pointer position and $r = 1$ using Holroyd's `arcball`. **(b)** angle between the axis of rotation and the z axis for $\boldsymbol{p_0} = (1.5, 0, 0)$ and $\boldsymbol{p_1} = \boldsymbol{m}(x, y)$ using Holroyd's `arcball`.

2.3 Quaternion from two vectors

After identifying two points over an sphere, what is the quaternion that transforms the first vector $\boldsymbol{m_0}$ into the second one $\boldsymbol{m_1}$?

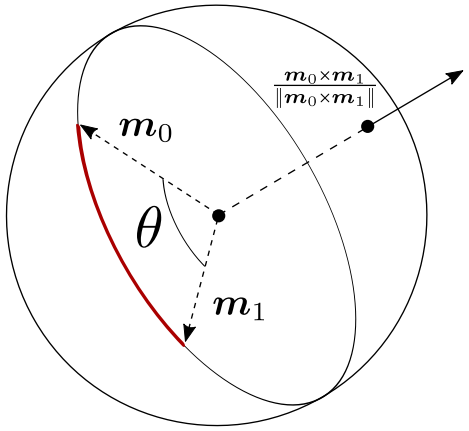

Figure 6: Rotation associated with $\boldsymbol{m_0}$ and $\boldsymbol{m_1}$.

Note that the necessary rotation will act in the perpendicular direction of $\boldsymbol{m_0}$ and $\boldsymbol{m_1}$ rotating the former an angle

$$\theta = \arccos\left(\frac{\boldsymbol{m_1}^\intercal \boldsymbol{m_0}}{\|\boldsymbol{m_1}\| \, \|\boldsymbol{m_0}\|}\right)$$

Therefore the quaternion that encodes the rotation can be calculated as

$$\mathring{q} = \left(\cos\left(\theta/2\right) , \sin\left(\theta/2\right) \frac{\boldsymbol{m_0} \times \boldsymbol{m_1}}{\|\boldsymbol{m_0} \times \boldsymbol{m_1}\|} \right)^{\mathsf{T}}$$

This way of achieving the quaternion is not too efficient, **read the article** in (2) to obtain and implement a more optimized formula.

2.4 Quaternion composition

Note that if points are captured continuously, for every new point at time instant k we can calculate a new quaternion $\delta\mathring{q}_k$. This quaternion represent the increment in the attitude given by the user, and therefore it must be composed with the previous attitude to obtain the actual quaternion

$$\mathring{q}_k = \delta\mathring{q}_k \mathring{q}_{k-1}$$

3 Arcball execution diagram

A possible diagram (not unique) of the execution of the `arcball` implementation is:

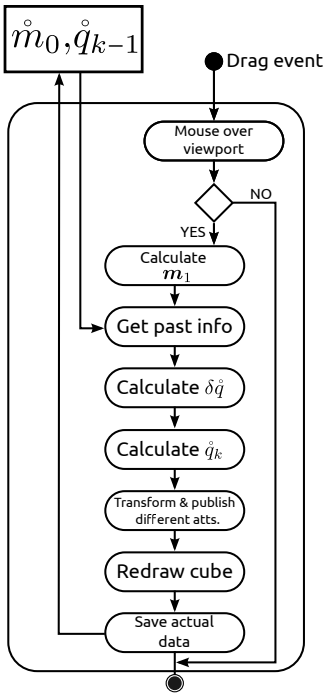

Figure 7: Execution diagram for the mouse drag callback.

4 Project definition

You will implement a UI, based on the template that you can find on the virtual campus with the next funcionalities:

- The cube that appears on the left has to respond to the user input following the presented method of the Holroyd's `arcball` and rotate the cube.
- 5 panels has to be shown on the right. Those panels will contain information of the equivalent attitude representation parametrized as
  - Quaternions
  - Euler principal Angle and Axis
  - Euler angles
  - Rotation vector
  - Rotation matrix

  The information must to be updated on run time while dragging the cube.
- Every panel except for the rotation matrix has to be editable and must contain a push button. The pushbutton must update the information in the other parametrizations accordingly and must rotate the figure to represent the attitude entered by the user.
- A general reset pushbutton has to be implemented. When pressed, the cube must transform to its original position and all the attitude parametrization must take the corresponding value of zero rotation as well.

In addition, proper paper documentation of the implemented code has to be delivered including at least:

- A list containing the description of **all** the implemented functions, containing information of the function purpose, inputs and outputs.
- A diagram relating every possible user action with the functions invoked and order of execution.

Finally a video-presentation has to be delivered where you explain the past two points and show the functionalities and the correct operation of the implemented tool.

References

[1] Henriksen, K., Sporring, J., and Hornbæk, K. Virtual trackballs revisited. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 206–216.

[2] Lol Engine, a community project. Beautiful maths simplification: quaternion from two vectors.

[3] OpenGL. Object mouse trackball.

[4] Shoemake, K. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface* (1992), vol. 92, pp. 151–156.