Test task – Cuboid-sphere

**NB!** Before starting the test, send a message to [martin.karu@lightcodephotonics.com](mailto:martin.karu@lightcodephotonics.com). The content of the message: estimate the complexity of the task (e.g. in hours) and set yourself a deadline.

## Included files

* cuboid-sphere.hdf5 - 3D depth map and intensity map of the scene
  + File contains:
    - Depth map (z-map in camera coordinates – see additional information)
    - Intensity map
    - Metadata:
      * Horizontal field-of-view in degrees
      * Vertical field-of-view in degrees
  + Scene contains:
    - 1x Sphere
    - 1x Cuboid
* cuboid-sphere.png - 2D visualization of the scene

## Task description

### Task 1

1. Read in the HDF5 file.
2. Calculate the point cloud.
3. For each object in the scene, please calculate:
   1. the surface area,
   2. the volume,
   3. the centroid (in global coordinates).
4. Make the calculations available as callable functions.
5. Save the point cloud as an HDF5 file.

### Task 2

1. Read in the PNG file
2. Programmatically visualize the edges on the main face of the objects
   1. Rectangle for the boxes
   2. Circle for the sphere
3. Calculate the area of the main face

### Task 3

We wish to build a Computer Vision application that can detect objects in a 3D scene. In the first iteration, we will only detect cuboids and spheres.

The 3D camera is connected to a PC and we’re able to read the 3D data programmatically as a point cloud.

You are included as a Data Science expert in the solution design.

1. Describe the main logical components required to build and deploy the solution to production. E.g. “Inference as a Flask API”
2. List up to 5 main problems that may arise during the deployment of the system.

## Technical implementation

The solution needs to:

* Be pushed to a git repository
* Be containerized (Docker)
* Have a README file describing how to execute the solution

## Additional information

The coordinate system used in the task is shown in the figure above. The 3D camera is located at point O, which has the global coordinates (X=0, Y=0, Z=0). The camera optical axis is oriented towards a point with global coordinates (X=0.5, Y=0.5, Z=1).

## 

*Figure 1: camera viewing position*