Data for the Pattern Recognition paper, including the structural and topological analysis of the images used in this research, as well as the algorithm employed to extract various shape descriptors..

César Eduardo Muñoz-Chávez Hermilo Sánchez-Cruz
Elisa Isabel Gómez-Gómez

September 11, 2024

1 Appendix C. Structural and topological analysis of the images.

This document presents the algorithm employed to extract key shape descriptors, specifically the enclosing surface and volume, as detailed in Algorithm 1. Additionally, Tables 1 and 2 outline the comprehensive structural and topological analysis conducted on each of the 66 3D voxelized objects examined in this study. The analysis includes descriptors such as object dimensions, Euler characteristics, and the number of cavities and tunnels. These descriptors are essential for understanding the spatial complexity and topology of the objects, which play a critical role in accurately modeling and analyzing the 3D structures.

This analysis highlights a critical issue of multicollinearity among the predictor variables, which occurs when independent variables are highly correlated. Multicollinearity complicates the ability of traditional regression models to estimate the individual effects of each predictor, resulting in unstable and unreliable coefficient estimates. To illustrate this, Figure 1 show this problem.

Table 1: Objects for testing and its and dimensions (Part 1).

Name	Dimension	Euler	Cavities	Tunnels
Object1	6×5×5	2	0	0
Object2	7×10×9	-4	0	5
Cup	29×25×41	0	0	1
Dragon	42×64×90	0	0	1
Sphere0	65×65×31	1	0	0
Sphere3	65×65×31	3	2	0
Sphere4	65×65×31	3	2	0
Sphere5	65×65×31	1	2	2
Torus	67×65×10	0	0	1
Torush	67×65×10	-1	0	2
Vase	69×53×41	0	0	1
Cheese	102×102×102	-2	1	4
Squirrel	128×128×128	1	0	0
Armchair	128×128×128	-1	0	2
Axe	128×128×128	0	0	1
Batmobile	128×128×128	5	57	130
Bird	128×128×128	-1	0	3
Owl	128×128×128	0	0	1
Dragon	128×128×128	-8	0	9
Elephant	128×128×128	1	0	0
Speakers	128×128×128	-5	0	7
French Horn	128×128×128	-88	0	104
GUN	128×128×128	-7	0	8
Moon	128×128×128	1	0	0
Bear	128×128×128	1	0	0
Sheep	128×128×128	1	0	0
Penguin	128×128×128	1	0	0
Pig	128×128×128	1	0	0
Turtle	128×128×128	1	0	0
Table Lamp	128×128×128	1	0	0
Bolt	256×256×256	1	0	0
Chair	256×256×256	1	0	0
Chameleon	256×256×256	3	0	0
House	256×256×256	-256	0	2258

Table 2: Objects for testing and its and dimensions (Part 2).

Name	**Dimension**	**Euler**	**Cavities**	**Tunnels**
Ship	256×256×256	-8	0	22
Skateboard	256×256×256	0	0	1
T34 Soviet Tank	256×256×256	-143	0	149
Axolotl	256×256×256	-1	0	2
Plane	256×256×256	1	0	0
Car	256×256×256	1	0	0
Creeper	256×256×256	1	0	0
Giraffe	256×256×256	1	0	0
Karambit	256×256×256	0	0	1
Wolf	256×256×256	1	0	0
Nintendo	256×256×256	760	0	1
Duck	256×256×256	1	0	0
Snake	256×256×256	0	0	1
Chandelier	520×520×520	-349	0	419
Dinosaur	520×520×520	1	0	0
Fish	520×520×520	-12	0	14
Kangaroo Chair	520×520×520	-2082	0	2512
Lioness	520×520×520	16	0	1
Red Hand	520×520×520	-23	0	46
Sofa	520×520×520	-9	0	10
Teleosaurus	520×520×520	0	0	1
Tulip Chair	520×520×520	1	0	0
Yak	520×520×520	-1	0	2
Batman	512×512×512	1	0	0
Knife	512×512×512	1	0	0
Staircase	512×512×512	1	0	0
Sphere	512×512×512	1	0	0
Sword	512×512×512	3	0	0
L	512×512×512	1	0	0
Brick	512×512×512	1	0	0
Peak	512×512×512	-3	0	4
Pyramid	512×512×512	1	0	0

Algorithm 1 Calculate Volume and Enclosing Surface of 1-Voxels Regions in a 3D Binary Image

Require: 3D binary image Image where 1 represents the region of interest

Ensure: Total volume and enclosing surface of the 1-pixel regions in the 3D image

```
Initialize Count_ones ← 0
Initialize Enclosing_surface ← 0
if Image.ndim ≠ 3 then
    raise ValueError("Input should be a 3D binary image.")
end if
for i ← 0 to Image.shape[0] − 1 do
    for j ← 0 to Image.shape[1] − 1 do
        for k ← 0 to Image.shape[2] − 1 do
            if Image[i, j, k] = 1 then
                Count_ones ← Count_ones + 1
                Initialize neighbors ← 0 {Initialize the neighbor count}
                if i = 0 or Image[i − 1, j, k] = 0 then
                    neighbors ← neighbors + 1
                end if
                if i = Image.shape[0] − 1 or Image[i + 1, j, k] = 0 then
                    neighbors ← neighbors + 1
                end if
                if j = 0 or Image[i, j − 1, k] = 0 then
                    neighbors ← neighbors + 1
                end if
                if j = Image.shape[1] − 1 or Image[i, j + 1, k] = 0 then
                    neighbors ← neighbors + 1
                end if
                if k = 0 or Image[i, j, k − 1] = 0 then
                    neighbors ← neighbors + 1
                end if
                if k = Image.shape[2] − 1 or Image[i, j, k + 1] = 0 then
                    neighbors ← neighbors + 1
                end if
                Enclosing_surface ← Enclosing_surface + neighbors {Add to total
                surface area}
            end if
        end for
    end for
end for
return Count_ones, Enclosing_surface
```

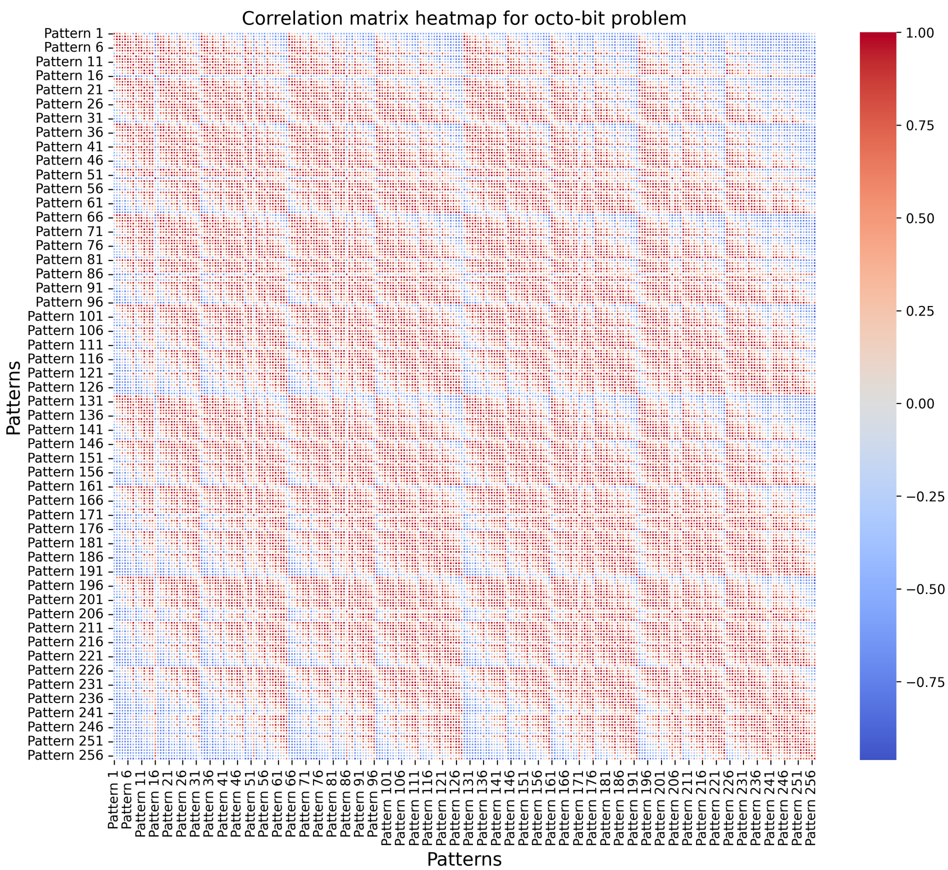

Figure 1: Heatmap of the octo-bits multicollinearity problem