Annex. Use of macros in POV-Ray codes*

Figure S1. Icosidodecahedron geometrical features.
Figure S2. Rhombicuboctahedron calculation of the k value from cube.
Figure S3. Rhombicuboctahedron deduction of k value from octahedron.

The use of macros is helpful in POV-Ray codes. A macro is declared by using an identifier, and a list of parameters. Its syntax is as follows:

```
Box 1. Commands in Pov-Ray Language to declare a Macro
#macro Identifier (parameters)
Tokens
#end
```

Macros need to be declared before they can be used. The manner to invoke them is as follows:

Macro_identifier (parameters list)

The next example is provided to facilitate the understanding of macros.

```
#macro Enlaces (first, final, Ve1, distan, kolor)
    // printing bonds as cylinders
    #declare i=first;
    #while (i< final-1)
     #declare j=i+1;
       #while (j< final)
        #declare L1= VDist( Ve1[i], Ve1[j]);
          #if(L1< a* (distan) +0.01)
            cylinder{ Ve1[i], Ve1[j] 0.05 pigment{color kolor} }
          #end
        #declare j=j+1;
        #end
      #declare i=i+1;
      #end
```

It can be used to calculate the edges of a solid as cuboctahedron.

* Annex related to macros, deduction of magnitude of the translation vectors of rhombicuboctahedron and truncated icosidodecahedron.

```
POV-Ray Code to make a cuboctahedron
// Insert here the last definition of libraries,
// light_source, camera, and background

#declare acube=2/sqrt(2);     //cube edges length
#declare Pos= array[8];        // cube positions
#declare cuboc= array [12]; // cuboctahedron vertices

// Cube positions
#declare Pos[0]= <acube/2,    acube/2,  acube/2>;
#declare Pos[1]= <-acube/2,  -acube/2, -acube/2>;
#declare Pos[2]= <-acube/2,   acube/2,  acube/2>;
#declare Pos[3]= <acube/2,   -acube/2,  acube/2>;
#declare Pos[4]= <acube/2,    acube/2,  -acube/2>;
#declare Pos[5]= <-acube/2,  -acube/2,  acube/2>;
#declare Pos[6]= <-acube/2,   acube/2,  -acube/2>;
#declare Pos[7]= <acube/2,   -acube/2,  -acube/2>;

#fopen cuboct "cuboctahedron.xyz" write

// This block is to calculate the distances among vertices
// and to define edges
//Calculating the center of cube edges
    #declare i=0;
    #declare ncub=0;
    #declare n=8;
    #while (i<n-1)
     #declare j=i+1;
       #while (j<n)
        #declare L= VDist(Pos[i],Pos[j]);
          #if(L< acube+0.1)
           #declare cuboc[ncub]=Pos[i]+0.5*(Pos[j]- Pos[i]);
            sphere{cuboc[ncub], 0.25 pigment{color Red}}
#write (cuboct,"Au", " ",vstr(3, cuboc[ncub]," ",3,5),"\n")
            #declare ncub=ncub+1;
          #end
        #declare j=j+1;
        #end
      #declare i=i+1;
     #end
```

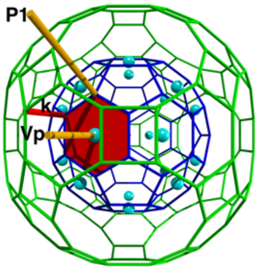
To show the edges, the macro is invoked as follows.
Enlaces (0, 11, cuboc, 1, Green).

The truncated icosidodecahedron

To select the hexagonal faces of the truncated icosahedron, it was necessary to calculate the dot product among the position vectors of the dodecahedron (cyan spheres in figure S1) and the position vectors of vertices of the truncated icosahedron (orange vectors).

With respect to the magnitude of the translation vector (k) it was necessary to subtract the relation of the distance from the origin to the center of hexagonal faces of truncated icosahedron and trun-cated icosidodecahedron. That relationship was found in the reference 22 (Weisstein, 2020).

Figure S1. Truncated icosidodecahedron from a truncated icosahedron.



Note: Both Archimedean solids have hexagonal faces, in such manner that the translation of the hexagonal faces of the truncated icosahedron produces the truncated icosidodecahedron.

Rhombicuboctahedron calculation of the magnitude (k) of the perpendicular vector

To obtain the magnitude of the perpendicular vector (k) to the cube faces, it was necessary to make a code to translate the cube vertices by a fraction. In the following box is included the used code.

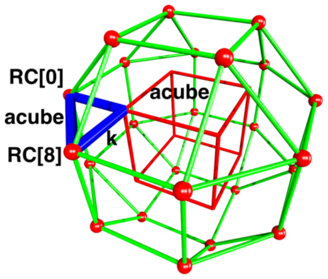
```
// Calculation of the magnitude (k) of the translation vector being perpen-
dicular to square faces of the cube

   #fopen RombiCuH “T_Rombicuboctahedron.txt” write
    #declare h=0.0001;     // INCREMENT
    #declare contadd=0;

#while(vlength(RC[8 ] - RC[0 ])< a)
        #local RC[0 ]=      RC[0 ]+    h * <1,0,0>;
        #local RC[8 ]=      RC[8]+     h * <0,0,1>;

// Checking the distance among 2 translated vertices
#write (RombiCuH, RC[8], RC[0], “distance= “,vlength(RC[8 ]- RC[0 ]),”  T=”,
contadd*h, “\n”)
        #declare T= contadd*h ;     // This is k
        #declare contadd= contadd+1;
      #end
```

Figure S2. Illustration of the calculation of the magnitude (k) of the displacement vector applied to cube vertices to obtain the rhombicuboctahedron. RC[0] and RC[8] are obtained by translating one vertex of the cube along perpendicular vectors to square faces.



In figure S2, the blue triangle can be used to analytically obtain the magnitude of the translation vector (k).

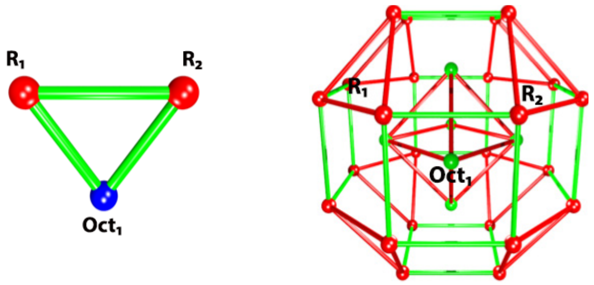
$$k^2 + k^2 = acube^2$$

Then

$$\mathrm{k} = \frac{acube}{\sqrt{2}}$$

Rhombicuboctahedron deduction of the magnitude (k) of the perpendicular vector

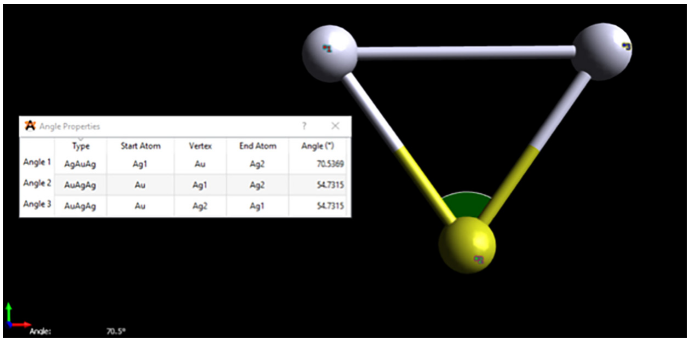
Figure S3. Illustration of the calculation of the magnitude of the displacement vector applied to octahedron vertices to obtain the rhombicuboctahedron.



As mention in the main manuscript, the Rhombicuboctahedron can be obtain from both the cube and its dual, the octahedron. In each case we have to translate the faces a certain distance (k). In the case of the octahedron, this length is *b*0.8660*, being b the edge of the octahedron. In the following, we explore how did we deduce this.

In the figure S3 the blue dot represents one of the octahedron's vertices. As we can see, 2 vectors come out from the octahedron's vertex (there are 4 vertices of the Rhombicuboctahedron for each octahedron's vertex but for simplicity issues we just consider two). These two vectors are the "translation vectors" from the octahedron's vertex to its rhombicuboctahedron's vertices. So, the distance between the two red dots (representing Rhombicuboctahedron's vertices) must be the same as the edge of the octahedron

At first we give these vectors a length of 1; which is the default value given by Pov-Ray with the function *VPerp_To_Plane* (*V1*, *V2*). This arbitrary value is just given because we aren't interest-ed in the length of the vectors yet, but in the angle between them.


Angle Properties

	Type	Start Atom	Vertex	End Atom	Angle (°)
Angle 1	AgAuAg	Ag1	Au	Ag2	70.5369
Angle 2	AuAgAg	Au	Ag1	Ag2	54.7315
Angle 3	AuAgAg	Au	Ag2	Ag1	54.7315

Angle: 70.5°

The program shows that the angle between the vectors is 70.5369° and the other angles are 54.7315°@ each. Given this information, we can use the sine rule in order to calculate the length of the vectors.

$$\frac{b}{\sin(1.2311\ radians)} = \frac{k}{(0.95245)}$$

Being b the edge of the octahedron and rhombicuboctahedronrhombicuboctahedron, and k the length of the translation vector.

$$k = \frac{b\sin(0.95245)}{\sin(1.2311\ radians)}$$

$k = b(0.8659)$

Coordinates of the rhombicuboctahedron obtain by the translation of octahedron's faces:

1	−0.70692	0.70692	1.70692	13	−1.70692	0.70692	0.70692
2	0.70692	0.70692	1.70692	14	−1.70692	0.70692	−0.70692
3	−0.70692	−0.70692	1.70692	15	−1.70692	−0.70692	0.70692
4	0.70692	−0.70692	1.70692	16	−1.70692	−0.70692	−0.70692
5	0.70692	1.70692	0.70692	17	−0.70692	−1.70692	0.70692
6	0.70692	1.70692	−0.70692	18	0.70692	−1.70692	0.70692
7	−0.70692	1.70692	0.70692	19	−0.70692	−1.70692	−0.70692
8	−0.70692	1.70692	−0.70692	20	0.70692	−1.70692	−0.70692
9	1.70692	−0.70692	0.70692	21	0.70692	0.70692	−1.70692
10	1.70692	0.70692	0.70692	22	0.70692	−0.70692	−1.70692
11	1.70692	−0.70692	−0.70692	23	−0.70692	0.70692	−1.70692
12	1.70692	0.70692	−0.70692	24	−0.70692	−0.70692	−1.70692