Zonohedra and Zonotopes

David Eppstein*

Dept. of Information & Computer Science
U.C. Irvine, CA, 92717
http://www.ics.uci.edu/~eppstein/

Tech. Report 95-53

December 15, 1995

Abstract: We use *Mathematica* to construct zonotopes and display zonohedra.

*Work supported in part by NSF grant CCR-9258355 and by matching funds from Xerox corp.

Introduction

A *zonotope* is a set of points in d-dimensional space constructed by the sum of scaled vectors a[[i]] v[[i]] where a[[i]] is a scalar between 0 and 1 and v[[i]] is a d-dimensional vector. Alternately it can be viewed as a *Minkowski sum* of line segments connecting the origin to the endpoint of each vector. It is called a zonotope because the faces parallel to each vector form a so-called *zone* wrapping around the polytope. A *zonohedron* is just a three-dimensional zonotope. This notebook contains code for constructing zonotopes and displaying zonohedra.

There is some confusion in the definition of zonotopes; Wells [W91] requires the generating vectors to be in *general position* (all d-tuples of vectors must span the whole space), so that all the faces of the zonotope are parallelotopes. Others [BEG95,Z95] do not make this restriction. Coxeter [C73] starts with one definition but soon switches to the other. We use the unrestricted definition here.

The combinatorics of the faces of a zonotope are equivalent to those of an arrangement of hyperplanes in a space of one fewer dimension, so for instance zonohedra correspond to planar line arrangements. This can be most easily seen by considering the space of d-dimensional hyperplanes tangent to the zonotope. The space of all d-dimensional unit vectors can be seen as a unit sphere, equivalent to oriented projective (d-1)-space. For any given unit vector, there is a unique hyperplane normal to the vector and tangent to the zonotope at some kind of face. One can swing a hyperplane tangent to a k-dimensional face through a collection of angles with (d-1-k) degrees of freedom; this corresponds to a cell with dimension (d-1-k) in this projective space. Thus the faces of the zonotope correspond to a dual cell decomposition of space (dual because the dimensions are reversed — high dimensional faces correspond to low dimensional cells). If there were only a single vector, this decomposition would be given by a single hyperplane (partitioning the tangents into those that touch the origin, those that touch the endpoint of the vector, and those parallel to the vector that touch both points). But the decomposition corresponding to a Minkowski sum is formed by overlaying the decompositions corresponding to the two summands, so the cell decomposition of the tangent space to a zonotope is exactly a hyperplane arrangement. The zone of faces parallel to a given vector corresponds exactly to a hyperplane in the arrangement.

Because of this correspondence, we can construct zonotopes in O(n^(d-1)) time. In particular it would take O(n^2) time to construct a zonohedron from an initial set of n vectors. However our implementation uses a slower algorithm better suited to the functional nature of *Mathematica* programming. We first find the subsets of vectors corresponding to the faces of the zonotope. These subsets are determined incrementally, by combining each new vector with previously generated subsets, and then forming additional faces pairing the new vector with any remaining

old vectors. Once we have generated all faces, we determine coordinates for their vertices (recursively, since each face is again a zonotope in one lower dimension) and lift them into position by adding appropriate sets of vectors.

Implementation

Manipulation of Individual Faces

In the face generation stage, we represent a face by the indices of the vectors generating it. When we add a new vector to the zonohedron, we test it against each face and determine whether it is coplanar (by computing a dxd determinant); if so it gets added to the list of indices. In the face placement stage, we instead represent a face by the coordinates of its vertices, so we need code to translate between the two representations. Once we have found the coordinates of a face recursively as a lower-dimensional zonotope, we lift two copies of it in place by using a similar determinant computation to determine which vectors to add to each copy.

In order to compute recursive faces we need a way of testing signs of determinants when there are fewer than d vectors involved. We simply add extra vectors on the *moment curve* (x,x^2,x^3,...), so that they are independent of each other and (hopefully) of the inputs.

In[1]:=

```
ZSignTest[vv_,f_] :=
    Det [ Part[vv,f] ~Join~
          Table[(i+200)^j,
                {i,Length[vv[[1]]]-Length[f]},
                {j,Length[vv[[1]]]}]]
```

This routine tests whether to add a new vector to an existing face.

In[2]:=

```
ZAddToFace[vv_,f_,i_,d_] :=
    If[ Length[f] < (d-1) ||
        ZSignTest[vv, Append[Take[f,d-1],i]] == 0,
        Append[f,i], f ]
```

The next two routines convert faces from subsets of indices to coordinates. This is simply a recursive call to Zonotope, except that when the face is one-dimensional the result is just a line segment and we stop the recursion to compute it more simply.

In[3]:=

```
ZOrigin[vv_] := Table[ 0, {Length[ vv[[1]] ]} ]
```

In[4]:=

```
ZFace[vv_,f_,d_] :=
    If [d == 2,
        {ZOrigin[vv], Plus @@ Part[vv,f]},
        Zonotope[ Part[vv,f], d-1 ]]
```

The remaining routines in this section are concerned with translating the converted faces into their appropriate positions in space. Each face appears in two copies, and each vector not contributing to the face is used to shift exactly one of the copies. Which one is determined by another determinant calculation.

In[5]:=

```
ZLiftVector[vv_,f_,i_,d_] :=
    If [ MemberQ[f, i], {ZOrigin[vv],ZOrigin[vv]},
        If [ ZSignTest[vv, Append[Take[f,d-1],i]] < 0,
            {vv[[i]], ZOrigin[vv]},
            {ZOrigin[vv], vv[[i]]}]]
```

I want to add the same vector to all level-one lists in a nested list structure, but *Mathematica* doesn't let Plus work this way (although it does work with scalars...)

In[6]:=

```
ZVecPlus[a_,v_] :=
    If[Head[First[a]] === List,
        ZVecPlus[#,v]& /@ a, a+v]
```

In[7]:=

```
ZLiftFace[vv_,f_,d_] :=
    {#1 ~ZVecPlus~ #2[[1]], #1 ~ZVecPlus~ #2[[2]]}& [
        ZFace[vv,f,d],
        Plus @@ Table[ZLiftVector[vv,f,j,d],{j,Length[vv]}]]
```

■ Face Generation Stage

As described earlier, we generate faces (as subsets of vectors) incrementally, adding one vector at a time to our zonotope. Adding one vector has two parts: including it in the faces with which it is coplanar, and then making new faces with which it is noncoplanar.

In[8]:=

```
ZOldFaces[vv_,ff_,i_,d_] :=
    (ZAddToFace[vv, #, i, d])& /@ ff
```

To generate the new faces, we make all d-tuples of indices involving index i, and then filter out the ones that are subsets of the indices in existing faces.

In[9]:=

```
ZTuples[i_,d_] :=
    If[d == 1, {{i}},
       (Append[#,i]&) /@ Join @@
           Table[ZTuples[j,d-1],{j,i-1}]]
```

In[10]:=

```
ZSubsetQ[s_,t_] := (Length[t] == Length[Intersection[s,t]])
```

In[11]:=

```
ZFilterTuple[ff_,t_] :=
    Not[Or @@ (ZSubsetQ[#,t]&) /@ ff]
```

In[12]:=

```
ZAddTuples[vv_,ff_,i_,d_] :=
    ff ~Join~ Select[ZTuples[i,d-1], ZFilterTuple[ff,#]&]
```

In[13]:=

```
ZNewFaces[vv_,ff_,i_,d_] :=
    ZAddTuples[vv, ZOldFaces[vv,ff,i,d], i, d]
```

In[14]:=

```
ZAllFaces[vv_,d_] :=
    Fold[ZNewFaces[vv,#1,#2,d]&,
        {{}}, Table[i,{i,Length[vv]}]]
```

■ Face Placement Stage

In[15]:=

```
ZLiftAll[vv_,ff_,d_] :=
    Join @@ Map[ZLiftFace[vv,#,d]&,ff]
```

■ Main Zonohedron Code

In[16]:=

```
Zonotope[vv_,d_] :=
    ZLiftAll[vv,ZAllFaces[vv,d],d]
```

The general zonotope code will produce a recursive description in which 2d faces are lists of 1d edges, etc. For nice drawings of zonohedra we need to convert these lists of edges into a single polygon.

In[17]:=

```
ZNext[f_,e_] :=
    Join @@ Map[
        If[ e[[2]] == #[[1]] && e[[1]] != #[[2]], #,
            If[ e[[2]] == #[[2]] && e[[1]] != #[[1]],
                Reverse[#], {}]]&, f]
```

In[18]:=

```
ZMakePolygon[f_] :=
    First /@ NestList[ ZNext[f,#]&, f[[1]], Length[f]-1 ]
```

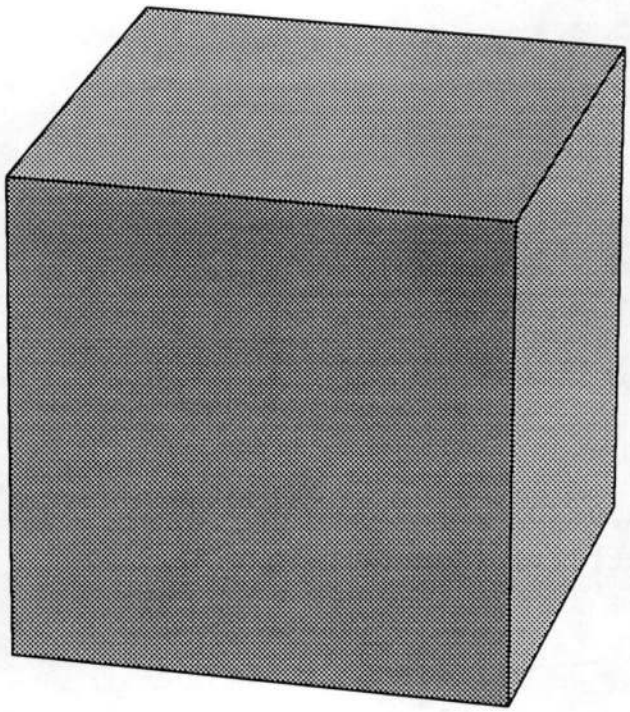
In[19]:=

```
Zonohedron[vv_] :=
    Show[Graphics3D[Polygon /@ ZMakePolygon /@
                        Zonotope[N[vv],3]],
         ViewPoint->{12,3,5},
         Boxed -> False]
```

Examples

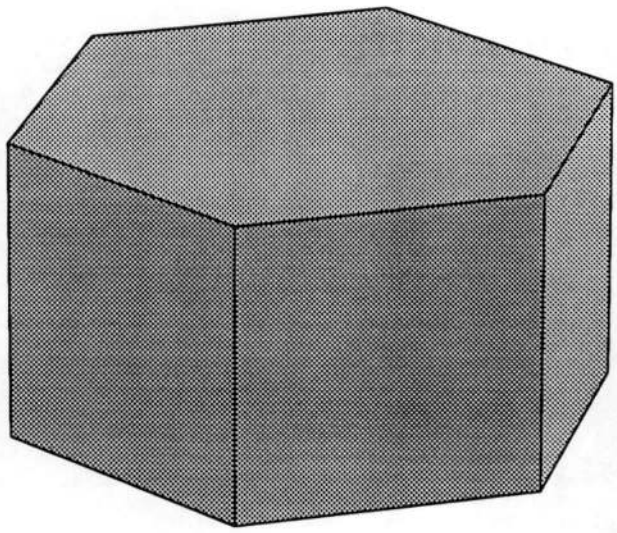
The Cube

```
Zonohedron[{{1,0,0},{0,1,0},{0,0,1}}]
```

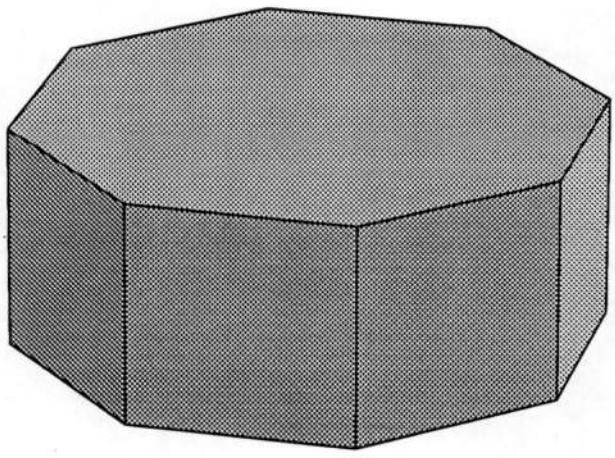


■ Prisms

```
Zonohedron[{{1,Sqrt[3],0},{1,-Sqrt[3],0},{2,0,0},{0,0,2}}]
```



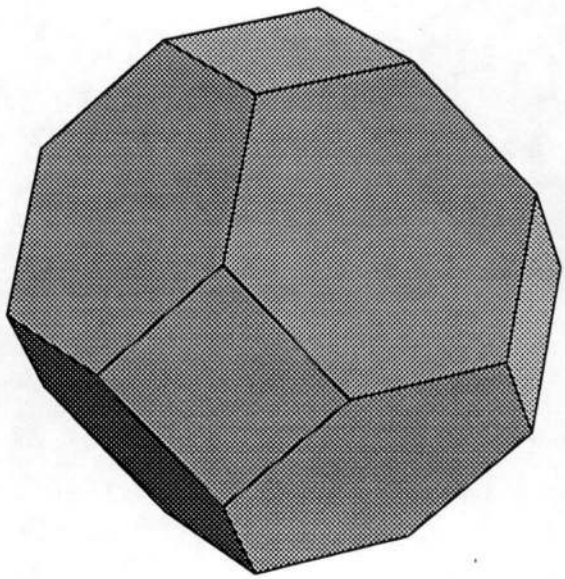
```
Zonohedron[{{2,0,0},{0,2,0},{0,0,2},
            {Sqrt[2],Sqrt[2],0},{Sqrt[2],-Sqrt[2],0}}]
```



The Truncated Octahedron

The octahedron itself is not a zonohedron, since its faces are triangular and do not form zones. However if one uses the edges of the octahedron as generators, one gets this zonohedron, in which the triangular faces of the octahedron have been truncated to hexagons and squares added to connect them. The twelve octahedron edges come in six pairs, so there are six generators. This shape can fill space without leaving any gaps.

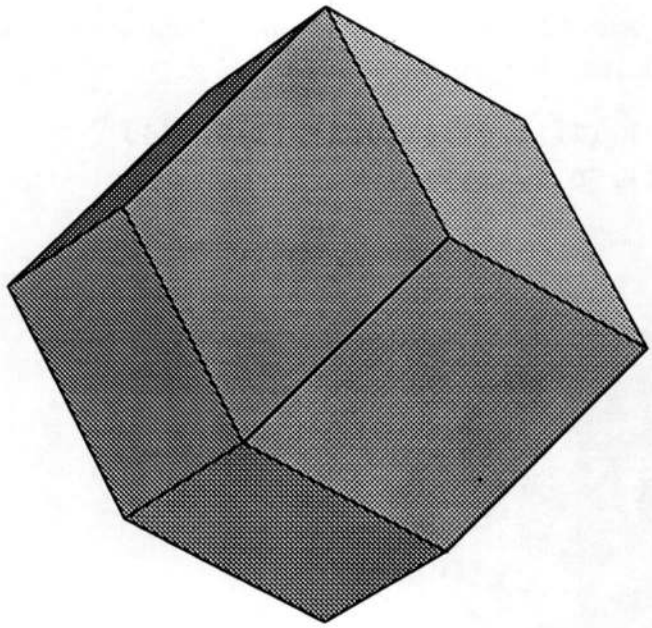
```
Zonohedron[{{1,1,0},{1,-1,0},{1,0,1},{1,0,-1},
           {0,1,1},{0,1,-1}}]
```



The Rhombic Dodecahedron

This is the dual to the cuboctahedron, an Archimedean solid formed by combining the six squares of a cube with the eight triangles of an octahedron. The cuboctahedron itself is not a zonohedron because of its triangular faces.

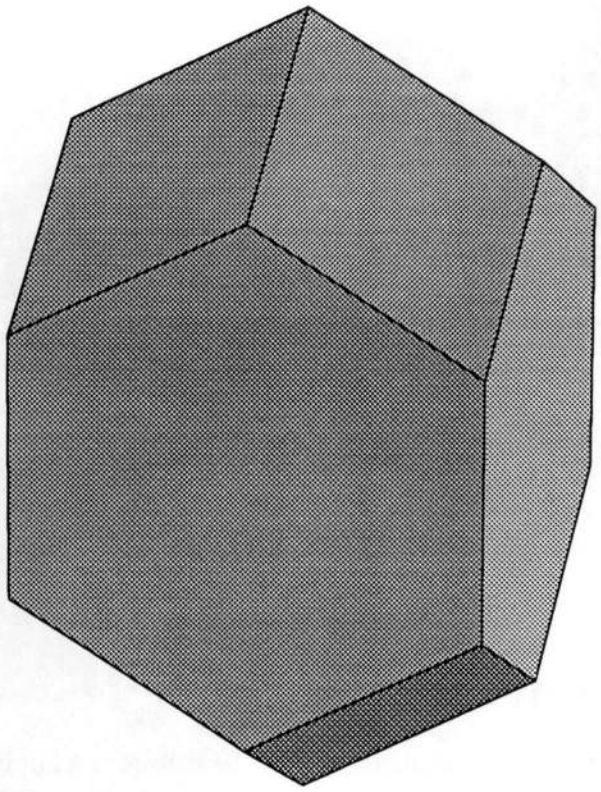
```
Zonohedron[{{1,1,1},{1,-1,1},{1,1,-1},{1,-1,-1}}]
```



■ The Extended Rhombic Dodecahedron

This zonotope is noteworthy for (like the cube, hexagonal prism, truncated octahedron, and rhombic dodecahedron) being able to fill space without leaving any gaps. The one here is drawn with different angles from the rhombic dodecahedron itself, to make the hexagonal sides regular.

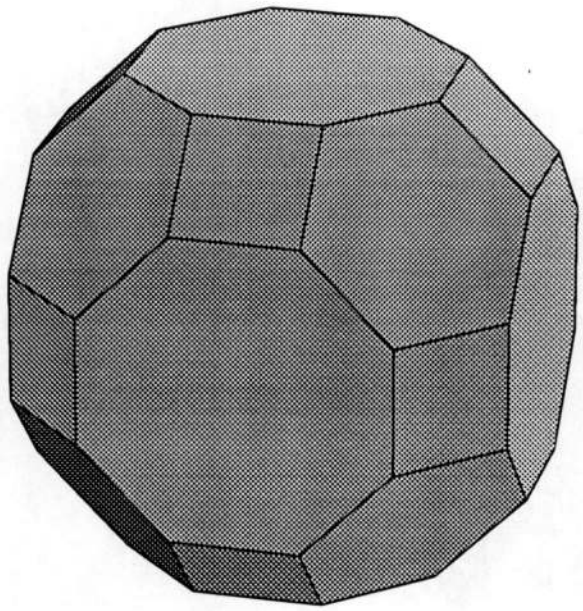
```
Zonohedron[{{0,Sqrt[3],1},{Sqrt[3],0,1},
            {0,-Sqrt[3],1},{-Sqrt[3],0,1},
            {0,0,2}}]
```



■ The Truncated Cuboctahedron

As can be seen from its set of generators, this is the Minkowski sum of a cube and a truncated octahedron. It is also known as the Great Rhombicuboctahedron.

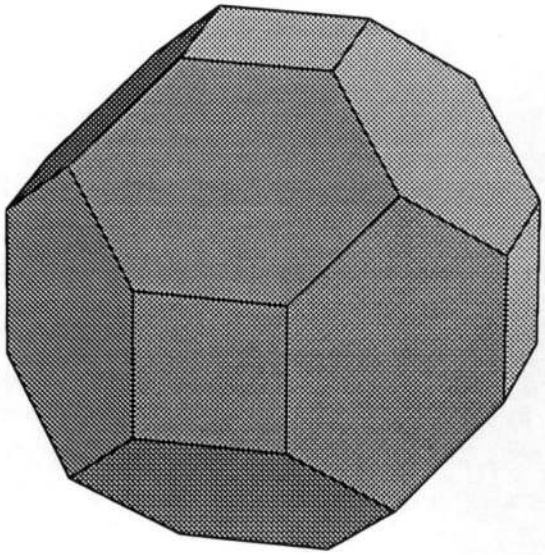
```
Zonohedron[{{1,1,0},{1,-1,0},{1,0,1},{1,0,-1},
            {0,1,1},{0,1,-1},
            {Sqrt[2],0,0},{0,Sqrt[2],0},{0,0,Sqrt[2]}}]
```



■ The Truncated Rhombic Dodecahedron

This is the Minkowski sum of a cube and a rhombic dodecahedron. The hexagons can not be regular, although they do have all sides the same length, since three regular hexagons would meet in a flat solid angle instead of the corners here. Although this shape does not fill space on its own, it can be combined with cubes in a regular space-filling pattern.

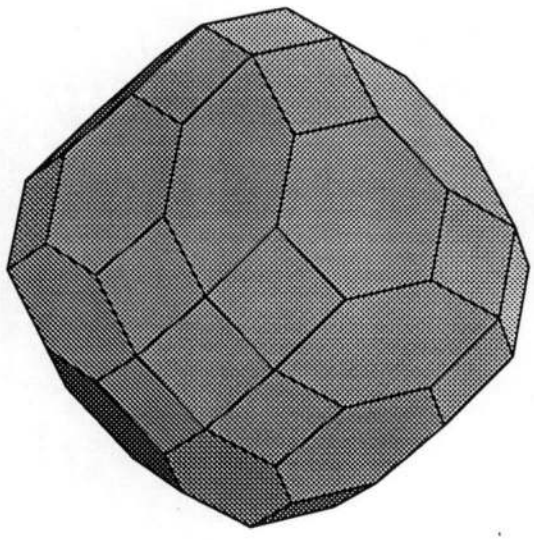
```
Zonohedron[{{1,1,1},{1,1,-1},{1,-1,1},{1,-1,-1},
            {Sqrt[3],0,0},{0,Sqrt[3],0},{0,0,Sqrt[3]}}]
```



■ Two non-Archimedean zonohedra

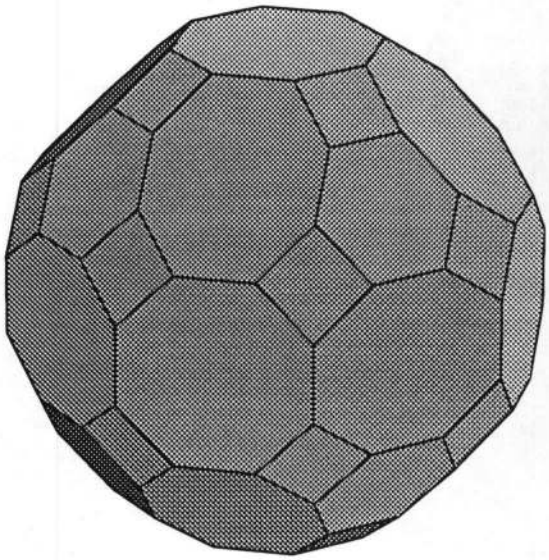
This is the Minkowski sum of a a truncated octahedron and a rhombic dodecahedron. Its faces are squares, regular hexagons, and long irregular hexagons.

```
Zonohedron[{{1,1,0},{1,-1,0},{1,0,1},{1,0,-1},
            {0,1,1},{0,1,-1},
            {Sqrt[2/3],Sqrt[2/3],Sqrt[2/3]},
            {Sqrt[2/3],Sqrt[2/3],-Sqrt[2/3]},
            {Sqrt[2/3],-Sqrt[2/3],Sqrt[2/3]},
            {Sqrt[2/3],-Sqrt[2/3],-Sqrt[2/3]}}]
```



This is the Minkowski sum of a cube, a truncated octahedron, and a rhombic dodecahedron. Some of the octagonal faces are non-regular.

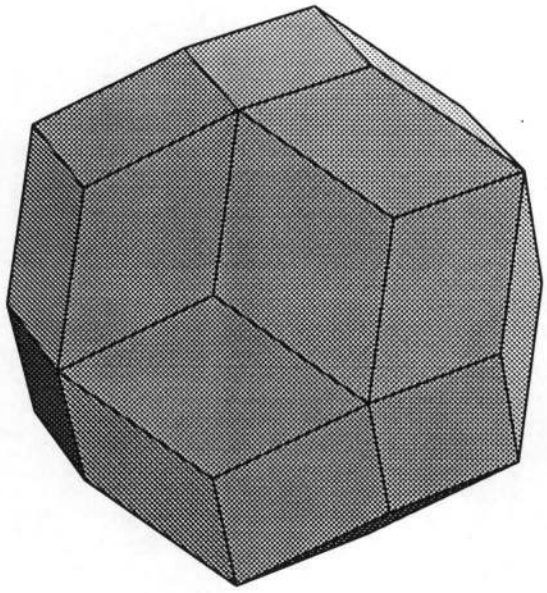
```
Zonohedron[{{1,1,0},{1,-1,0},{1,0,1},{1,0,-1},
            {0,1,1},{0,1,-1},
            {Sqrt[2/3],Sqrt[2/3],Sqrt[2/3]},
            {Sqrt[2/3],Sqrt[2/3],-Sqrt[2/3]},
            {Sqrt[2/3],-Sqrt[2/3],Sqrt[2/3]},
            {Sqrt[2/3],-Sqrt[2/3],-Sqrt[2/3]},
            {Sqrt[2],0,0},{0,Sqrt[2],0},{0,0,Sqrt[2]}}]
```



The Rhombic Triacontahedron

This is the dual of the icosidodecahedron, one of the Archimedian solids that (like the cuboctahedron) is not a zonohedron as it has odd faces.

```
Zonohedron[{{1,0,-GoldenRatio},{1,0,GoldenRatio},
            {0,-GoldenRatio,1},{0,GoldenRatio,1},
            {-GoldenRatio,1,0},{GoldenRatio,1,0}}]
```



The Truncated Icosidodecahedron

This is also known as the Great Rhombicosidodecahedron.

```
Zonohedron[{
{1,GoldenRatio,GoldenRatio-1},{1,-GoldenRatio,GoldenRatio-1},
{1,-GoldenRatio,1-GoldenRatio},{1,GoldenRatio,1-GoldenRatio},
{GoldenRatio,1-GoldenRatio,1},{GoldenRatio,1-GoldenRatio,-1},
{GoldenRatio,GoldenRatio-1,-1},{GoldenRatio,GoldenRatio-1,1},
{GoldenRatio-1,1,GoldenRatio},{GoldenRatio-1,-1,-GoldenRatio},
{GoldenRatio-1,1,-GoldenRatio},{GoldenRatio-1,-1,GoldenRatio},
{2,0,0},{0,2,0},{0,0,2}}]
```