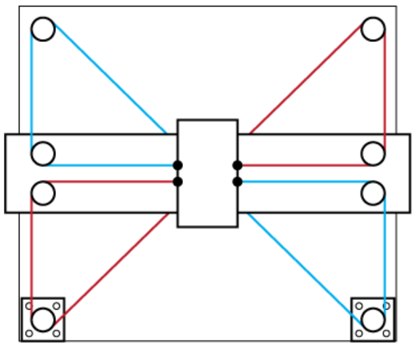
CoreXYZ

Ryan Carlyle 4/26/2014

This is a novel three-axis gantry mechanism that has no moving motors. It takes the belt principle used by CoreXY and extends it to three dimensions to make CoreXYZ.

Introduction – Principles of CoreXY

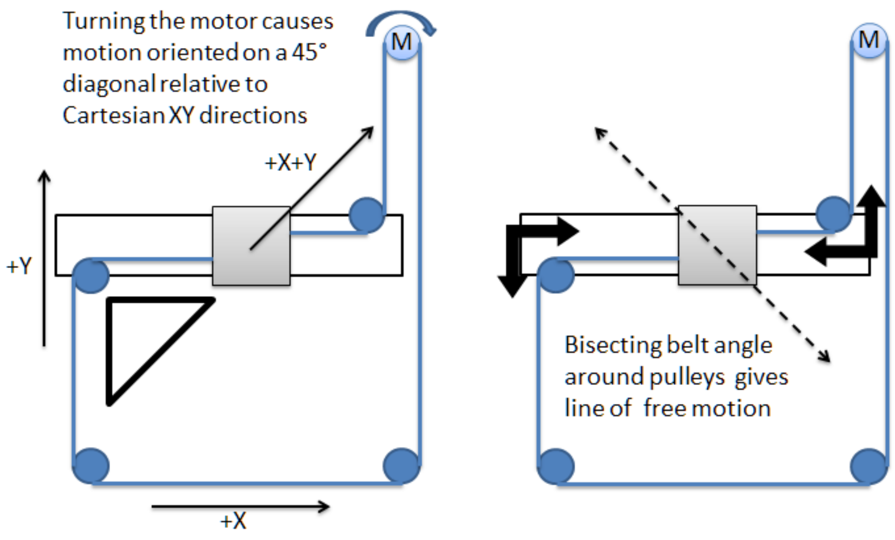
As a refresher, here is the simplest theoretical belt path for CoreXY:



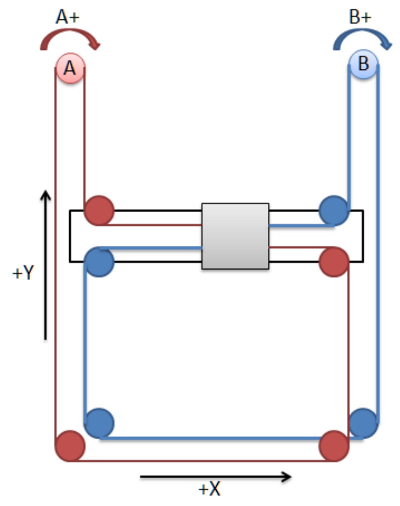
coreXY.com

The basic principle of operation is that the symmetrical deflection pulleys on the bridge translate *perpendicular belt travel* into *diagonal carriage motions*. This causes the carriage to move on a 45° diagonal relative to the gantry's mechanical X and Y coordinates.

Each belt controls motion in one diagonal direction. Thus if only one belt is installed, the carriage is free to move on the perpendicular axis. (This is the same as standard Cartesian gantries, except rotated 45°.) **Each belt defines an imaginary "line of free motion." With two belts, the carriage must sit at the intersection of these two lines.** All XY gantry systems have two degrees of freedom and thus require two axis control mechanisms (eg belt+motor). This concept is important later. Each control belt defines a line of freedom.


Turning the motor causes motion oriented on a 45° diagonal relative to Cartesian XY directions
M
+X+Y
+Y
+X
M
Bisecting belt angle around pulleys gives line of free motion

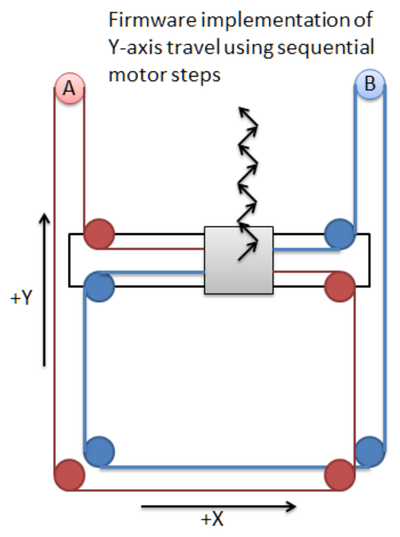
In a normal Cartesian gantry, the X motor causes X motion and the Y motor causes Y motion. But the CoreXY gantry is more complex – each motor causes motion in both X and Y. So let's name the motors A and B instead. **Rotation in the +A direction can be said to produce +X-Y motion and rotation in the +B direction can be said to produce +X+Y motion.** This is the basic coordinate transform function from *angular position of the motors* into *linear motion of the carriage*. Firmware motion control performs the inverse transform to convert G-code motion commands to motor steps.



Motor Direction	Cartesian Direction X	Cartesian Direction Y
+A	+	-
-A	-	+
+B	+	+
-B	-	-

But these diagonal motion directions do not correspond to the physical arrangement of gantry components. The physical gantry moves in X and Y directions. And the Y bridge must have higher moving mass than the X carriage. So for optimal motion performance, the acceleration planner should associate moving masses with Cartesian directions. This means the motion planner first works in Cartesian coordinates, and then a coordinate transform is required to convert Cartesian motion into the 45° diagonal coordinate system associated with motor rotation.

Standard CoreXY gantry control implementations (eg Sailfish) perform this 45° transform via firmware. Then motor step commands are sent sequentially – one motor at a time. Travel in a Cartesian direction is accomplished by rapidly alternating motors to approximate a straight line in X or Y. The commanded carriage position actually traces out a tiny zig-zag path (width ~= 0.01mm), but the mass of the gantry effectively damps this oscillation into a straight line motion. (Again, this is the same as Cartesian gantries, but rotated 45°.)


Firmware implementation of Y-axis travel using sequential motor steps
A
B
+Y
+X

However, an alternate transform method is possible via hardware, by synchronizing the motors. If [+A]=[+X-Y] and [+B]=[+X+Y] then vector addition tells us [+A] plus [+B] gives 2[+X]. The Y component cancels. Turning both motors simultaneously, at equal speed but in various combinations of directions, will actually produce true Cartesian motion aligned with the gantry axes.

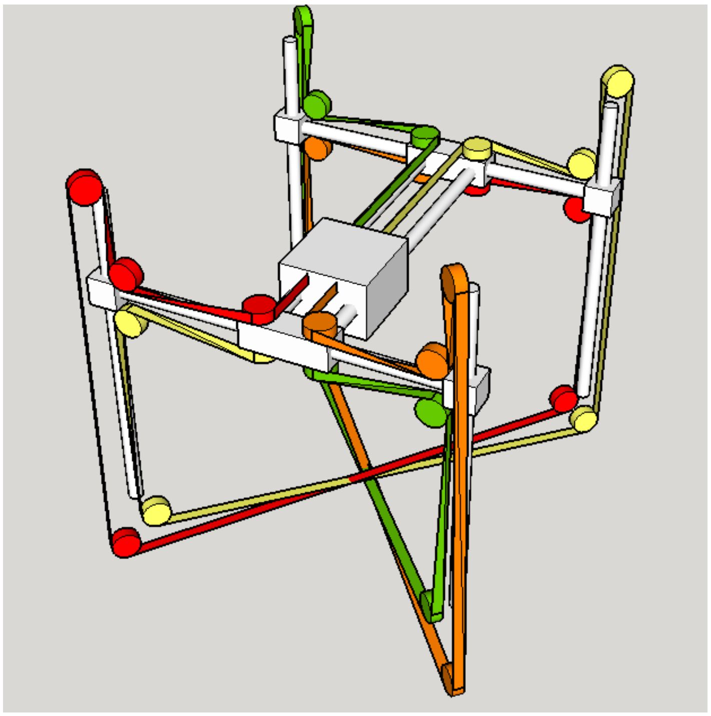
	Motor Direction	
Cartesian Direction	A	B
+X	+	+
-X	-	-
+Y	-	+
-Y	+	-

Synchronizing motor motion in this manner would allow a CoreXY gantry to achieve true Cartesian motion with no zig-zag travel pathing. This technique is not in use today because existing motor control systems can only step one motor at a time. But simultaneous stepping could be implemented fairly easily via a hardware solution such as a simple FPGA harness inserted between the main control board and stepper drivers. The FPGA gate logic would accomplish the above AB/XY transform so that standard Cartesian firmware could drive a CoreXY gantry as if it were Cartesian. The firmware would simply command a Y direction step like normal, and the FPGA would coordinate both stepper drivers to execute that move in CoreXY hardware.

This synchronized stepping concept has not yet been used for standard CoreXY, because it adds complexity with only minor advantages over a firmware-transform implementation. But it has significant advantages for CoreXYZ which will be seen later.

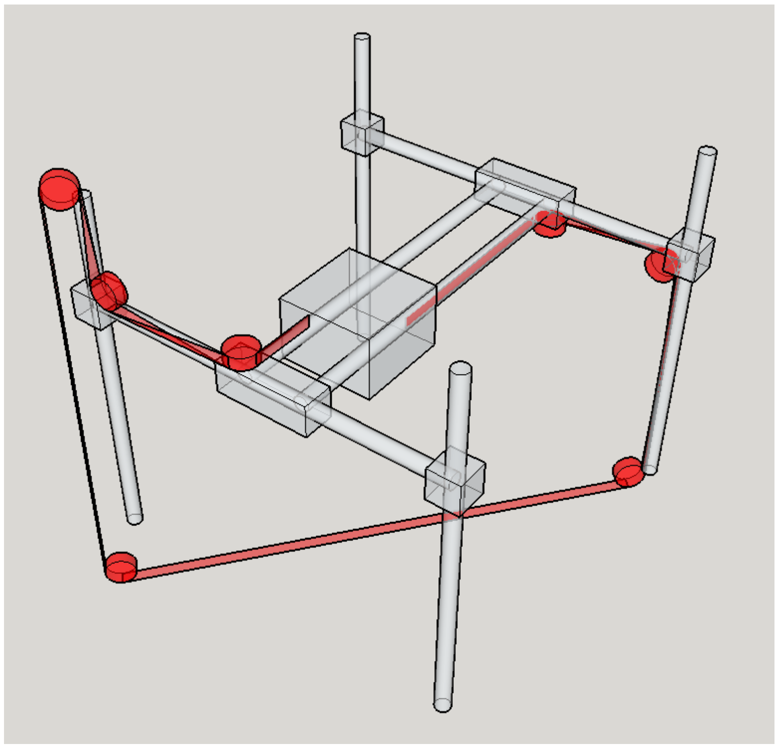
Principles of CoreXYZ

Three-axis gantry, drawn with all four CoreXYZ belts:

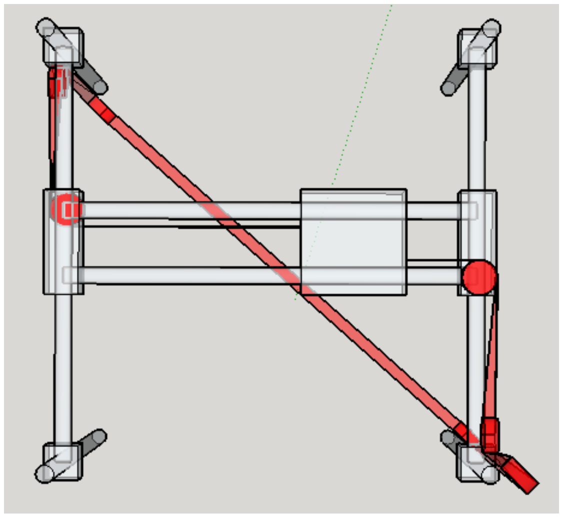


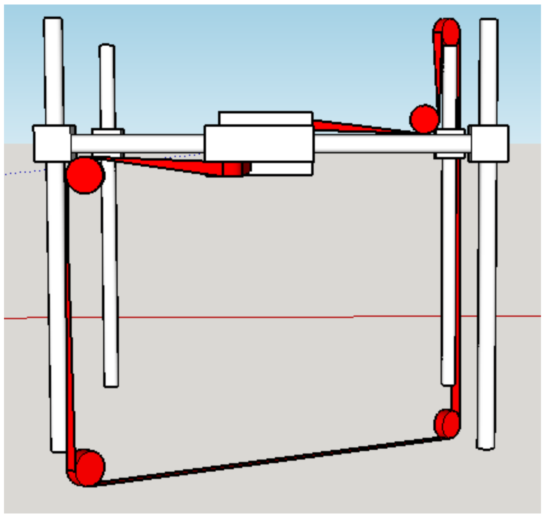
This is a three-axis motion mechanism, where the carriage travels in all directions. The carriage moves in X, on a bridge that moves in Y, on a gantry that moves in Z. Normally, three-axis motion requires high-mass components to move – either steppers mounted on the gantry, or an independent mechanism to move the build table. CoreXYZ has the unique advantage that the drive motors and build table are all stationary. This could significantly reduce moving mass, while allowing larger motors and studier build tables.

For clarity, here is a single CoreXYZ belt stage:



When viewed from above or the side, CoreXYZ utilizes the same basic belt arrangements as CoreXY:



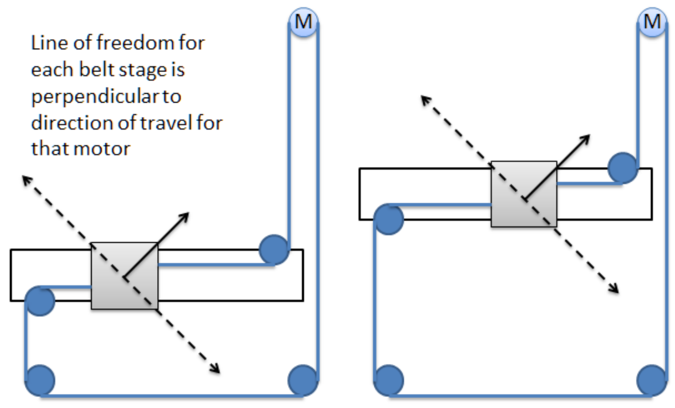


The novel aspect is putting two sets of orthogonal deflection pulleys on each leg. This is the CoreXY principle extended into three dimensions. The symmetrical pulleys on the bridge and gantry translate perpendicular belt travel into diagonal carriage motions. This causes the carriage to move on a 45° diagonal relative to the gantry's mechanical X, Y, and Z coordinates.

Motion Characteristics

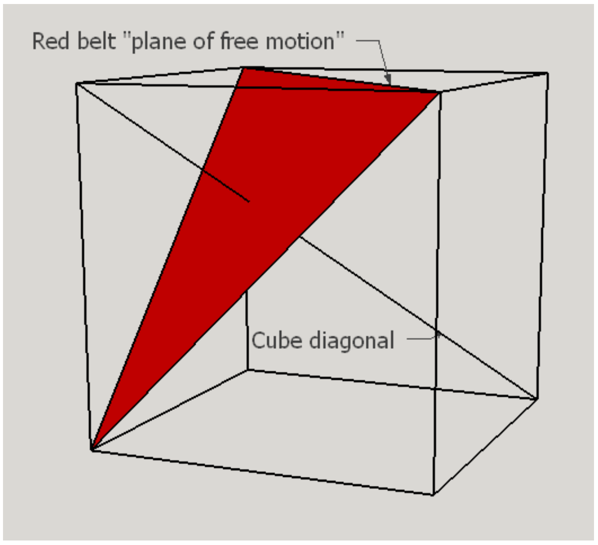
With traditional CoreXY, each belt defines a line of free motion in the XY plane. Driving the motor shifts this line between opposite corners. Adding the second belt then adds a second orthogonal line of free motion, which intersects with the first to uniquely define a location in XY space.

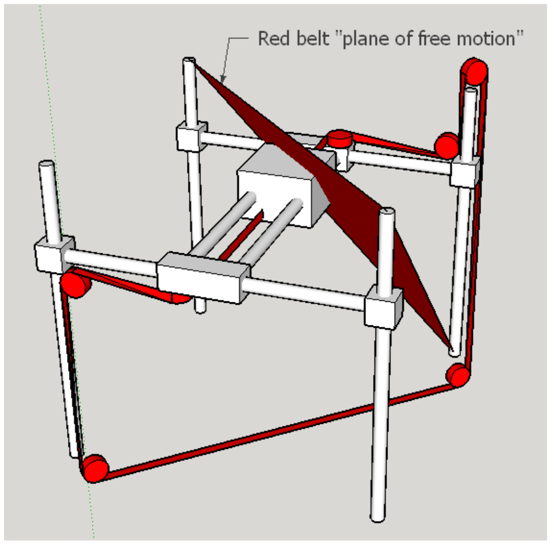
Carriage line of freedom for one belt with CoreXY:


Line of freedom for each belt stage is perpendicular to direction of travel for that motor
M
M

But with CoreXYZ, the use of two orthogonal deflection pulleys on each side of the carriage creates an additional degree of freedom for each belt.

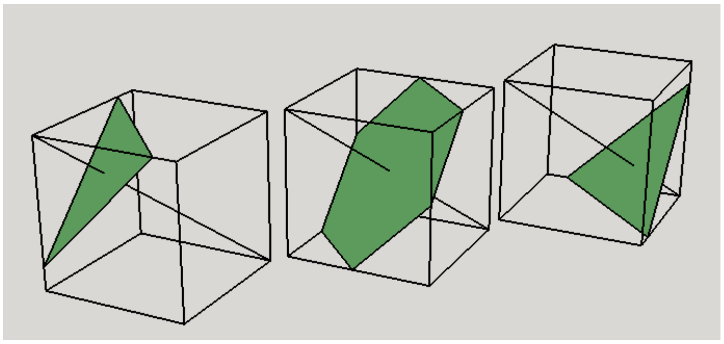
Carriage plane of freedom for one belt with CoreXYZ:



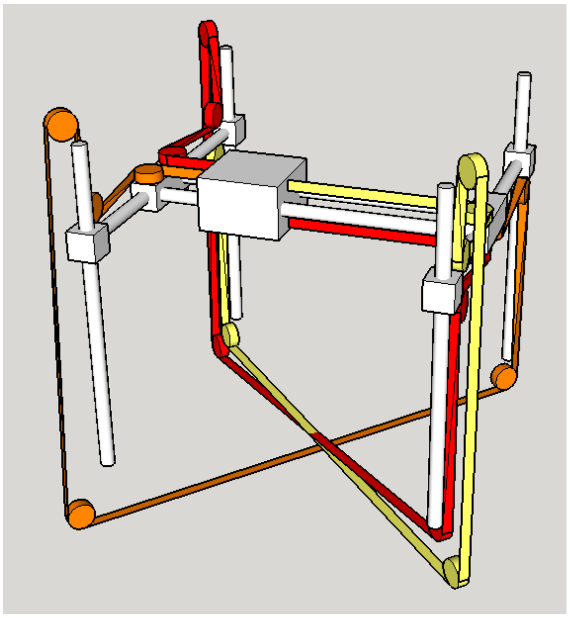


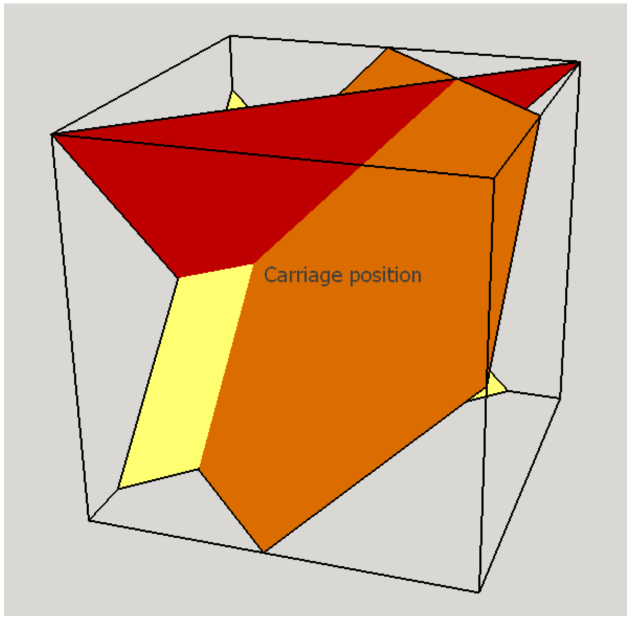
If only one belt is installed and the motor is locked, the carriage has two degrees of freedom and thus is constrained to a plane. This plane is orthogonal to the line connecting opposite corners of the cube.

Driving the motor shifts that plane of free motion between opposite corners:



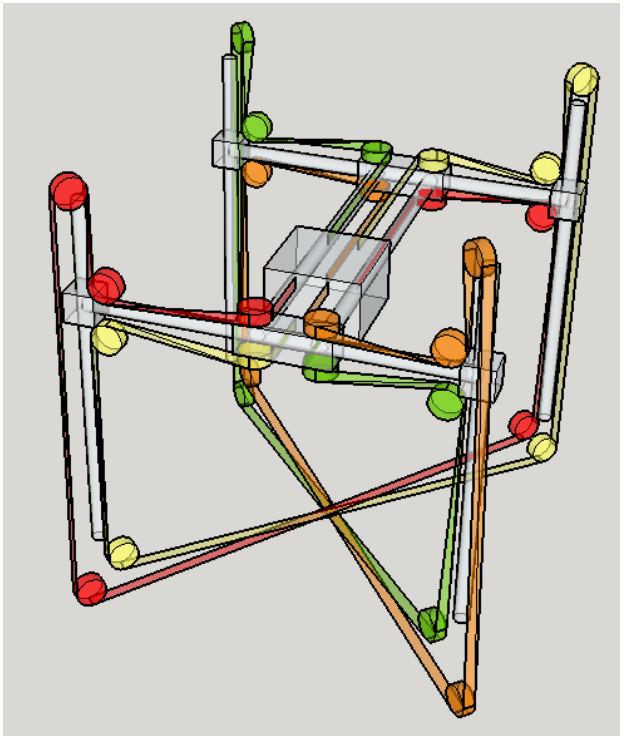
Adding two additional belts adds two additional plane constraints. This uniquely defines the carriage position at the three-way plane intersection point:





However, with the three-belt arrangement, one of the belts is not balanced and this creates a racking force on the gantry. So we add an additional belt to populate the fourth cube diagonal.

All four belts:

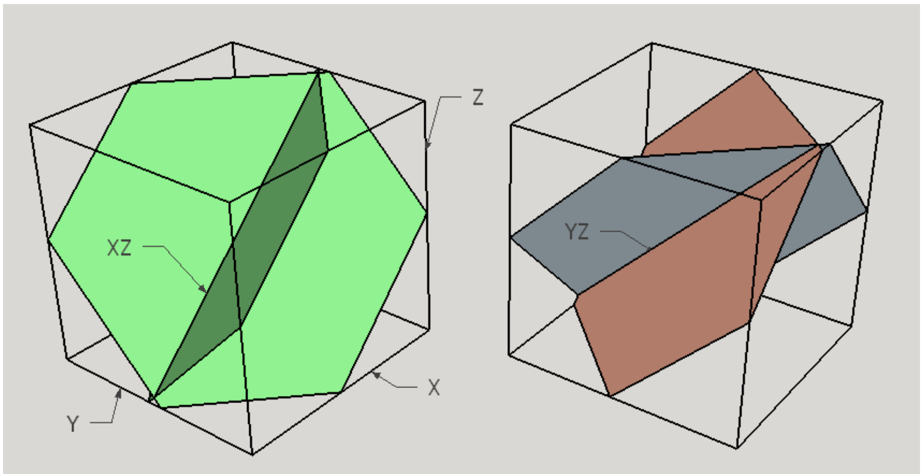


This is where the difference between sequential motor steps (as used today) and synchronized motor steps starts to become important. With four belts, the system is over-constrained. All four planes physically must intersect at one point but this cannot be accomplished by moving only one of four planes. Furthermore, existing motion control hardware generally does not have a spare output channel for a fourth motor. **So sequential stepper control probably requires the fourth belt be an undriven idler.** This belt's tension balances the racking force when the gantry is stationary, but it remains to be seen how much racking will occur during carriage motion.

The other downside to sequential stepper control is that each belt causes an XYZ motion when independently moved. So now instead of the 2-dimensional zig-zag of CoreXY, Cartesian-axis motion requires tracing out a helix that includes a Z component. The gantry mass should damp the Z component so there is no impact on print layer height, but this requires physical testing to confirm.

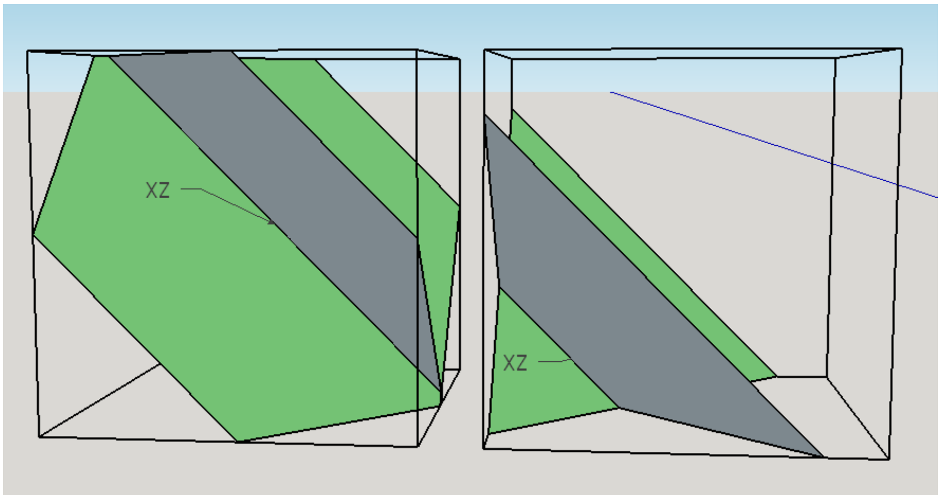
A superior implementation utilizes synchronized motor steps to coordinate all four belts. Surprisingly, running all four motors in sync converts CoreXYZ into a true-Cartesian motion system, in exactly the same way CoreXY can be converted to Cartesian by running two motors in sync.

This remarkable property occurs because the intersection of any two belt planes defines a line that is flattened (two dimensional) on an XZ or YZ axis:



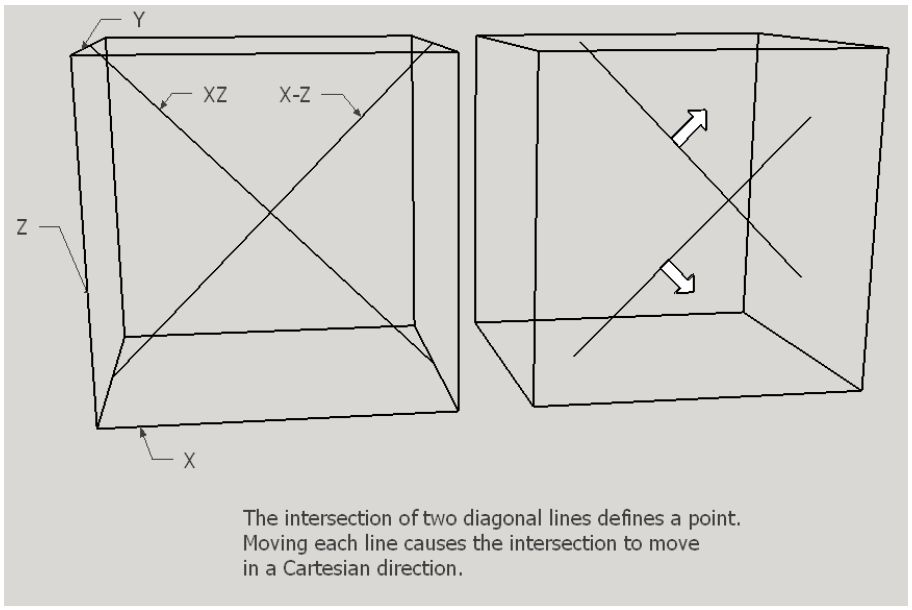
Any two belt planes define a 45 degree line,
oriented either XZ or ZY

Moving those intersecting planes in sync (by running the motors in sync) causes the intersection line to move in a "flat" direction, canceling one of the axes of motion.



Moving two steppers in sync moves the diagonal intersection line
perpendicular to its path (eg XZ line moves in +X-Z or -X+Z direction)

So any two of the four belts define a line of free motion for the carriage. And the other two belts define a second, perpendicular line of free motion for the carriage. These two lines function exactly like the lines of free motion in a CoreXY arrangement.


Y
XZ
X-Z
Z
X
The intersection of two diagonal lines defines a point.
Moving each line causes the intersection to move
in a Cartesian direction.

When the motors are run in pairs, CoreXYZ has identical physics of motion to CoreXY. Except the combination of motor pairs can be changed to control motion on the third axis.

This means the point of intersection of the four planes will move in Cartesian space when the motors are run in sync. This is a fantastic property because it has the potential to dramatically simplify motion planning. The axis-to-motor transform function is simple enough to easily implement via hardware logic:

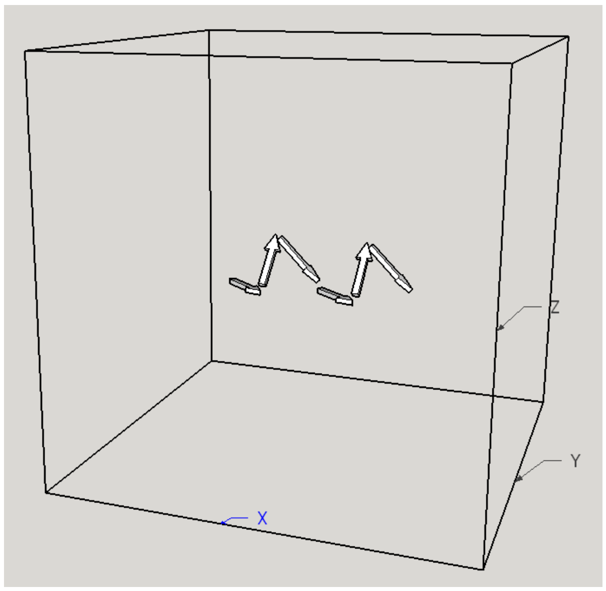
	Motor Direction			
Cartesian Direction	A	B	C	D
+X	+	+	-	-
-X	-	-	+	+
+Y	+	-	-	+
-Y	-	+	+	-
+Z	+	+	+	+
-Z	-	-	-	-

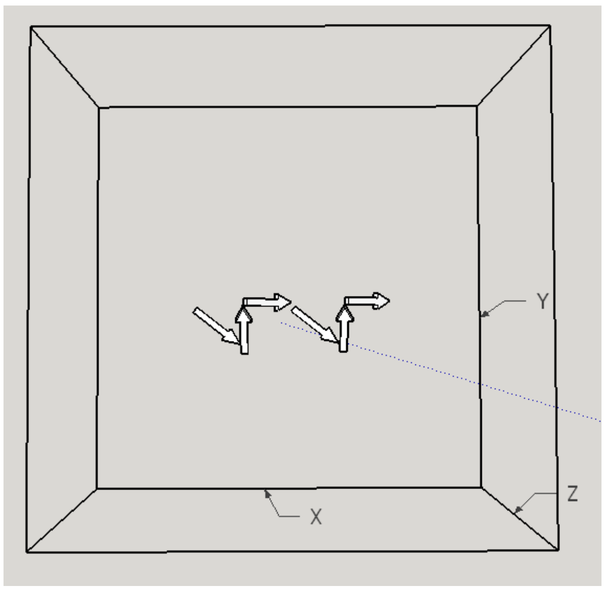
So this is the ideal control case – four motors synchronized by hardware, with the firmware merely commanding Cartesian motions.

But a normal “sequential” control implementation via firmware would step the motors in a similar manner. For example, to move one step in +X, it would step +A,+B,-C in sequence. (D is an idler belt and is moved slightly during each A,B,C motion.) One –Z step would require stepping –A,-B,-C in sequence.

This sequential step mode will cause the commanded carriage position to trace out a repeating three-legged helix, in a similar manner to the zig-zag required for CoreXY to move in Cartesian directions.

Side and overhead views of this helical motion:



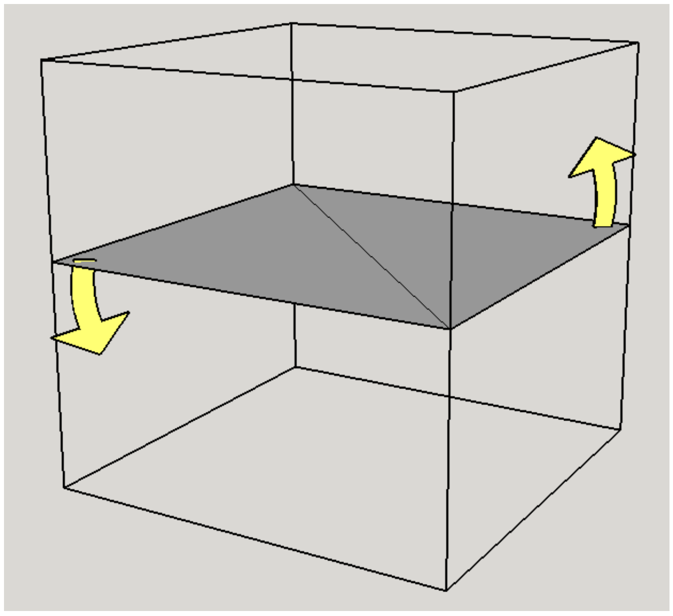


The result of the three sequential steps is a net displacement along the X axis, exactly the same as if the three motors were stepped simultaneously. Presumably the system's moving mass will damp the helix and result in smooth linear motion.

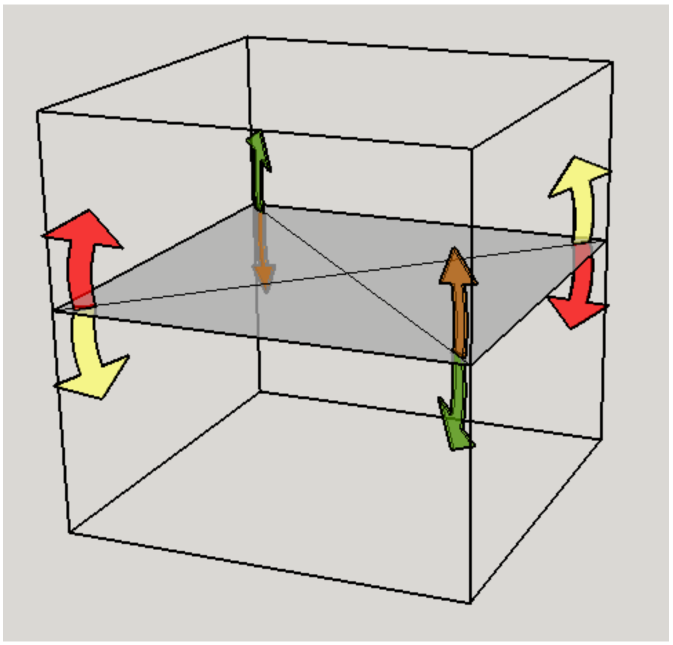
Gantry Leveling

The build plate is static and rigid. Leveling the XY gantry relative to the build plate is performed by skewing the corner heights via belt adjustment.

Tightening one belt results in a diagonal twisting/skewing force on the XY gantry relative to the Z rails:

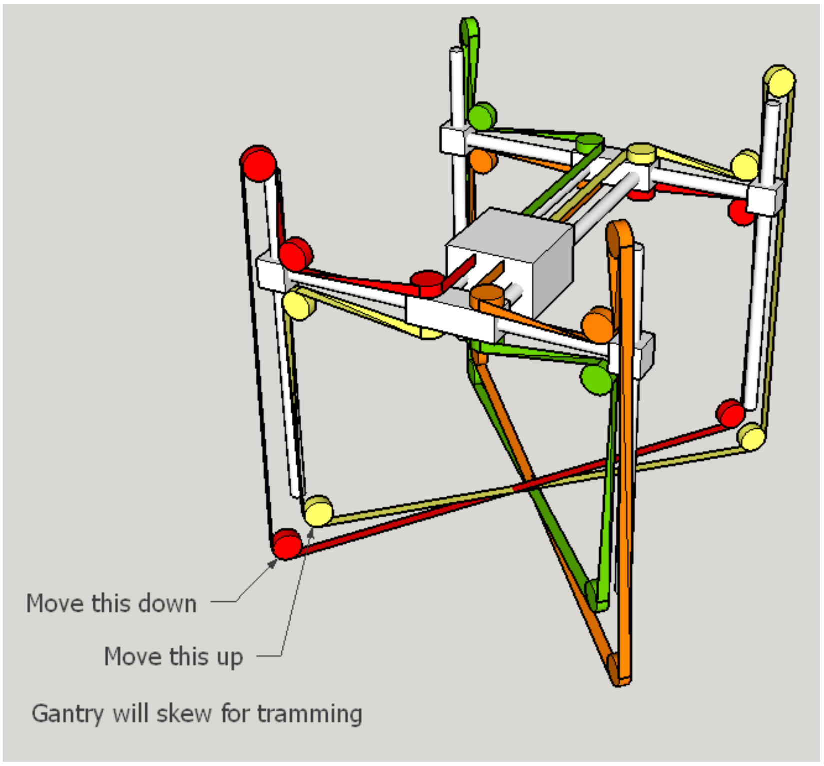


The combination of four belts allows skewing the gantry in any direction:



However, adjusting individual belt tension in this manner will cause racking forces, both on the gantry Z rails and bridge Y rails. It also means complex four-point leveling where each belt's tension acts against the opposite belt.

The most user-friendly way to level the gantry is probably *opposing pulley adjustment* by moving pairs of pulleys in opposite directions to change two belts at once. Belt tension is kept constant across all four belts with matched spring idlers, and a pair of external belt routing pulleys is adjusted to change the effective path length of an opposing pair of belts.


Move this down
Move this up
Gantry will skew for tramming