Kuper Virtual Axes Convention and Data Interchange

Properly calibrated, a Kuper is capable of producing extremely accurate moves in Cartesian space, via virtual (synthetic) axis synthesis.

All modes, programming, shooting, and data manipulation are available as virtual axes, the obvious limitation being that the virtual move has to fit into the physical envelope of the rig, with a little space to spare.

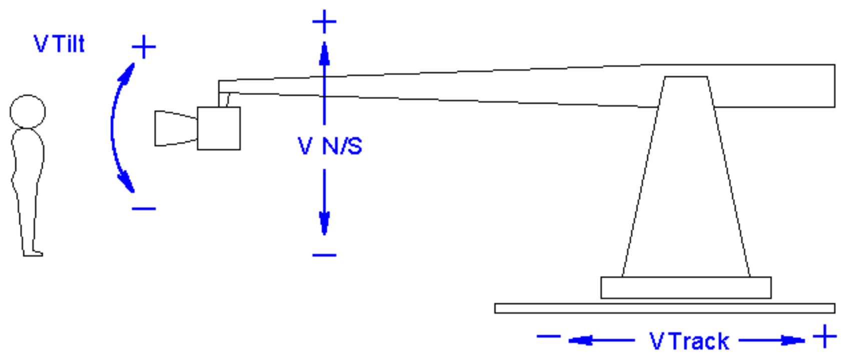
Typically an operator will want to ensure he leaves a couple of extra feet of available track travel behind the rig, since it will be used to compensate swing, and that he tries to stay within about 70 degrees with the swing axis, since one of the math terms gets very large as the swing approaches 90 degrees. If this happens, you get a divide-by-infinity error and the Kuper will lock you out of your move.

Some axes may be counter-intuitive, since the Kuper space uses direction conventions from ye olde days optical printing, rather than right-hand space CGI conventions.

By convention, virtual axes are always calibrated in real world units. If data scaling is needed, say to match a miniature, the data is scaled to the real units of the rig. Likewise, by convention, data exported from motion control rigs is exported at real world size; the CG application receiving the data scales it up from miniature size as necessary.

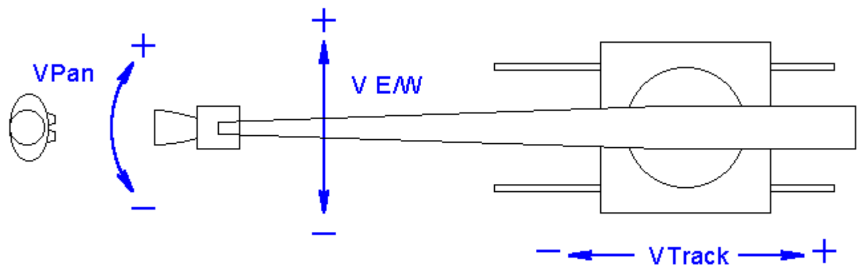
By convention, all stage rigs are calibrated in real inches and degrees or real centimeters and degrees. feet and meters, though often used in CG space, are too big for stage use, and Kuper specifically requires degrees, angles in radians will crash the virtual software.

For a typical application where a motion control rig is set up “nose into” the set, the axis conventions look like this

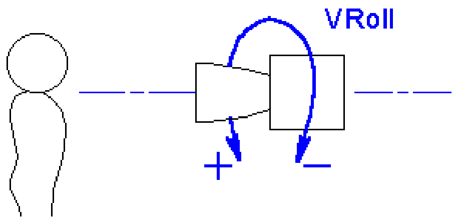


Vtrack	Virtual Track; Calibrated in real-world centimeters. Vtrack is roughly analogous to the depth axis in a typical CG setup. Note that Vtrack *increases* moving out *away* from the set, and *decreases* moving in *toward* the set, this may be counter-intuitive to many operators. Zero is arbitrary.

VN/S	Virtual North/South; Calibrated in real-world centimeters. VN/S is analogous to the vertical axis in a typical CG setup. VN/S increases when rising, and decreases when falling. Zero is almost always boom level.
VTilt	Virtual Tilt; Calibrated in real-world degrees. Vtilt increases tilting up, and decreases tilting down. Zero is tilt level.



VE/W	Virtual East/West; Calibrated in real-world centimeters. VE/W is analogous to the X axis in a typical CG setup. VE/W increases as the boom swings right, and decreases as the boom swings left. Zero is the boom centered over the track.
VE/W	Virtual Pan; Calibrated in real-world degrees. Note that Vpan is a left-handed axis, and it *increases panning right*, and *decreases panning left*. This conforms to optical printer conventions, but is backward compared to most right-handed CG conventions. Zero is the camera looking directly down the track in the –Z direction.

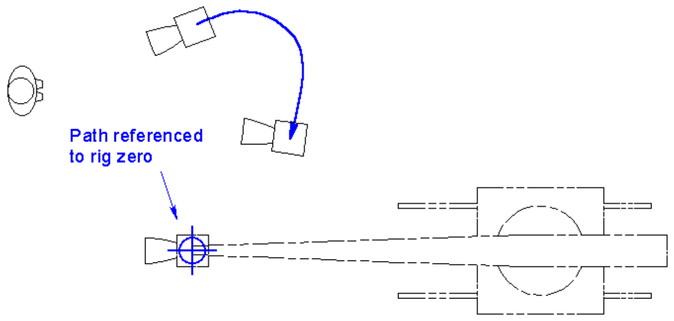


VRoll	Virtual Roll; Calibrated in real-world degrees. Vroll increases as the top of the camera rolls to the right, and decreases and decreases as the top of the camera rolls to the left. The other way to think about it is that roll increases as the image rolls counter-clockwise, and decreases as the image rolls clockwise. Zero is roll level.
I/O	Interoccular; Calibrated in real-world millimeters. I/O increases as the camera moves toward the right eye, and decreases to zero as the camera moves toward the left eye. Zero is usually the left eye.

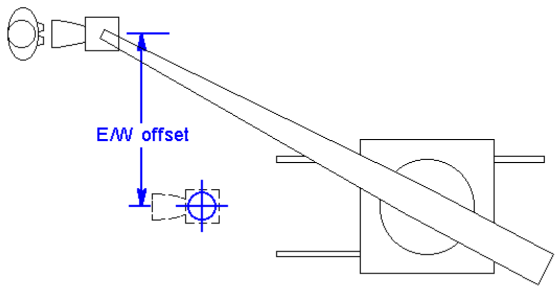
Referencing the Origin

A Kuper move is a true, fully defined, path, but unfortunately that path is referenced to nothing more than the arbitrary point in space where the rig was zeroed.

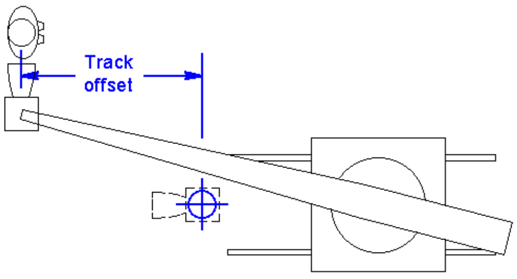
Useful information, surely, but eventually, all data has to reference back to the set, and the way to do that is by determining and recording the position of the rig zero and major axes relative to the set.



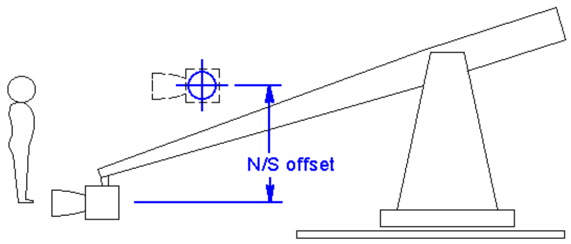
Fortunately, in most cases, you can use the rig itself to quickly measure the position some prominent points in the set in your coordinate system.



Say, for example, you wanted to measure the spot where a subject was standing. You could lock off the pan axis at 0°. That would ensure that you were looking directly down the Vtrack axis. If you carefully approached the spot you wanted to measure, and centered it in your finder, since you were looking straight down Vtrack, your position on the VE/W axis would be the same as the spots VE/W coordinate.



Likewise, if you backed off a little, panned 90° right and locked it there, you’d now be looking straight down the VE/W axis you could approach the spot from the side, and when you were directly across from it, that would be the spots Vtrack Coordinate.



Alternately, you could tilt straight down, and move until you were looking straight down on the spot, and get both horizontal coordinates that way.

Similarly, if you lock the tilt at 0°, you could approach the spot and read its height. On many setups, it's unlikely you could get low enough to actually measure the floor, but you could look through the finder at a ruler and add or subtract the correct offset.
The number of points you record, and the accuracy will depend on how tightly the CG generated elements have to integrate with the plate. Elements that visibly sit on the floor have to track very well, and ideally, in cases like this you want to carefully record the positions of several prominent reference objects.

Elements that won't suffer much from float, like flying foreground objects or atmospherics, need much less data.

If the set has a prominent axis, you should record any angle between that axis and the motion control rig.

Two notes for the CG guys

Vpan is a left-hand axis that's "backwards" relative to the vertical vector. Vpan *increases panning right*, and *decreases panning left*. This conforms to optical printer conventions, but is opposite most right-handed CG conventions.

The Pan / Tilt / Roll axes are set up like a Yaw / Pitch / Roll space and not a RotX / RotY / RotZ space. The stages are nested, with the roll moving relative to the tilt, and the tilt moving relative to the pan stage, instead of being measured relative to the global coordinate axes. Your system must account for the nesting order and effect. Nesting order is VPan then Vtilt then Vroll.

Data and Disk Formats

Kuper uses two file formats, a native format with lots of metadata, and a simpler tabulated text format (called an ASCII file) which is useful for moving data between systems that can't read each other's native files.

Some CG developers have written plug-ins that directly read and write native Kuper files, but since the software landscape in a busy compositing operation is ever changing, having the right translator at hand is fairly rare. Most interchange between machines takes place via ASCII text files.

An ASCII file is to a Kuper move file like a **.**txt file is to a **.**doc file. It's just the raw position data for each frame, in ASCII text, with none of the metadata or formatting.

A typical Kuper ASCII move file, when opened with a text editor, using fixed-width fonts, looks like this…

```
Axes = VTrack, VEW, VNS, VPan, VTilt, VRoll
100.6510  13.9740  10.5370  -5.8610  -3.3848  13.1000
100.6510  13.9740  10.5370  -5.8611  -3.3848  13.1000
100.6510  13.9740  10.5370  -5.8612  -3.3848  13.1000
100.6510  13.9740  10.5370  -5.8613  -3.3848  13.1000
                          •
                          •
                          •
```

… and so on, with one row per frame. The 6 columns above are typical, but there can be additional columns that track additional axes, or a courtesy column tracking frame numbers.

Kuper files are created and used on a machine running an industrial version of DOS, and files intended to work with Kuper should adhere to the 8**.**3 naming convention and DOS legal formatting.

By convention, native Kuper moves have either a **.**mov extension or no extension, and ASCII files have a **.**asc extension.

Exporting data

ASCII files are created by Kuper using the "AsciiFile" command on the main page (right column, halfway down). Clicking this command opens a dialogue box prompting you to either save or load an ASCII file.

If you choose to save a file to disk, you open a dialogue page very similar to the normal "save file" command, and you can select which axes you want to save.

There is one option available in ASCII file save that is not available on a regular move is the option of where you want to specify the sample. Upon providing a file name and invoking the "save file" command, you'll get a dialogue box asking "sample shutter closed / sample center blur image"

This is asking whether you want to specify the even frame boundaries (1.0, 2.0, 3.0, etc) where the shutter would traditionally be closed, or the center frame points (1.5, 2.5, 3.5, etc) where a film camera shutter would traditionally be open.

On a film shoot, you'll pick "center blur, since this is where the actual exposures were made. On digital animation shoot where the Kuper is controlled remotely, you'd typically specify "center shutter closed" since the most of these systems issue even frame numbers.

Although it's not necessary, many operators add a dummy axis named "frame" to the move, and set up a linear move from 0 at frame 0 to x at frame x, that way there's a courtesy counter in the data that clocks off frame numbers in the file, and your file comes out looking like this…

```
Axes = Frame, VTrack, VEW, VNS, VPan, VTilt, VRoll
0.000  100.6510  13.9740  10.5370  -5.8610  -3.3848  13.1000
1.000  100.6510  13.9740  10.5370  -5.8611  -3.3848  13.1000
2.000  100.6510  13.9740  10.5370  -5.8612  -3.3848  13.1000
3.000  100.6510  13.9740  10.5370  -5.8613  -3.3848  13.1000
                              •
                              •
                              •
```

… which saves a lot of confusion in post.