

Koviz User's Guide

Version 0.0.1

August 2025

National Aeronautics and Space Administration
Johnson Space Center
Houston, Texas



Table of Contents

.....	2
Introduction.....	4
RUN Directories.....	4
RUN Files.....	4
MONTE Directories.....	4
DP Files.....	5
DP File Syntax.....	5
DP Line and Symbol Styles.....	7
DP File Example.....	7
Session Files.....	8
External Programs.....	10
External Program DP File Specification.....	10
External Program DP File Example.....	10
External Program Source Code Example.....	11
Building Example External Program On Linux.....	11
External Program Building.....	12
Linux.....	12
MacOS X.....	12
External Program Summary.....	12
External Program Problems And Caveats.....	12
Scripts.....	13
Plot Interaction.....	14
Maps.....	16
Groups.....	17
Video.....	18
Introduction.....	18
Building Koviz with MPV Support.....	18
Data Synchronization.....	18
Commandline -video and -videoOffset options.....	18
Commandline -videoList option.....	19
Session Files.....	19
Video Directory Within RUN.....	19
Multiple RUNs And Multiple Videos.....	20
Blender.....	21
Introduction.....	21
Blender Addon Setup.....	21
Animate Blender Default Cube.....	21
Sync Koviz To Default Cube Animation.....	22
Sync Blender To Koviz With ES's Vismo.....	23
Live Trick TV Plots.....	24

Introduction.....	24
Plotting TV Params.....	24
Changing Live Plot Domain.....	25
Live DP Files.....	25
Live Spring Plot Example.....	25
Profiling Trick Jobs.....	27
Add Sleeps To Ball L2 Sim.....	27
Run And View Sleep Effect.....	27
Modify and Build Profiler.....	28
Add Profiler To Sim.....	28
Run Sim And See Profile Results.....	28

Introduction

Koviz is a simulation data plotting and visualization tool. It is designed especially for Trick monte carlo data analysis, comparing simulation runs, analyzing data spikes and creating report quality plot booklets. Koviz can be run interactively via the GUI or can be run in batch. Koviz supports Trick binary data and CSV. Koviz also offers:

- A real-time analysis report for Trick real-time data recordings
- A plugin-like functionality, external programs, to transform simulation data
- Data/video synchronization to view video alongside associated data plots
- Data/Blender graphics synchronization
- Live Trick simulation plots

The main usage for koviz is:

```
UNIX Prompt> koviz <RUN_dir|MONTE_dir|RUN file> [DP_file1...] [options]
```

To see a list of options:

```
UNIX Prompt> koviz <ENTER>
```

RUN Directories

Koviz plots data from RUN directories which contain log files (normally from Trick simulation runs). Koviz is efficient at coplotting multiple RUNs:

```
UNIX Prompt> koviz RUN*
```

RUN Files

Instead of giving koviz RUN directories that hold a set of log files, one can also directly plot files. For example, one can compare two *.csv files:

```
UNIX Prompt> koviz my-time-history.csv my-motion-capture.mot
```

MONTE Directories

Trick Monte carlo directories, typically prefixed by MONTE, contain a set of RUNs and a monte carlo inputs file. The Koviz GUI correlates RUN inputs to plot curves:

```
UNIX Prompt> koviz MONTE_dir
```

DP Files

The data product (DP) specification file syntax is shown below. All keywords are case insensitive. There are three main specifications in the file: PLOTS, TABLES, and PROGRAM. PLOTS refers to one or more pages of XY plots. TABLES refers to one or more pages of ASCII tables. The DP file may contain any number of Pages and Tables. A page may also have any number of plots. PROGRAM refers to an external program – a user created shared library dynamically linked into koviz for manipulating recorded data for display. A DP file can be created from the koviz GUI using the main menu's "Save As DP" option.

DP File Syntax

[Title]

[PLOTS:

PAGE page_number_integer: "page_title"

[TAB_LABEL: "Widget Tab Label"]

[FOREGROUND_COLOR: "#rrggbb|color_name"]

[BACKGROUND_COLOR: "#rrggbb|color_name"]

PLOT plot_number: "plot title"

[X_AXIS_LABEL: "x_axis_label"]

[Y_AXIS_LABEL: "y_axis_label"]

[X_MIN_RANGE: x_min_value]

[X_MAX_RANGE: x_max_value]

[Y_MIN_RANGE: y_min_value]

[Y_MAX_RANGE: y_max_value]

[PLOT_X_SCALE: "log|linear"]

[PLOT_Y_SCALE: "log|linear"]

[PLOT_RATIO: "square"]

[GRID: <on|off>]

[RECT: x%, y%, width%, height%]

[PRESENTATION: compare|error|error+compare]

[MAJOR_X_TICS: comma separated list of floats]

[MINOR_X_TICS: comma separated list of floats]

[MAJOR_Y_TICS: comma separated list of floats]

[MINOR_Y_TICS: comma separated list of floats]

... continued on next page

```

[HLIN: float]
  [COLOR: "#rrggbb|color_name"]
  [LABEL: "printf format - %g for value, %s for unit"]
  [UNITS: "unit"]
  [LABEL_UNITS: "unit"]
  [LABEL_ORIENT: "left|center|right|0<=placement<=1"]
X_VARIABLE: "param_name"
  [LABEL: "label"]
  [UNITS: "unit"]
  [SCALE_FACTOR: scale_value]
  [BIAS: bias_value]
Y_VARIABLE: "param_name"
  (Same attributes as X_VARIABLE)
  [SYMBOL_STYLE: symbol_style]
  [SYMBOL_SIZE: symbol_size]
  [SYMBOL_END: symbol_style]
  [LINE_STYLE: line_style]
  [LINE_COLOR: "#rrggbb|color_name"]
[Y_VARIABLE: ... ]
]
[TABLES:
  TABLE table_number_integer: "table_title"
    [TAB_LABEL: "Widget Tab Label"]
    VARIABLE: "param_name"
    (Same attributes as X_VARIABLE in PLOT)
    [VARIABLE: ...]
  ]
[PROGRAM: "path_to_program_shared_library":
  IN: "param_name1" "param_name2" ...
  OUT: "param_name1" param_name2" ...
]

```

DP Line and Symbol Styles

SYMBOL_STYLE:

None|Square|Circle|Star|XX|Triangle|
 Solid_Square|Solid_Circle|Thick_Square|Thick_Circle|
 Number_[0-9]|satellite

SYMBOL_END:

Same symbols as SYMBOL_STYLE

SYMBOL_SIZE:

Tiny|Small|Medium|Large

LINE_STYLE:

"No_Line|Thick_Line|X_Thick_Line|Fine_Dash|Med_Fine_Dash|
 Long_Dash|X_Long_Dash|Dot_Dash|2_Dot_Dash|3_Dot_Dash|
 4_Dot_Dash|Plain|Dash|Scatter"

LINE_COLOR:

"#rrggbb|<color_name>"

DP File Example

Example barebones DP file:

PLOTS:

PAGE 1: "Page Title"

PLOT 1: "Ball X Position"

X_VARIABLE: "sys.exec.out.time"

Y_VARIABLE: "ball.state.out.position[0]"

Session Files

To launch koviz with a set of RUNs and DPs, the -session command-line option may be used:

```
UNIX Prompt> koviz -session <session_file>
```

The syntax for the session file is as follows:

```
[# At beginning of line is a comment]
[Title]
[DEVICE: <File <file_name>|Terminal> ]
[FREQUENCY: cycle_time]
[TIME_MATCH_TOLERANCE: delta_time]
[PRESENTATION: Compare|Error|Error+Compare]
PRODUCT: DP_filename
[PRODUCT: DP_filename_n ...]
RUN: RUN_directory
[RUN: RUN_directory_n ...]
[T[1-4]: title1-title4]
[C[1-7]: line color 1-7]
[L[1-7]: legend label 1-7]
[LS[1-7]: linestyle 1-7]
[S[1-7]: symbol 1-7]
[FG: foreground color]
[BG: background color]
[START: start_time]
[STOP: stop_time]
[ORIENT: portrait|landscape]
[SHIFT: RUN shift times]
[TimeName: time_name]
[MapFile: map_filename]
[LEGEND: on|off]
[ShowTables: on|off]
[EXCLUDE: exclude pattern]
[FILTER: filter pattern]
[Video: filename]
[VideoOffset: time]
[ShowPageTitle: yes|no]
[ShowPlotLegend: yes|no]
```


[PlotLegendPosition: ne|e|se|s|sw|w|nw|n]

The DEVICE: statement specifies the visualization device for data output. Device types are Terminal (the default) and File. When Terminal is specified, the koviz GUI will be displayed. When File is specified the output is re-directed as PDF to the <file_name> file.

The FREQUENCY: statement specifies the delta time between data points when loading curve data. <cycle_time> is specified in seconds. If the optional FREQUENCY statement is unspecified or if the <cycle_time> is less than the recorded data cycle time, all recorded curve data will be loaded. If the <cycle_time> is larger, curve data points will be culled.

The TIME_MATCH_TOLERANCE: statement specifies the maximum allowed delta between the time stamps for two sets of data points used in generating the error plot. The default value is 1.0e-6. Koviz does a best match for timestamps within the tolerance.

The PRESENTATION: statement is only useful when two RUNs are specified. If presentation is “error”, a difference curve between two RUNs will be displayed. The points in the difference curve are matched by the TIME_MATCH_TOLERANCE. Interpolation is not done. If presentation is “compare”, curves from each RUN will be shown together. Finally, if presentation is “error+compare”, three curves will be shown: the difference curve alongside the curves from each RUN.

The PRODUCT: statement specifies a data product (DP) file. One or more DP files must be specified. DP files specify the simulation parameters to display, and the display attributes for each parameter. See DP Files for more detail.

The RUN: statement specifies a simulation RUN directory from which to retrieve logged data, <RUN_directory>. One or more RUNs must be specified.

The SHIFT: statement is used to shift RUN time(s). If there is a single RUN, specify a double value e.g. SHIFT: 13.56. If there are multiple RUNs, use a comma delimited list of RUN:<shift_value> specs e.g. SHIFT: RUN_a:13.56,RUN_b:77.28,RUN_c:23.0.

The TimeName: statement is used when time is not the default “sys.exec.out.time” e.g. TimeName: myTimeName.

The MapFile: statement is used to tell koviz where the map file is located. Maps are described later in this document, but in a nutshell maps are used to map RUNs with different parameter names so that the RUNs can be compared and difference plotted.

The EXCLUDE: and FILTER: statements either cull out or filter for logfiles within the RUN directories. This can help speed up data loading and/or show a subset of all variables logged.

External Programs

The DP PROGRAM specification provides a means for transforming data. Users build a program that is dynamically linked into Koviz for manipulating data specified in the DP specification file.

External Program DP File Specification

The <program_name> argument is a full path to a program which accepts the INs and generates the OUTs. This program must adhere to strict interface requirements. This program will be dynamically linked into the data products, which implies it must be built under specific guidelines. Only ONE external program may be defined per DP file.

The IN parameters are specified as a space delimited list of simulation variable names and external program OUT token names. The OUT list is a user defined name list which provides a unique token name for each of the external program output arguments. These output tokens may be used throughout the product specification file wherever a simulation variable name is used.

Inputs will be cast to doubles going to the external program, and outputs must be doubles as well.

As an example, a user would like to plot the rss of a vector. The product specification file (DP_*.file) might look like the following (NOTE: all strings surrounded in quotes):

External Program DP File Example

PROGRAM: "/home/kvetter/test/ext_programs/librss.so"

IN:

"Chaser.stVeh.stFFrDock.vecF_Frame[0]"

"Chaser.stVeh.stFFrDock.vecF_Frame[1]"

"Chaser.stVeh.stFFrDock.vecF_Frame[2]"

OUT:

"vecF_Frame_rss"

PLOTS:

PAGE 1: "External Program Example"

PLOT 1: "RSS"

X_VARIABLE: "sys.exec.out.time"

Y_VARIABLE: "vecF_Frame_rss"

External Program Source Code Example

The following is an example external program source file (rss.c) which takes three inputs and generates a single output. Bold font indicates code that is mandatory (i.e. “kovizProgram” is the function name that koviz expects for external programs).

```
#include <math.h>

void kovizProgram( double* in, int numIn, double* out, int numOut )
{
    out[0] = sqrt(in[0]*in[0] + in[1]*in[1] + in[2]*in[2]);
}
```

Building Example External Program On Linux

```
% cc -fPIC -c rss.c
% ld -shared -o librss.so rss.o -lm -lc
```

External Program Building

The external program created by the user must be built so that it can be dynamically linked into koviz.

In order to build, take the following steps according to platform.

Linux

Step 1. `cc -fPIC -c <myprogram1>.c` (compile all individual object this way)

Step 2. `ld -shared -o <myprogram>.so <myfunction1>.o <myfunction2>.o... [<myLib>.a] -lc`

MacOS X

Step 1. `cc -c <myprogram1>.c` (compile all individual object this way)

Step 2. `cc -bundle -o <myprogram>.so <myfunction1>.o <myfunction2>.o ... [<myLib>.a] -lc`

`<myprogram>.so` is the name that needs to be specified in the DP specification file. Set `LD_LIBRARY_PATH` so that koviz can find the shared object.

Try the following to see if the newly create shared object has unresolved dependencies:

```
UNIX% nm <myprogram>.so
```

External Program Summary

To use an external program:

Step 1. Build a DP spec file with the program name, inputs and outputs.

Step 2. Write an external program

Step 3. Build the external program.

Step 4. UNIX Prompt> `koviz DP_external_program RUN_name`

External Program Problems And Caveats

- Can't load shared library!!! Ensure that `$LD_LIBRARY_PATH` is set so that koviz can find the *.so file. The external program (*.so program) may have unresolved dependencies. Try "nm" on the external program, and look for "U"s.
- Koviz will not scale or bias X values with external programs.
- External programs convert everything to doubles, and only accept and output doubles.
- External programs have no notion of unit conversion.

Scripts

One may give koviz a script (or list of scripts) on the commandline. The intention of the script capability is to allow users to call their custom script with the current RUN as the first argument to the custom script. This allows users to, for example, select a RUN and fire off a custom visualization for the current RUN. The usage is as follows:

```
koviz -script "script1 arg1 arg2... argN, script2 arg1 arg2... argN, ..."
```

When koviz is brought up the UI will put the scripts in koviz' main menu. When the user selects the script off the menu, the script will be called with the current RUN as the first argument followed by the user supplied arguments. Additionally, one may place an ampersand in front of a character to hot-key the menu item.

For example, suppose one has a this bash script, show-run.sh:

```
#!/usr/bin/env bash
xmessage "$1"
```

Try the following with some RUN. Make sure double and single quotes are normal ASCII for this commandline call i.e. beware of cut-n-paste:

```
% koviz -script "&show-run.sh, &koviz -t1 'Calling Koviz'" <RUN>
```

Once koviz comes up, select Scripts > show-run.sh



Plot Interaction

Select Curve	Mouse Left Click Curve
Drag Curve	Mouse Left Click Curve to select Then Keypress Ctrl + Mouse Left Drag
Drag Curve Onto Another Curve	Mouse Left Click Curve to select Then Keypress Alt+Ctrl+Mouse Left Drag making init points of both curves close enough to “snap”
Zoom In	Mouse Middle Click and Drag Zoom Box
Zoom Out	Mouse Right Click or Keypress Escape
Pan	Mouse Left Drag In Middle Of Plot
Scale	Mouse Left Drag In Outter Extremity Of Plot
Toggle Single/Multiplot View	Mouse Left Click On Y-Axis Label or Plot Title
Toggle Y Axis Units	Mouse Wheel Over Y-Axis Unit In Curly Braces {unit}
Toggle Linear/Logscale	Mouse Wheel over x or y tic values
Suspend Live Coordinate Updates	Keypress Shift While Moving Mouse Off Plot
Toggle Presentation (compare/error)	Keypress Spacebar
Print Current Coordinate & Dx,Dy	Keypress Period
Measure Distance On Plot	Keypress Alt + Mouse Left Drag (updates koviz status bar)
Hop To Prev/Next Plot Coord	Keypress Left/Right Arrow (after curve selected)
Place Marker On Plot Coord	Keypress Comma
Change X Axis Variable (normal)	Options->Drag-n-Drop. Drag-n-drop variable onto x-axis label.
Change X Axis Variable (live data)	See section at end on Live Sim Data
Frequency Domain Toggle	Keypress f
Butterworth Filter	Keypress b
Savitzky-Golay Filter	Keypress g
Flip Curve About X-Axis	Keypress -
Derivative Of Curve	Keypress d
Integral Of Curve	Keypress i (use entry box for initial value, default to 0.0)
... continued on next page	...

Summation of all curves on plot	Keypress s
Magnitude of all curves on plot	Keypress m
Plot expanded live sim parameters on individual pages	Left Mouse Click Parameter
Plot expanded live sim parameters on individual plots on same page	Keypress Ctrl + Left Mouse Click Parameter
CoPlot expanded live sim parameters on same plot	Keypress Alt + Left Mouse Click Parameter
Start a new plot page and honor optional Ctrl and Alt keys	Keypress Shift + [Ctrl Alt] + Left Mouse click parameter

Maps

Maps are used to compare simulation RUNs whose variable names do not match. A map is either contained in a file or can be specified directly on the commandline. The map is a comma delimited set of key-values in the following form:

“key1=value1=value2...,key2=value1=value2,...”

Here is an example of comparing simulation position and velocity names differ.

```
UNIX Prompt> koviz -mapFile “spots2dyn.kvz” RUN_spots RUN_dyn
```

```
UNIX Prompt> cat spots2dyn.kvz
```

```
posx = spots.vehicle.pos[0] = veh[0].dynamics.posX,  
posy = spots.vehicle.pos[1] = veh[0].dynamics.posY,  
velx = spots.vehicle.vel[1] = veh[0].dynamics.velY,  
vely = spots.vehicle.vel[1] = veh[0].dynamics.velY
```

Maps may also be used to scale and bias variables. If the logged data has no units, units may be applied in the map file as well. Here is an example of map file with scale, bias and unit specs:

```
ForceX = Fx                = inertialForceZ {N},  
ForceY = Fy bias(-12.25)   = inertialForceX {N},  
ForceZ = Fz scale(-1.0)    = inertialForceY {N}
```

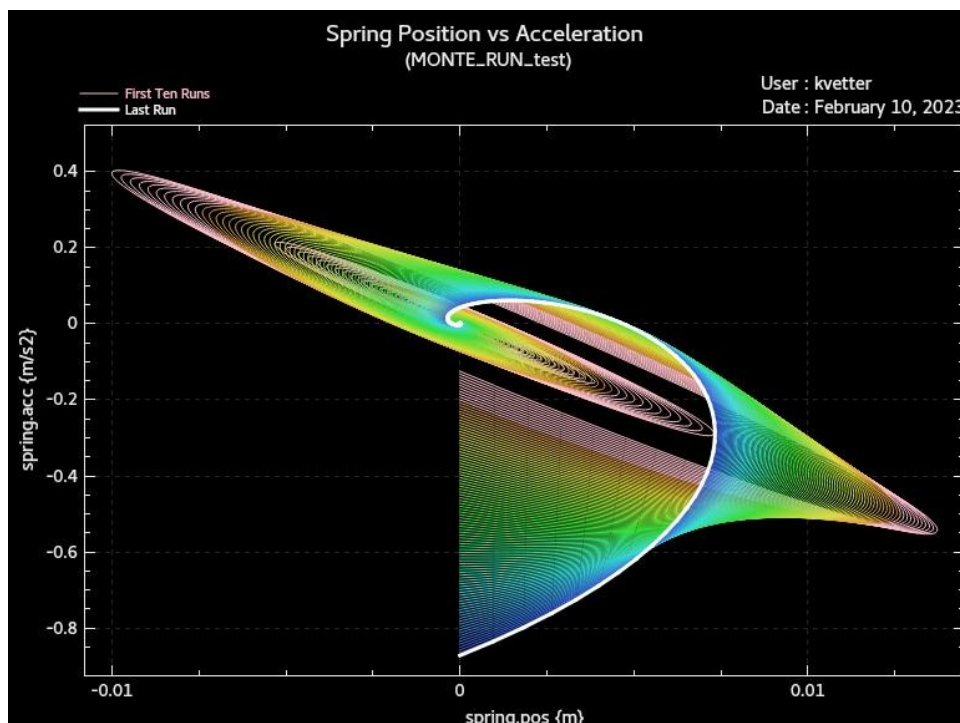

Groups

When plotting multiple RUNs, one can group RUN subsets together for coloring, styling and labeling. Grouping is done via regular expressions and ranges. For example, for the following set of RUNs: RUN_damp_1, RUN_damp_2, RUN_clamped_1, RUN_clamped_2, RUN_free_1, RUN_free_2, RUN_free_3, one could group “damped” and “clamped” together and group “free” RUNs with the following command:

```
UNIX Prompt> koviz RUN* \
                -g1 "damp|clamp" -l1 "Damped and Clamped" -c1 blue -ls1 plain \
                -g2 free -l2 Free -c1 green -ls1 x_thick_line
```

The following command uses ranges to group the first ten RUNs as pink and the 99th RUN as thick white:

```
UNIX Prompt> koviz MONTE_RUN_test -bg black -fg "#ededed" \
                -g1 "0,10" -c1 pink -l1 "First Ten Runs" \
                -g2 99 -c2 white -ls2 x_thick_line -l2 "Last Run" DP_pos_vs_acc
```



Video

Introduction

Koviz uses mpv as the backend video player. Any video that mpv can play e.g. *.mp4, *.avi can be tied to data.

Building Koviz with MPV Support

To build koviz with video support, the mpv development libraries must be installed. On CentOS7, this requires the “Nux Dextop” RPM repository. Once Nux is enabled, something like the following will install the mpv dependencies and build koviz with video.

```
% sudo yum install mpv-libs mpv-libs-devel mpv
% cd <koviz-repo>
% make distclean
% qmake-qt5
% make
```

Data Synchronization

Normally, video and data are alignable by a time offset. The time offset is given to koviz either on the commandline or in a file. Finding the time offset is a manual process unless the video and data were recorded together by a time synchronized system. For example, consider a GoPro camera recording a video of a person striking a loadcell with a hammer. The offset must be set to a value that aligns the hammer strike event in the video to the spike in the loadcell logged force data. Once a single event is synchronized, the video and data are completely synced – assuming the normal case where the video and data are real-time. Finding the exact video frame of the event that corresponds to the exact time in the data may not be possible. For instance, the hammer strike may not be seen in the video, only before and after the strike. In practice, this is okay, but is something to be aware of if the analysis requires that kind of fidelity.

Commandline -video and -videoOffset options

The simplest way to quickly tie a single video to data is to use the -video and -videoOffset options. For example:

```
% koviz RUN_loadcell -video hammerstrike.mp4 -videoOffset 12.345
```

Commandline -videoList option

The -videoList commandline option can be used to tie multiple videos to a single RUN. The -videoList option takes a comma separate list of video:offsets using a colon to separate the video file name from the offset e.g:

```
% koviz RUN_fun -videoList "Video/2023-03-22T0939_R02_Cam01_REC1_3.mp4:15.8450,
                             Video/2023-03-22T0939_R02_Cam02_REC2_3.mp4:15.6080,
                             Video/2023-03-22T0939_R02_Cam03_REC3_3.mp4:15.6060,
                             Video/2023-03-22T0939_R02_CamCL_REC4_3.mp4:15.6150,
                             Video/2023-03-22T0939_R02_CamF1_REC5_3.mp4:15.4820"
```

Session Files

A session file may work better if there is a need to save the video and offset for future use. Example session file (session_hammer):

```
Session – Hammer Video Sync
RUN: RUN_loadcell
PRODUCT: DP_forces
video: Videos/hammerstrike.mp4
videoOffset: 12.345
```

To run:

```
% koviz -session session_hammer
```

Video Directory Within RUN

Another way, when there are multiple RUNs tied to single videos, is to create a “video” directory (or customize 21) in the RUN dir. The video file, e.g. hammerstrike.mp4, is placed in the “video” directory. An optional file, explicitly called “video-offset.txt”, contains the video offset for the video. For the hammertime example, the directory structure would look like:

```
RUN_loadcell/
    loadcell_data.csv
    video/
        hammerstrike.mp4
        video-offset.txt (containing 12.345)
```

If the RUN directory is setup this way the following will bring up the data and the associated video:

```
% koviz RUN_loadcell
```

Multiple RUNs And Multiple Videos

Finally, to tie multiple videos per RUN, place a set of videos in the “RUN/video” directory (or customize 21) and create a file called “video-offsets”. The format of “video-offsets” is a comma separated list of video/offsets as shown in the example below. Using the video directory within a RUN directory technique, a set of RUNs can be tied to their associated videos. Once all RUNs have their associated videos in place, koviz can be launched with the entire set of RUNs. This makes it so one may hop from video to video by simply clicking on a curve of interest. Here is an example layout of two RUNs with multiple videos:

```
RUN_a/
  log_RddrSimhost_DtaMtaRelMot.trk
  video/
    video-offsets.txt (contents below)
    2023-03-13T1109_R05_Cam01_REC1_6.mp4
    2023-03-13T1109_R05_Cam02_REC2_6.mp4
    2023-03-13T1109_R05_Cam03_REC3_6.mp4
    2023-03-13T1109_R05_CamCL_REC4_6.mp4
    2023-03-13T1109_R05_CamF1_REC5_6.mp4
    2023-03-13T1109_R05_CamF2_REC6_6.mp4

RUN_b/
  log_RddrSimhost_DtaMtaRelMot.trk
  video/
    video-offsets.txt
    2023-03-22T0939_R02_Cam01_REC1_3.mp4
    2023-03-22T0939_R02_Cam02_REC2_3.mp4
    2023-03-22T0939_R02_Cam03_REC3_3.mp4
```

```
% cat RUN_a/video/video-offsets.txt:
```

```
2023-03-13T1109_R05_Cam01_REC1_6.mp4, 18.479
2023-03-13T1109_R05_Cam02_REC2_6.mp4, 18.236
2023-03-13T1109_R05_Cam03_REC3_6.mp4, 18.257
2023-03-13T1109_R05_CamCL_REC4_6.mp4, 18.207
2023-03-13T1109_R05_CamF1_REC5_6.mp4, 18.123
2023-03-13T1109_R05_CamF2_REC6_6.mp4, 18.112
```

Custom Video Directory Path

By default, koviz will search for a “video” directory in the RUN directory. If one would rather place the videos associated with the RUN data in another location, one can give koviz a relative path from the RUN to the video directory. There are two ways to set the video directory. First, one can use the commandline option `-videoDir`. Second, one can set the video directory path in the main UI window: Options > VideoDirectory. For example, if the data is structured like below, one would set the Option > VideoDirectory to “../Video” since the relative path from the directory with the sim data to the Video directory is “../Video”:

```
RUN_a/  
  DATA/  
    log_Mars_environment.trk  
    log_Mars_rover.trk  
  Video/  
    video-offsets.txt  
    video-1.mp4  
    video-2.mp4
```

Blender

Introduction

Koviz is able to synchronize its plots to Blender animations via a koviz Blender plugin. The start/end times of the koviz simulation run data are mapped to the begin/end frames of the Blender animation. After selecting a koviz curve, mouse movement updates koviz live time interactively. The koviz live time is sent to Blender while the mouse moves. Blender interpolates the koviz time to the animation frame and updates the Blender scene. The sync works from Blender to koviz as well. With mouse/keyboard focus on the Blender window, running the Blender animation will send koviz time updates and update koviz plot live time.

Blender Addon Setup

First, add koviz to Blender's addons. Example:

```
% cd $HOME/.config/blender/2.93/scripts/addons  
  
% ln -s <koviz-home>/python/koviz.py koviz.py
```

Animate Blender Default Cube

As an example, make an animation of Blender's cube:

1. Bring up blender:
% blender
2. Right click > Insert Keyframe > Location
3. Slide animation frame to 250
4. Move cube up 6: KeyPress gz6
5. Right click > Insert Keyframe > Location
6. Run animation by mouse clicking |< followed by >
7. Save as cube.blend
8. Close blender

Sync Koviz To Default Cube Animation

Blender argument order matters. The -P option must follow the model.

```
% blender cube.blend -P <koviz-home>/blender/koviz-hello-world.py  
# Should see "Listen for koviz connection"
```

Run koviz on the supplied position.csv file with 250 points. Note that time will go from 1 to 250 and map to the Blender frames. Position linearly moves from 0 to 6. This directly maps sim time to blender frames. It's not necessary to make this direct frame to time map, it just made for a nice example. In general, koviz maps the start Blender frame to first time stamp and last Blender frame to the last time stamp.

```
% koviz <koviz-home>/blender/cube-position.csv  
# Blender will say "Accept koviz connection!"
```

To see koviz time synced to the Blender animation:

1. In koviz, click position
2. Mouse select the curve
3. Move mouse to see Blender animation sync
!! If it does not sync, click off of koviz curve, reselect curve and try moving mouse again.

To see Blender animation sync to koviz:

1. Select curve in koviz (if not already done)
2. Click Blender window so Blender has focus
3. Run Blender animation by mouse clicking the play button >

Sync Blender To Koviz With ES's Vismo

The Engineering Structures group have tied their vismo Blender tool to koviz. If you are lucky enough to have vismo, here is an example. The -k option tells vismo to sync using koviz:

```
% cd <rddr-nds>
% . .Trick_user_profile ; # To put vismo.py in $PATH
% cd <rddr-nds>/sims/rddr/nds/SIM_nds
% trick-CP
% ./S_main* SET_checkout/SET_m1u/RUN_checkout_MaxJackknife/input.py
% vismo.py -k SET_checkout/SET_m1u/RUN_checkout_MaxJackknife
    # Listen for koviz connection...
% koviz SET_checkout/SET_m1u/RUN_checkout_MaxJackknife
    # Should see "Connected to host=127.0.0.1 port=64053!"
```

To see the koviz to Blender sync:

1. In the koviz search box, type "Fqu.daActuator"
2. Scroll over all 6 vars to coplot 6 actuators on single page
3. Select a curve and move mouse to see data synced to docking animation

To see the Blender to koviz sync:

1. In the Blender window click the |< to put the animation at the beginning
2. Press > to play
3. The koviz live time should update along with the animation
4. Pause the Blender animation by pressing the || pause button
5. Press the keyboard right and left arrow keys to step the animation and watch koviz update frame by frame

Live Trick TV Plots

Introduction

Koviz has the ability to not only plot Trick recorded data but can plot live Trick sim data as well. When koviz is given the `-trickport <port>` option, it connects to the Trick variable server on the specified port and creates a TV panel for plotting sim parameters from the SIE database, for example, to connect to a running sim on the localhost and port 4545:

```
% koviz RUN_realtime -trickport 4545
```

If the Trick sim is running on a different host, use the `-trickhost <host>` option.

Plotting TV Params

The parameters in the koviz TV panel parameter list are not expanded with their dimensions. For example, one might see a parameter called `ball.state.out.position`, when in reality there are two sim variables since position has two dimensions e.g.:

```
ball.state.out.position[0]
ball.state.out.position[1]
```

When one clicks a koviz-tv parameter, by default it makes a single plot on its own page for each of the expanded parameters. For example, if one clicks `ball.state.out.position`, two pages will be added with `ball.state.out.position[0]` on one page and `ball.state.out.position[1]` on the other page. If one holds the Ctrl-Key down, the expanded parameters will be added on individual plots, but on the same page. If the Alt-Key is held when clicking a parameter, the expanded parameters will be coplotted on the same plot. Finally, if one holds the Shift-Key down while optionally using Ctrl or Alt, the plots will be placed on a new page. In summary:

Plot expanded parameter on individual pages	Left Mouse Click parameter
Plot expanded parameter on different plots on same page	Ctrl-Key + Left Mouse Click parameter
Coplot expanded parameter on same plot	Alt-Key + Left Mouse Click parameter
Start a new plot page and honor optional Ctrl and Alt keys	Shift-Key + [Ctrl Alt] + Left Mouse click parameter

Changing Live Plot Domain

To change the plot domain (x-axis variable) one must select Options->EnableDragAndDrop, then drag a TV parameter onto the x-axis of the plot to change. By default the first dimension of the drag-n-dropped parameter is used when dropping a parameter onto the x-axis. Use the Ctrl-key when dragging and dropping for the second parameter of the expansion, and the Alt-key for the third. If one wants to use the 4th or higher, a DP file must be made with the x dimensioned parameter of choice.

Live DP Files

One may use the same DP files for both recorded data and live sim data. One can either specify the DP files on commandline or select the DPs from the DP panel when in live mode. The example below has koviz connect to the Trick variable server on port 4545 and plot the variables specified in the DP_tv.xml file.

```
% koviz RUN_realtime -trickport 4545 DP_tv.xml
```

Live Spring Plot Example

Koviz contains a spring sim example. To try the spring sim with live plotting, follow these steps:

1. Build the koviz spring sim

```
% cd <koviz-home>/sims/SIM_spring
% trick-CP
```
2. Launch the realtime spring sim (do not press start yet)

```
% ./S_main*.exe RUN_realtime/input.py &
```
3. Launch koviz and connect to the trick sim

```
% koviz -trickport 4545
```
4. Select params to plot
While holding the Ctrl key down, click:

```
spring.pos
spring.vel
spring.acc
```

5. Run sim by pressing “Start” button on sim control
6. Plots should update after pressing start

Profiling Trick Jobs

This is an example of how to profile a Trick sim using koviz' mini profiler. The example induces a couple sleeps in the Trick Ball L2 sim and shows how one would profile the sim.

Add Sleeps To Ball L2 Sim

```
% cd $HOME
% mkdir sims
% cd sims
% cp -r <trick>/trick_sims/Ball .
% cd Ball/models/ball/L2/src
% vi ball_floor.c
    # Add this #include to top of floor function
    #include <time.h>

    // Add 3ms and 7ms sleeps to body of floor function
    struct timespec ts3;
    ts3.tv_sec = 0;
    ts3.tv_nsec = 3 * 1000000; // 3 ms

    struct timespec ts7;
    ts7.tv_sec = 0;
    ts7.tv_nsec = 7 * 1000000; // 7 ms

    // Two sleeps
    nanosleep(&ts3, NULL);
    nanosleep(&ts7, NULL);
```

Run And View Sleep Effect

```
% cd $HOME/sims/Ball/SIM_ball_L2
% vi Modified_data/realtime.py
    # Turn frame log on
    trick.frame_log_on()
% vi RUN_realtime/input.py
    # Change stop time to 30 seconds
    trick.sim_services.exec_set_terminate_time(30.0)
% trick-CP
% ./S_main* RUN_realtime/input.py
% koviz RUN_realtime
```

See ball floor timing (should be a little over 10ms with possible spikes)

Modify and Build Profiler

```
% cd <koviz>/profiler
% ./clean.sh
% vi koviz.h
  1. Uncomment out #define TRICK
  2. Change KOVIZ_MAX_STATS to 2 (to time 2 sleeps)
  3. Change KOVIZ_NRECS 3000 (100Hz*30 second run)
% cc -I<trick>/include -c koviz.c
% ar cr libkoviz.a koviz.o
```

Add Profiler To Sim

```
% vi $HOME/sims/Ball/models/ball/L2/src/ball_floor.c
  # Add profiler header to ball_floor.c
  #include <koviz>/profiler/koviz.h

  # Add two profiler tags
  koviz_start("gtod", "koviz.sleep.3");
  nanosleep(&ts3, NULL);
  koviz_stop("gtod", "koviz.sleep.3");
  koviz_start("gtod", "koviz.sleep.7");
  nanosleep(&ts7, NULL);
  koviz_stop("gtod", "koviz.sleep.7");

% cd $HOME/sims/Ball/SIM_ball_L2
% vi S_overrides.mk
  # Add
  TRICK_USER_LINK_LIBS += <koviz>/profiler/libkoviz.a

# Rebuild sim with profiler
% trick-CP
```

Run Sim And See Profile Results

```
% ./S_main* RUN_realtime/input.py
% mv koviz.trk RUN_realtime
% koviz RUN_realtime
  # Put koviz|floor in search box and coplot
```