Demonstration of Vega-Video: Incorporating Video into the Grammar of Interactive Graphics

Dominik Winecki
winecki.1@osu.edu
The Ohio State University
Columbus, Ohio, USA

Arnab Nandi
nandi.9@osu.edu
The Ohio State University
Columbus, Ohio, USA

Abstract

We demonstrate *Vega-Video*, a tool for creating interactive visualizations combining video data with conventional data. The declarative Vega & Vega-Lite grammars enable performant data visualization and have made data exploration and presentation easier. Vega-Video supports three classes of techniques: synchronization, annotation, and transformation. Synchronization with playback state and annotating with overlays are established paradigms, now available in a declarative and portable form. For example, Vega-Video enables synchronizing plots with a video's timeline while annotating frames with bounding boxes over objects. Further, video transformation allows real-time updates to a video's content, such as dynamically updating video compilations in response to brushing operations. Vega-Video reconciles these three paradigms with the Grammar of Graphics, enabling interactive visualizations that tightly integrate video and conventional data. Combining these three video interaction techniques with existing data interaction methods greatly reduces the friction of analyzing and exploring video datasets.

CCS Concepts

• **Human-centered computing → Visualization systems and tools**.

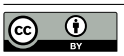
Keywords

video, visualization, multimodal data exploration

ACM Reference Format:
Dominik Winecki and Arnab Nandi. 2026. Demonstration of Vega-Video: Incorporating Video into the Grammar of Interactive Graphics. In *Companion of the International Conference on Management of Data (SIGMOD Companion '26), May 31-June 05, 2026, Bengaluru, India.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3788853.3801588

1 Introduction

Advancements in databases, computer vision, and machine learning have made *video* data more valuable than ever. However, human exploration and analysis of video datasets remains difficult. Video is rarely a standalone data modality. Multimodal data pipelines often store accompanying time series data, tabular data, or model inference data. This necessitates interfaces that combine both conventional data visualization and video data visualization.



SIGMOD Companion '26, Bengaluru, India

ACM ISBN 979-8-4007-2450-3/2026/05
https://doi.org/10.1145/3788853.3801588

While conceptually simple, building hybrid video/data visualizations requires navigating the performance constraints of video. Conventional visualizations can store entire datasets in memory and render hundreds of times per second. Video data is accessed differently, video playback state resists rapid changes, and the web ecosystem's approach to video makes even simple synchronization tasks difficult to get right.

While bespoke applications have been created for visualization within specific domains, a generalized approach for creating interactive video/data visualizations remains unexplored. We demonstrate *Vega-Video*, which addresses this gap. Vega-Video extends Vega's declarative grammar to support three classes of operations: video synchronization, annotation, and transformation. Synchronization is performed via signals that follow semantic intent rather than direct execution, giving the appearance of a single unified state. Annotation is analogous to Vega's *marks* concept. Video transformation is modeled as a video-specific Vega transform implemented as symbolic operations on Video on Demand (VOD) manifest files. A declarative specification defines a visualization linked to one or more video players. At runtime, Vega-Video transparently binds Vega's dataflow graph to the video players' state. This enables portable, performant interactive visualizations that combine conventional and video data, making multimodal data exploration faster and easier.

2 Related Work

Wilkinson's Grammar of Graphics [15] is foundational to modern data visualization systems. Two of the most successful implementations are Vega [11] and Vega-Lite [10]. Vega uses a JSON-based specification grammar and a dataflow graph to propagate interaction state. This state is represented as signals, which are updated by events. Vega-Lite is a higher-level specification grammar that compiles to Vega. Vega-Lite makes common visualization patterns much easier to represent, specifically brushing and linking, which are unwieldy in their pure-Vega representation. Vega-Video supports both Vega and Vega-Lite specifications.

Many systems perform some form of video/data visualization, which we group by class. Synchronization, the most common, is used primarily for time series visualization. ChronoViz [5] is built for time-coded information. BEDA [6] is similar for physiology research, and MUVTIME [14] for behavioral sciences. The next class, annotation, is ubiquitous in computer vision, generally as tools for labeling objects with bounding boxes [2, 3, 12, 13]. However, in practice, annotating videos is generally done with OpenCV [1] or Supervision [9]. Finally, video transformation is the most abstract and least common. Early declarative video editing systems, such as DIVA [7], provided a "stream algebra" for annotating and rearranging video clips. Our prior work, V2V [16], is the first to

```
"players": [                                    ①
  { "name": "main",
    "sources": [{ "src": "data/video.mp4"}]}
],
"signals": [
  { "name": "@main.time",                       ②
    "on": [{
      "events": "@seekSurface:pointerdown,
      ↳   @seekSurface:pointermove[event.buttons]",
      "update": "invert('x', clamp(x(), 0, width))"}]}
],
"marks": [
  { "type": "rect",
    "name": "seekSurface",
    "encode": {
      "enter": { "fill": { "value": "transparent" } },
      "update": {
        "x": { "value": 0 }, "x2": { "signal": "width" },
        "y": { "value": 0 }, "y2": { "signal": "height" }}}},
  { "type": "line", /* blue line mark omitted */ },
  { "type": "rule",
    "encode": { "update": {
        "strokeDash": {"value": [6, 4]},        ④
        "x": {"scale": "x", "signal": "@main.itime"},
        "y": {"value": 0}, "y2": {"signal": "height"}}}},
  { "type": "rule",
    "encode": { "update": {
        "stroke": {"value": "red"},             ③
        "x": {"scale": "x", "signal": "@main.time"},
        "y": {"value": 0}, "y2": {"signal": "height"}}}}]
```

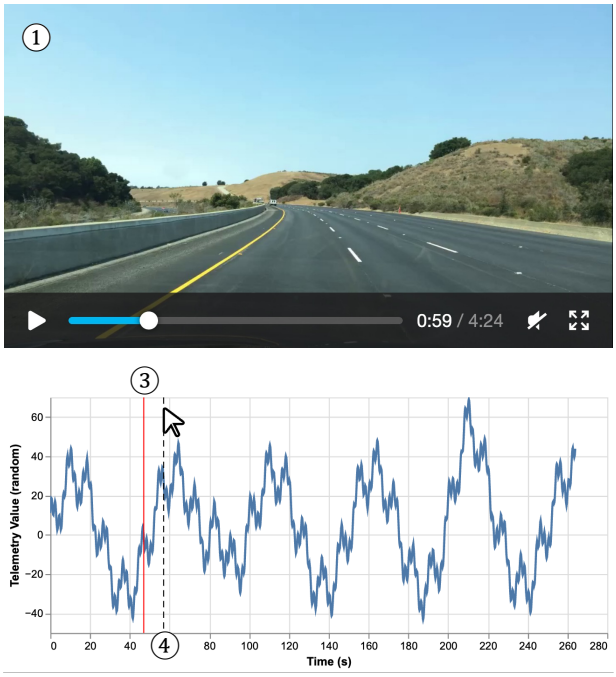
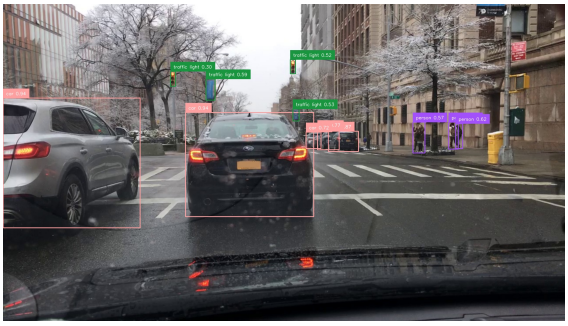


Figure 1: A partial Vega visualization syncing time series data with a video (①). Clicking or dragging on the plot updates the video's time (②). A red playback cursor, at 47s, displays the time of the currently presented frame (③). A dashed playback cursor, at 59s, displays the frame a user is targeting (④). Video frames in this and all subsequent figures from the BDD100K dataset [17] © 2018 The Regents of the University of California.

a) Annotated video frame



b) Python + Supervision annotation code

```
frame, dets = ...
dets = dets[dets.confidence >= 0.30]
dets = dets[dets.class_name.isin(
                    ["car", "person", "traffic light"])]
labels = [f"{class_name} {confidence:.2f}"
              for class_name, confidence in
                zip(dets["class_name"], dets.confidence)]
frame = BoundingBoxAnnotator().annotate(frame, dets)
frame = LabelAnnotator().annotate(frame, dets, labels=labels)
```

c) Vega-Video video annotation spec.

```
"marks": [
  { "type": "boundingbox",
    "transform": [
      { "type": "filter",
        "field": "class_name",
        "oneOf": ["person", "car", "traffic light"]}
      { "type": "filter",
        "field": "confidence", "gte": 0.30}],
    "encode": { "update": {
        "xyxy": {"field": "xyxy"},
        "class_id": {"field": "class_id"},
        "frame_index": {"field": "frame_index"}}}},
  { "type": "label",
    "transform": [
      { "type": "filter",
        "field": "class_name",
        "oneOf": ["person", "car", "traffic light"]}
      { "type": "filter",
        "field": "confidence", "gte": 0.30}],
    "encode": { "update": {
        "xyxy": {"field": "xyxy"},
        "class_id": {"field": "class_id"},
        "label": {
          "signal": "datum.class_name + ' ' 
                    ↳ + format(datum.confidence, '.2f')"},
        "frame_index": {"field": "frame_index"}}}}]
```

Figure 2: Frame annotation specification using our proposed grammar (c), as well as the equivalent Python + Supervision code (b), for drawing bounding boxes and a custom-formatted text label on objects in a video frame (a).

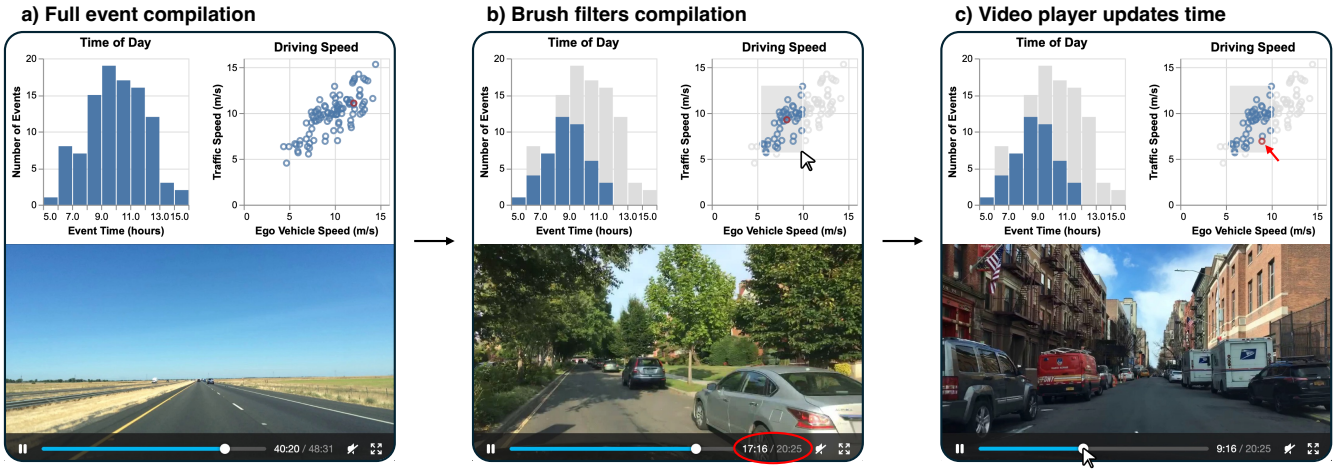
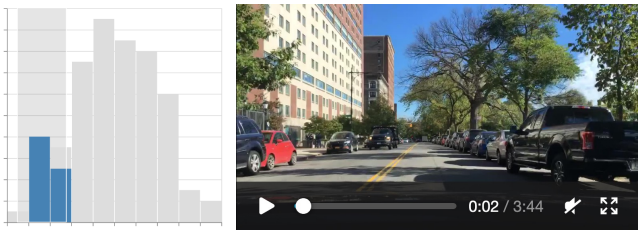


Figure 3: Brushing and linking with a video compilation. A brush selection (b) updates both a linked plot and a linked video compilation. A marker on the scatter plot follows the currently displayed event in the compilation (c).




```
{
  "data": /* omitted */
  "layer": [
    { "mark": "bar",
      "params": [
        { "name": "hist",
          "select": { "type": "interval",
                      "encodings": ["x"] }}],
      "encoding": {
        "x": { "field": /* val */, "type": "quantitative",
               "bin": { "maxbins": 15 } },
        "y": { "aggregate": "count" },
        "color": { "value": "#dddddd" }}},
    { "mark": "bar",
      "transform": [{ "filter": { "param": "hist" } }],
      "encoding": {
        "x": { "field": /* val */, "type": "quantitative",
               "bin": { "maxbins": 15 } },
        "y": { "aggregate": "count" },
        "color": { "value": "steelblue" }}}]
  "players": [
    { "name": "main",
      "playlist": {
        "transform": [{ "filter": { "param": "hist" } }]}}]
}
```

Figure 4: A Vega-Lite specification for brushing a histogram and linking a video compilation. A `"players"` object is configured with a filter.

explore declarative video editing as a performance optimization, efficiently synthesizing video results from queries. Video Lens [8] enables quick viewing of specific clips, including via a brushing-and-linking approach.

3 Visualization With Vega-Video

Vega-Video specifications are a superset of the Vega/Vega-Lite grammar. Vega-Video binds to video players via a top-level `"players"` attribute. An example is shown at ① in Figure 1. Similar to Vega-Lite, Vega-Video compiles specifications to pure Vega.

3.1 Synchronization

A video player's state is exposed via signals, as shown in Figure 1. Signals have intuitive names, such as `@player.duration` and `@player.playing`, the latter of which can be set to play or pause the video. There are *two* signals for representing a video player's time:

- **`@player.time`** is the *presented* time of the current frame
- **`@player.itime`** is the *intended* next-frame time

The `@player.time` signal is for frame-exact synchronization with a video, reflecting the time of the currently displayed frame, but is subject to the slow update time of a video player during scrubbing (dragging the playhead across the timeline). `@player.itime` provides the instantaneous desired video position, but may not correspond to the currently displayed frame during scrubbing. These two time values can be seen in Figure 1 at ③ and ④, respectively. Scrubbing is one of the primary interactions in video/data visualizations [4], so Vega-Video allows explicit control over this behavior.

Figure 1 shows a visualization drawing these two time values on a time series line plot. Additionally, a seek surface (②) allows a user to drag across the chart to jump to the corresponding location in the video. The state is shown during a scrub operation via a mouse-drag; otherwise, the two time values would coincide.

3.2 Annotation

Vega-Video supports overlaying annotations on videos. Annotating videos primarily involves using computer vision results, such as drawing bounding boxes around objects. Vega-Video adopts Supervision's [9] detection format and annotators, a widely-used standard in computer vision. This equivalence is shown in Figure 2.

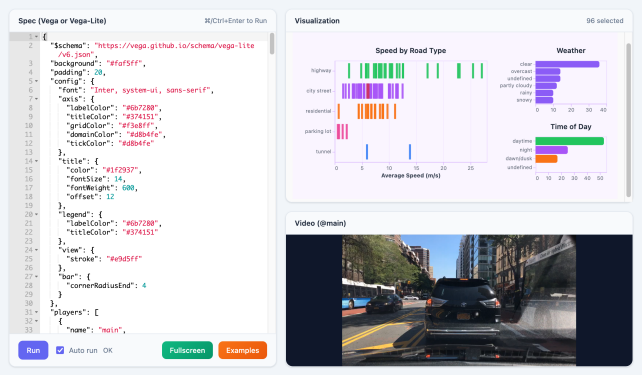


Figure 5: Vega-Video Playground, our editor for iterating on video visualizations. On the left is a specification editor, and on the right is the visualization display and video player.

Annotation has a natural parallel to Vega's marks when treating the video as a discrete axis. Each annotation type, such as a bounding box or label, is analogous to a mark type.

3.3 Transformation

Vega-Video supports *transforming* videos in real time. Figure 3 shows brushing-and-linking, where selecting a region in one view filters linked views, with a video compilation. A collection of video events is spliced together to form a compilation. These events can be filtered via brushing, and other linked plots and videos update in response. As more events are filtered out, the video becomes shorter. In our demonstration, these videos update in approximately 150–250 ms. In addition, the current event in the video is highlighted on the scatter plot, always following the currently displayed event.

Video transformation is specified through video-specific transforms. These include filtering, clipping, sorting, and erosion/dilation for managing short events. A minimal specification for video transformation is shown in Figure 4.

Vega-Video implements this real-time video transformation by exploiting VOD protocols. Each event is treated as a separate VOD stream, and operations are applied to each stream's manifest file. Finally, all the streams are spliced together to form the final transformed video stream, which is then passed to a VOD driver.

3.4 Demonstration Environment

Our demonstration consists of showing the visualizations from Figures 1 to 5 in action. We demonstrate Vega-Video through its playground website, shown in Figure 5, which is available to try on laptops and smartphones. This environment allows rapid iteration when developing video/data visualizations, and comes preloaded with examples. Our demo video is available on YouTube.

4 Conclusion

We demonstrate Vega-Video, an extension for Vega/Vega-Lite that incorporates video into the grammar of interactive graphics. We present interactive visualizations that use the three classes of video data visualization: synchronization, annotation, and transformation. Each is integrated into Vega's Grammar of Graphics-based specification language, allowing idiomatic interactions across video and conventional visualizations. Most notably, we demonstrate novel interactions for video transformation, including applying filters to video event compilations in real time. Vega-Video is available as open source software at https://github.com/ixlab/vega-video.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1910356 and the NSF OAC 2118240 Imageomics Institute award. We also acknowledge neurologist Pietro Mazzoni, whose experiences working with video data in his research motivated parts of this work.

References

[1] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
[2] A. Dutta, A. Gupta, and A. Zisserman. 2016. VGG Image Annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/.
[3] Abhishek Dutta and Andrew Zisserman. 2019. The VIA Annotation Software for Images, Audio and Video. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. ACM, New York, NY, USA, 4 pages. doi:10.1145/3343031.3350535
[4] Adam Fouse. 2013. *Navigation of Time-Coded Data.* Ph. D. Dissertation. University of California, San Diego. https://escholarship.org/uc/item/6nx3x60t
[5] Adam Fouse, Nadir Weibel, Edwin Hutchins, and James D. Hollan. 2011. ChronoViz: a system for supporting navigation of time-coded data. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. Association for Computing Machinery, 299–304. doi:10.1145/1979742.1979706
[6] Jennifer Kim, Melinda Snodgrass, Mary Pietrowicz, Karrie Karahalios, and Jim Halle. 2013. BEDA: Visual analytics for behavioral and physiological data. In *Workshop on Visual Analytics in Healthcare. Washington DC*. 23–27.
[7] Wendy E. Mackay and Michel Beaudouin-Lafon. 1998. DIVA: exploratory data analysis with multimedia streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '98)*. ACM Press/Addison-Wesley Publishing Co., USA, 416–423. doi:10.1145/274644.274701
[8] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2014. Video lens: rapid playback and exploration of large video collections and associated metadata. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 541–550. doi:10.1145/2642918.2647366
[9] Roboflow. [n. d.]. *Supervision.* https://github.com/roboflow/supervision
[10] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2017). doi:10.1109/TVCG.2016.2599030
[11] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. 2014. Declarative Interaction Design for Data Visualization. In *Proc. ACM User Interface Software & Technology (UIST)*. doi:10.1145/2642918.2647360
[12] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a-andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. 2020. opencv/cvat. doi:10.5281/zenodo.4009388
[13] Snehesh Shrestha, William Sentosatio, Huiashu Peng, Cornelia Fermuller, and Yiannis Aloimonos. 2023. FEVA: Fast Event Video Annotation Tool. *arXiv preprint arXiv:2301.00482* (2023).
[14] Emanuel Sousa, Tiago Malheiro, Estela Bicho, Wolfram Erlhagen, Jorge Santos, and Alfredo Pereira. 2016. MUVTIME: A Multivariate Time Series Visualizer for Behavioral Science. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016) - IVAPP*. INSTICC, SciTePress, 165–176. doi:10.5220/0005725301650176
[15] Leland Wilkinson. 2005. *The Grammar of Graphics.* Springer-Verlag, New York. doi:10.1007/0-387-28695-0
[16] Dominik Winecki and Arnab Nandi. 2024. V2V: Efficiently Synthesizing Video Results for Video Queries. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 5614–5621. doi:10.1109/ICDE60146.2024.00449
[17] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2633–2642. doi:10.1109/CVPR42600.2020.00271