VideoMap: Video Editing in Latent Space

David Chuan-En Lin[1], Fabian Caba Heilbron[2],
Joon-Young Lee[2], Oliver Wang[2], Nikolas Martelaro[1]
[1]Carnegie Mellon University, [2]Adobe Research
[1]chuanenl@cs.cmu.edu, nikmart@cmu.edu, [2]{caba, jolee, owang}@adobe.com

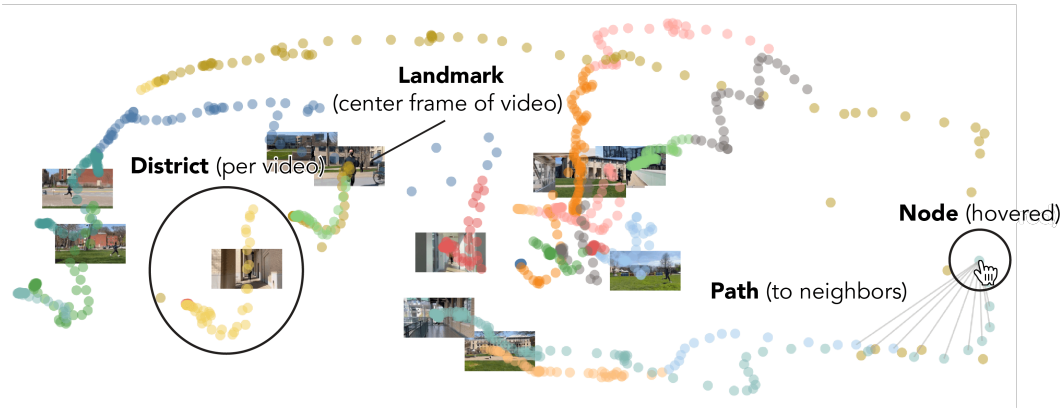

Figure 1: VideoMap maps video frames into a 2D latent space. We introduce map-inspired metaphors (node, path, district, and landmark) to help users navigate the map.

1 Introduction

Video has become a dominant form of media. However, video editing interfaces have remained largely unchanged over the past two decades. Such interfaces typically consist of a grid-like asset management panel and a linear editing timeline [1]. When working with a large number of video clips, it can be difficult to sort through them all and identify patterns within (e.g. opportunities for smooth transitions and storytelling). In this work, we imagine a new paradigm for video editing by mapping videos into a 2D latent space and building a proof-of-concept interface.

2 VideoMap

Creating the Latent Space. VideoMap is based on a latent space representation of video frames. Specifically, we encode video frames into vector representations such that the latent space is meaningful for a particular video editing task. For example, we can create a latent space map to help editors perform shape match cuts, a type of cut which cleverly links together two scenes that contain similar shapes to create a smooth transition (see example from Kubrick's 2001: A Space Odyssey). To achieve this, we train a convolutional autoencoder with the objective of generating a foreground mask from a video frame (Figure 3). We use DeepLabv3 [2], an image masking model, to create the ground truth masks and compute the reconstruction loss with the generated masks to optimize the autoencoder. After training, we feed the video frames through the autoencoder's encoder and extract the `z` vector from the bottleneck layer before the decoder. Due to the foreground mask generation objective, the bottleneck vector `z` implicitly encodes valuable shape-related information. Frames that contain similar shapes are in similar regions on the projected latent space (Figure 2).

Developing Swappable Lenses. We call the different methods of encoding video frames into meaningful vector representations as different *lenses*. In the example above, we create a `shape` lens.

Preprint. Under review.

Under the `shape` lens, editors can easily identify shape-based match cut opportunities. Similarly, one may develop new lenses by finding other ways of encoding useful `z` vectors for other use cases. For example, we can use an image colorization autoencoder [3] to create a `color` lens (latent space organized by color) and use CLIP [5], a neural network that has learned semantic concepts, to create a `semantic` lens (latent space organized by semantics). Rather than creating the "best map" in a one-size-fits-all manner, we allow users to flexibly swap between different lenses and reconfigure VideoMap according to different criteria.

Navigating with Nodes, Paths, Districts, and Landmarks. We implement four map-inspired navigational metaphors to help users intuitively navigate through the map. The four elements are inspired by Lynch's The Image of the City [4]. In this seminal work, Lynch conducted a five-year study to uncover the elements people use to form mental maps of a city.

1. The *node* is a point (i.e. an encoded video frame) that the user hovers over with the cursor (Figure 4a). We follow Schneiderman's details-on-demand mantra [6] by expanding the node's detailed information, visualizing the node's corresponding video frame and drawing outward paths.

2. *Paths* are lines that connect from a user-selected node to its neighboring points (Figure 5b). When a user hovers over a node, we display ten paths that start from the node and connect to the ten closest neighboring points, which are the points with the smallest cosine distances to the node.

3. *Districts* are groups of points with common characteristics (Figure 5a). In Figure 1, the map portrays multiple videos and we bin the points that belong to the same video into a district. We color code each unique district to visually separate them. Thus, the user can see whether the paths from a node connect to neighboring points of the same video or of different videos. We also consider alternative ways of defining districts, such as through common semantics, in Section 3.

4. *Landmarks* are reference markers on the map. We visualize the corresponding video frames of some points on the map as landmarks. With landmarks, the user can obtain an overview of the map at a glance (Figure 5b). In Figure 1, we generate one landmark for each district by visualizing the video frame of the point closest to the district's centroid. We also explore allowing the user to define custom landmarks in Section 3.

3 Applications

Match Cuts. The user can uncover smooth match cut transition opportunities from a large collection of videos by identifying *paths* that connect across two districts (i.e. two videos). Given a collection of videos from an author's weekend in San Francisco, we demonstrate three types of match cuts with *swappable lenses*: (1) shape match cut with the `shape` lens (Figure 6) (see example result), (2) color match cut with the `color` lens (Figure 7) (see example result), and (3) semantic match cut with the `semantic` lens (Figure 8) (see example result).

Video Summarization. The user can generate a summary video of a longer video by creating *semantic districts*. We create the semantics districts with k-means clustering on a map with the `semantic` lens (each semantic district represents a main event of the video). We then take the centroid frames of each district to create a summary video (Figure 9) (see example result).

Semantic Search. The user can perform semantic image search within a video by creating *custom landmarks*. Given a user-specified query image, we can project it onto a map with the `semantic` lens (i.e. adding a new point). The user can hover over points around the custom landmark to inspect semantically-close neighbors (Figures 10 and 11) (see example results).

4 Conclusion

In this paper, we explore the possibility of video editing in a 2D latent space map. We introduce the concept of swappable lenses and map-inspired navigational metaphors to build a proof-of-concept interface. We then illustrate the design space of VideoMap with three potential applications. For future work, we hope to investigate how editors may use VideoMap in their editing tasks and expand on the concept of swappable lenses, such as designing a process for people to build their own lenses.

References

[1] Get to know the Premiere Pro interface. URL https://helpx.adobe.com/premiere-pro/how-to/overview-interface-premiere-cc.html.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6837–6845, 2017.

[4] Kevin Lynch. *The image of the city*. MIT press, 1964.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[6] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.

A VideoMap with the `Shape` Lens

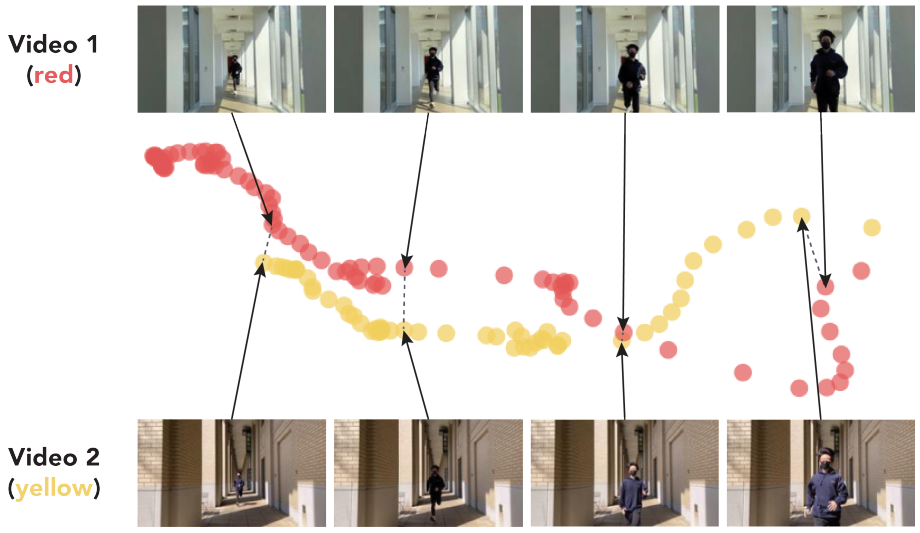

Figure 2: Two videos in VideoMap with the `shape` lens. We see that video frames with similar shapes are relatively close to each other. Some example shape-based match cut opportunities are shown by the dashed lines.

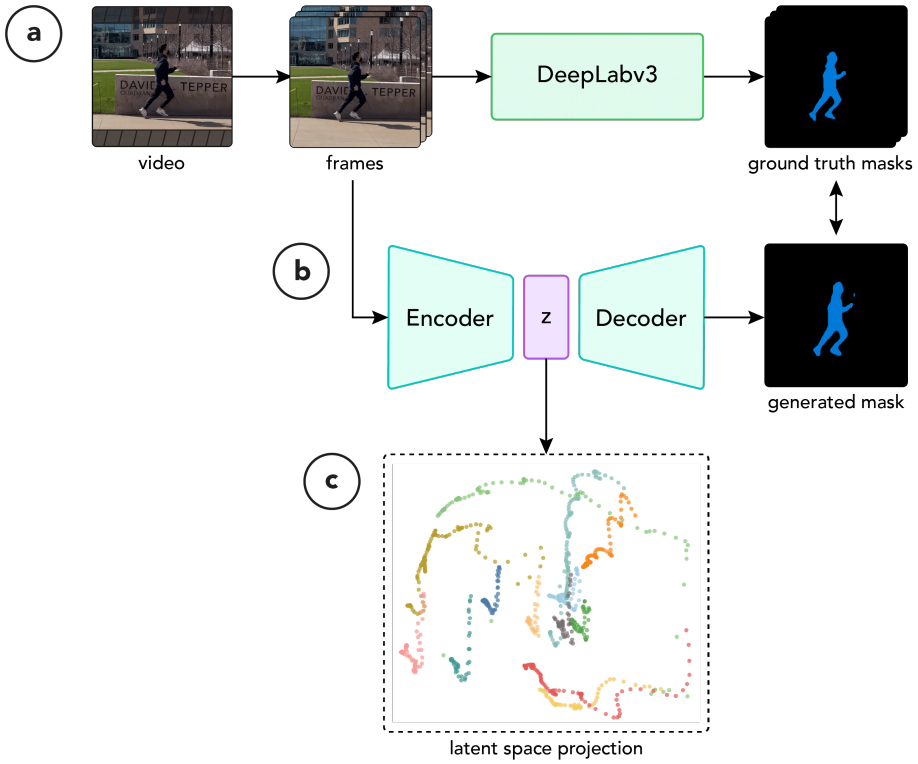

Figure 3: The pipeline for the `shape` lens. We first generate ground truth masks with DeepLabv3 [2] (a). We then train an autoencoder to reconstruct the masks from frames (b). Finally, we take the bottleneck vectors to create the latent space projection (c).

B Map Metaphors

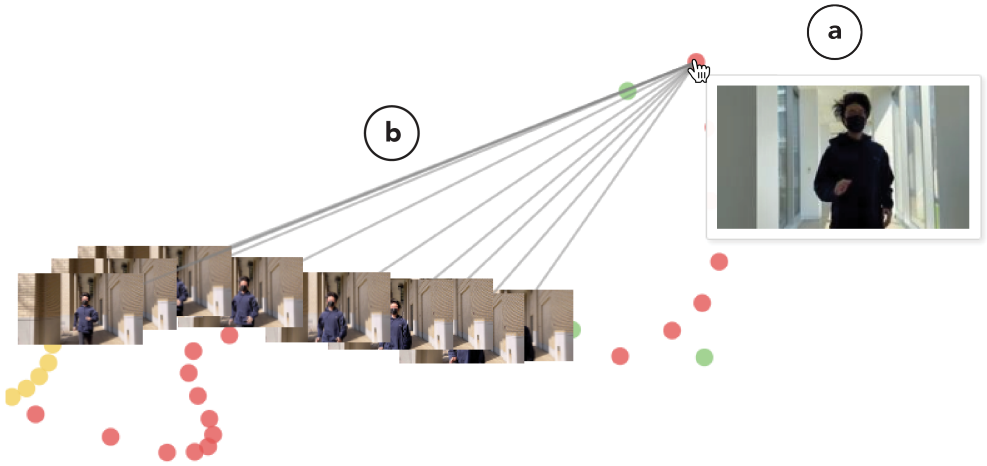

Figure 4: As the user hovers over a node (a), we visualize its corresponding video frame and draw ten paths to neighboring points (b).

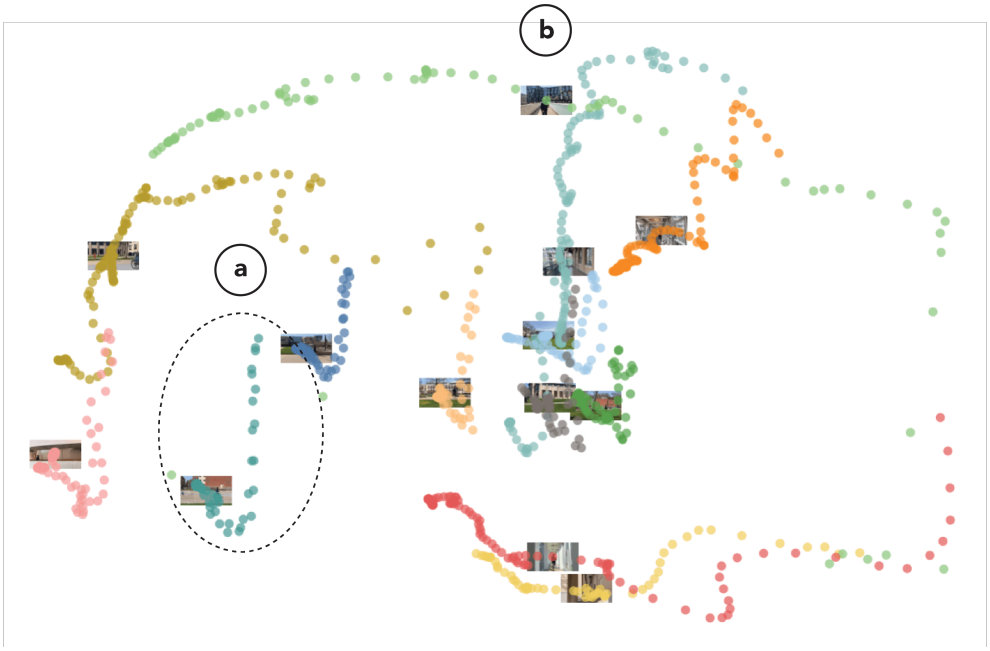

Figure 5: Each district (color coded) represents the points that belong to the same video (a). We visualize the centroid video frame of each district as the landmark (b).

C Example Applications

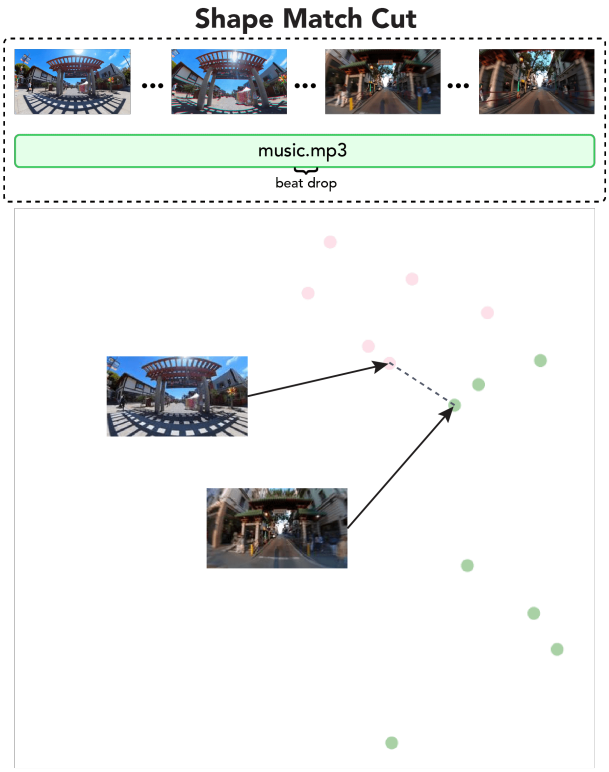

Figure 6: Under the `shape` lens, we can follow a path that connects across two districts to perform a shape match cut. See example result with music added.

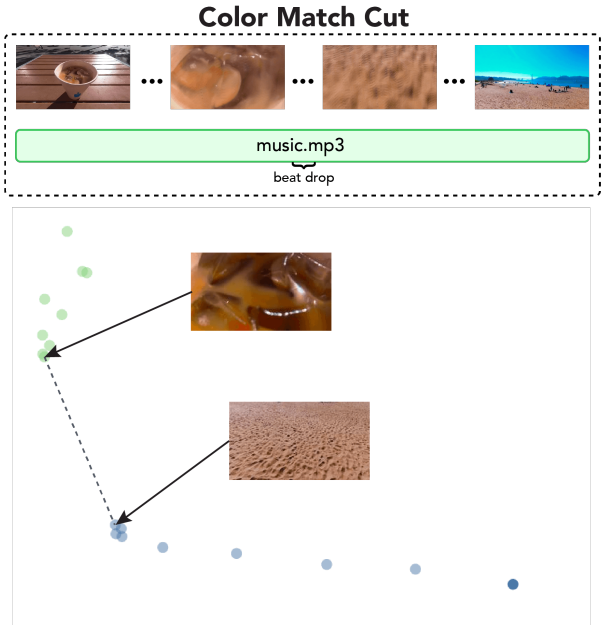

Figure 7: Under the `color` lens, we can follow a path that connects across two districts to perform a color match cut. See example result with music added.

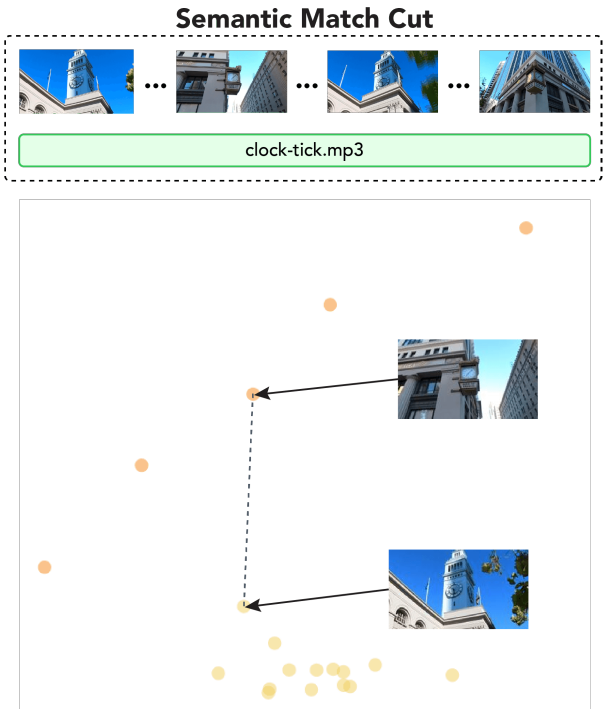

Figure 8: Under the `semantic` lens, we can follow a path that connects across two districts to perform a semantic match cut. See example result with music added.

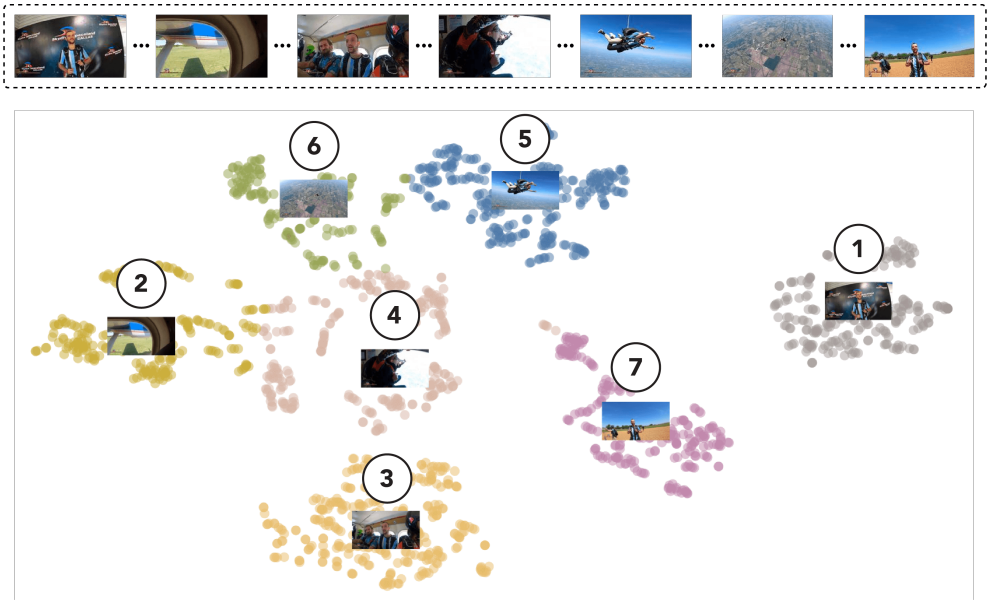

Figure 9: Under the `semantic` lens, we can create semantic districts by clustering neighbors. By taking a centroid sample from each district, we can create a summary video.

Figure 10: Under the `semantic` lens, we can perform semantic image search by projecting a custom image landmark onto the map and viewing its neighbors. The figure depicts an image query of skydiving freefall.

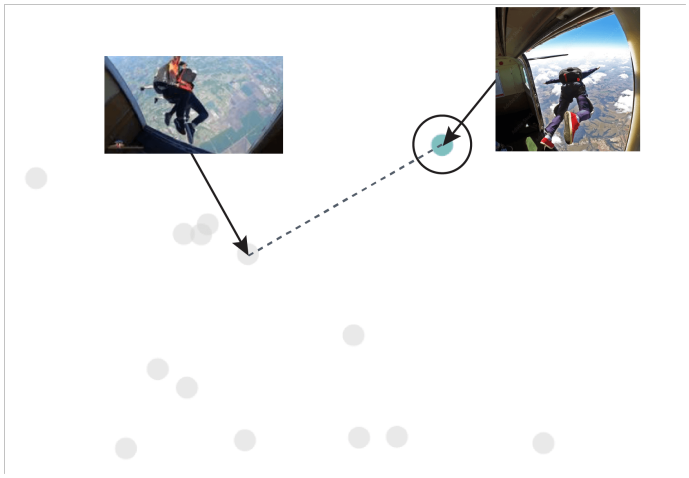

Figure 11: The figure depicts an image query of jumping out the plane under the `semantic` lens.