AR-Video Query

WOOJIN KO, University of California, Berkeley, USA
JAMES SMITH, University of California, Berkeley, USA
BJÖRN HARTMANN, University of California, Berkeley, USA

Insert abstract at the end

CCS Concepts: • **Human-centered computing → Mixed / augmented reality**; *Graphical user interfaces*; • **Information systems → Video search**.

Additional Key Words and Phrases: augmented reality, information retrieval, spatial interaction techniques, video query

1 Introduction

The amount of videos being recorded is increasing substantially. Roughly 50% of people in the world own and use smartphones daily[1], most of which come equipped with cameras capable of recording videos. Over 500 hours of video are uploaded to the website YouTube every minute[2]. Naturally, more than two decades of research has delved into building software to search within videos for specific content. This concept of capturing, storing, and navigating video content was proposed by Low and Jain[8] as one of the three grand challenges in multimedia computing. The video queries specifically can be classified into 3 categories of search[9]:

1) Textual Keyword (searching by keyword query for manually annotated video segments with matching text)
2) Visual Example (searching by image or sketch for visually "similar" video segments)
3) Concept (searching by semantic concept, commonly generated by machine learning, for related video segments).

Videos are sequences of images with timestamps of when those images are recorded. Images are made up of a two-dimensional (2D) array of pixels, also known as a single frame, and each pixel is commonly described by 3 intensities of red, green, and blue values. Much of the existing literature on searching for content in video focuses on first taking in such 2D videos as input and then extracting higher-level concepts from this data via various post-hoc computer vision techniques. Examples of common applications include classification, tracking, and 3D reconstruction.

At the same time, augmented reality (AR) applications have begun to take a hold in the smartphone market. Smartphones provide a type of AR experience commonly coined as video see-through AR, which superimposes virtual information upon a live video view of the real scene. The virtual objects are registered to the real scene by tracking the phone camera's physical motions and positioning a calibrated virtual camera that renders the virtual objects in the correct location. To do so, AR-capable smartphones combine video data with other types of sensor data to track the 3D position of the camera, scene, and sometimes scene geometry, in real-time.

Depth sensors are a class of devices capable of producing 3D point clouds of the local environment. This technology comes in many forms, such as stereo camera pairs[3], infrared projection[4], time of flight[5], and light detection and

[1]https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/
[2]https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/
[3]https://www.stereolabs.com/zed-2/
[4]https://www.intelrealsense.com/
[5]https://www.sony-depthsensing.com/

Authors' addresses: Woojin Ko, woojin_ko@berkeley.edu, University of California, Berkeley, USA; James Smith, james.smith@berkeley.edu, University of California, Berkeley, USA; Björn Hartmann, University of California, Berkeley, USA, bjoern@eecs.berkeley.edu.

ranging (LIDAR). Recently, LIDAR has been added to newer smartphones[6] to enable AR tracking at even higher fidelity. Smartphones equipped with LIDAR are able to produce a 3D mesh of the environment that approximates the local space's size and shape. This 3D reconstruction of the environment is produced by feeding video and sensor information on the AR device into a Simultaneous Localization and Mapping algorithm (SLAM)[10].

With all this in mind, we propose that by saving AR tracking information and grounding each video frame to a point in space, we can enable new ways of searching for content in videos. Spatial information can be saved every time a frame of video is recorded. This spatial information includes the camera's intrinsic (focal length, principal point, image size) and extrinsic (world position, rotation) parameters, as well as a reconstructed 3D model of the environment. Concurrently, temporal information comes in the form of video frames that are associated with time stamps. We call the artifact of joining recorded AR information and associated video frames an "AR-Video". We can develop interactions on AR-Videos to perform spatio-temporal queries across the entire length of one or more videos recorded in a given space. The final 3D model of the environment can also be used to align two or more recorded videos in space using the Iterative Closest Point algorithm [11], allowing us to process video taken from multiple points in time and space. We built an interactive system called AR-Video Query to explore the utility and applicability of performing spatio-temporal queries on videos with recorded AR tracking.

Why would someone want to make spatio-temporal queries? We will consider two possible applications.

First, we consider searching for information in sousveillance applications. "Sousveillance" shares common ground with "surveillance" in that videos are being recorded, but differs in that individuals, as opposed to governments, own and control the video cameras. The sousveillance model embodies a bottom-up approach where citizens can shed light on abuses of power by figures of authority [7]. In recent history, there have been many examples of social events such as police brutality and mass violence which have been captured in this way via multiple users' smartphones. Often, these events tend to be recorded over a series of short videos from many different perspectives. As a result, if one wanted to find a single point in space where a particular event might be occurring, one would find it quite difficult and time-consuming to do so across the entire set of video recordings of that event. We hypothesize that a spatio-temporal query against a 3D model of the environment in the event of interest would allow users to easily pinpoint that location within that set of videos.

Second, it is fairly common for people training convolutional neural networks to need to generate sets of training examples. This training data comes in the form of images with the areas of interest outlined with a bounding box. It is common to need to generate a set of training data that has a large variety of examples. A machine learning practitioner could use a spatio-temporal query to efficiently generate a set of training data across a whole sequence of videos.

A spatio-temporal query can be performed through either a spatial or temporal interaction. Spatial interactions can produce temporal results, and temporal interactions can produce temporal and spatial results. This project outlines one such spatial interaction, in which the user is able to draw a paint stroke over the objects that they are interested in querying for.

2 Related Work

We draw from related works in the fields of video querying, sousveillance, and spatial interaction techniques.

[6]https://developer.apple.com/augmented-reality/arkit/
[7]https://en.reset.org/knowledge/sourveillance-11152016

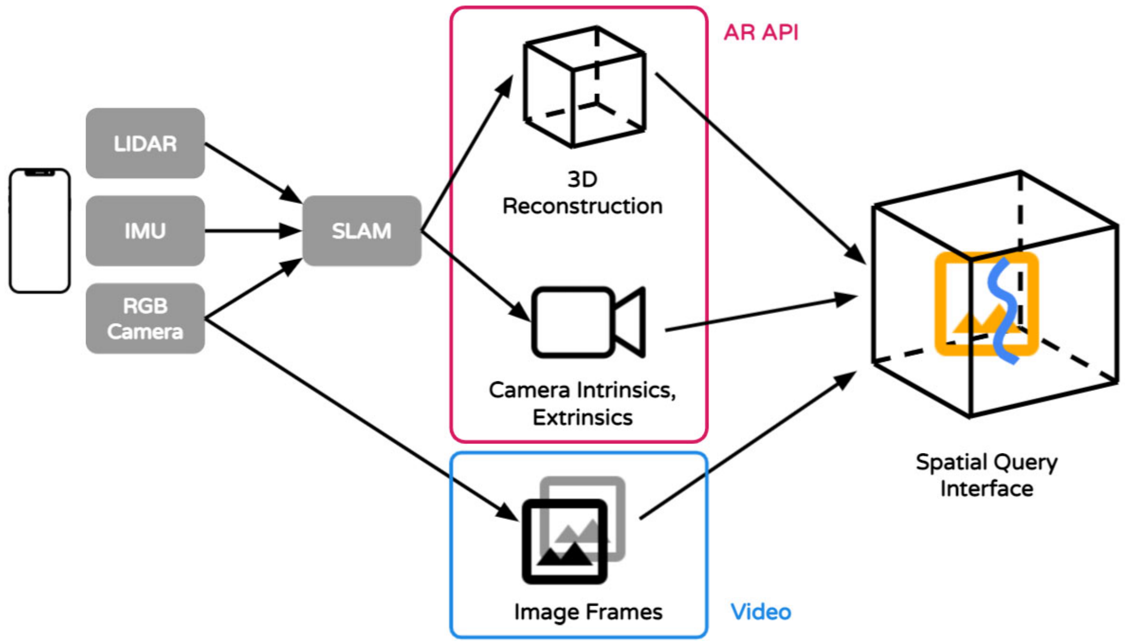

Fig. 1. Conceptual overview of the AR-Video Query data process. All data is extract from the iOS ARKit API using a custom video recording application.

Querying Videos

Video files can contain gigabytes of information and searching through this information has been a challenge in multimedia systems for decades. In 2005, the ACM SIGMM group met and outlined a set of future directions for multimedia research [8]. In this report they list 3 grand challenges in multimedia, the third of which is "... to make capturing, storing, finding, and using digital media an everyday occurrence in our computing environment". Our system contributes to this problem in that we are attempting to allow users to search across many videos with new interaction techniques. Other attempts at solving this problem include doing content analysis on chunks of video and providing summaries for a coarse search through video [9]. Our project differs in that it provides a way for users to interact directly with the subject that they wish to search for. There have been other direct manipulation techniques applied to searching for content in videos, as shown by Dragicevic et al[3]. This work allows users to find alternate locations for objects of interest by drawing a trajectory that the object follows in the video. SceneSkim[7] shows that structured query of video can be greatly enhanced if additional metadata is present, such as synchronized script text along with movies and closed captions. Video lens[6] shows that faceted search across a video is possible with linked numerical metadata. Our approach differs from both of these (but is not exclusive from) in that we associate videos with spatial data.

Sousveillance

Sousvellance is defined by Mann, Nolan and Wellman[5] "as both a conceptual framework for and as a performance of various techniques of self-empowerment in opposition to modern technologies of surveillance". Their work provides examples of five ways in which sousveillance is structured in modern society. Sousveillance devices such as the Microsoft SenseCam have been studied to determine what meaningful ways sousveillance can be used to improve daily lives. The Rashomon Project[1] is an interface for synchronizing multiple sousveillance streams for easier browser. Our project is similar in that we also allow users to navigate multiple synchronized streams, but we provide interaction techniques for users to be able to search for information in such an interface.

Interaction Techniques

We take inspiration for our interaction techniques from a number of previous projects. Distancecut[2] describes an algorithm that allows users to scribble roughly over particular objects in images, and will automatically segment those objects from the rest of the image. We implement a similar, though more coarse, type of scribble interaction. SpaceTop[4] joins 2D and 3D spatial interactions in a single system. We take inspiration from many of the types of interactions demonstrated in this work, although we differ in that our system has users interaction with a computer screen as opposed to a touch surface.

3 System Design

We have created AR-Video Query with the goal of enabling users to spatio-temporally explore multiple streams of video data captured within the same physical environment. In order to achieve this, we have designed a comprehensive system pipeline.

3.1 Input Capture

The pipeline begins with the collection of all of the necessary channels of input data for our system. We seek to gather data captured by the RGB camera, inertial measurement unit (IMU), and Light Detection and Ranging (LiDAR) scanner on a user's smartphone and feed these data channels into a Simultaneous Localization and Mapping (SLAM) algorithm. From this process, we can extract and compute 3 types of output data: recorded video in mp4 format, 3D mesh of the environment in obj format, and camera extrinsic and intrinsic parameters in JSON text format.

3.2 Query User Application

For the user-facing application, our interface enables a multitude of user queries, leveraging the data streams we have collected and synchronized. The query process can be broken down into a closed-loop system of 3 states: spatial, temporal, and results. Each of these states are simultaneously visualized, and they each offer various avenues for interacting with the other states.

3.2.1 Spatial

For the spatial interface, we first focus on empowering a user to freely view and navigate around the generated surrounding 3D environment. Since the 3D environment mesh itself is made up of detailed 3D shapes that lack any texture, we project the rendered view from the current video frames from each video camera on to the 3D mesh. This visualization design proves fruitful in reflecting real-time video frames during temporal query interactions, which will be described in further detail in the temporal state section.

We also design an interaction tool that allows a user to focus on points or sections of the 3D mesh that they are interested in querying. These query interactions can be conducted either within the 3D mesh space or on top of the 2D video visualizations, which then flow naturally into being reflected in the other two states (temporal and results).

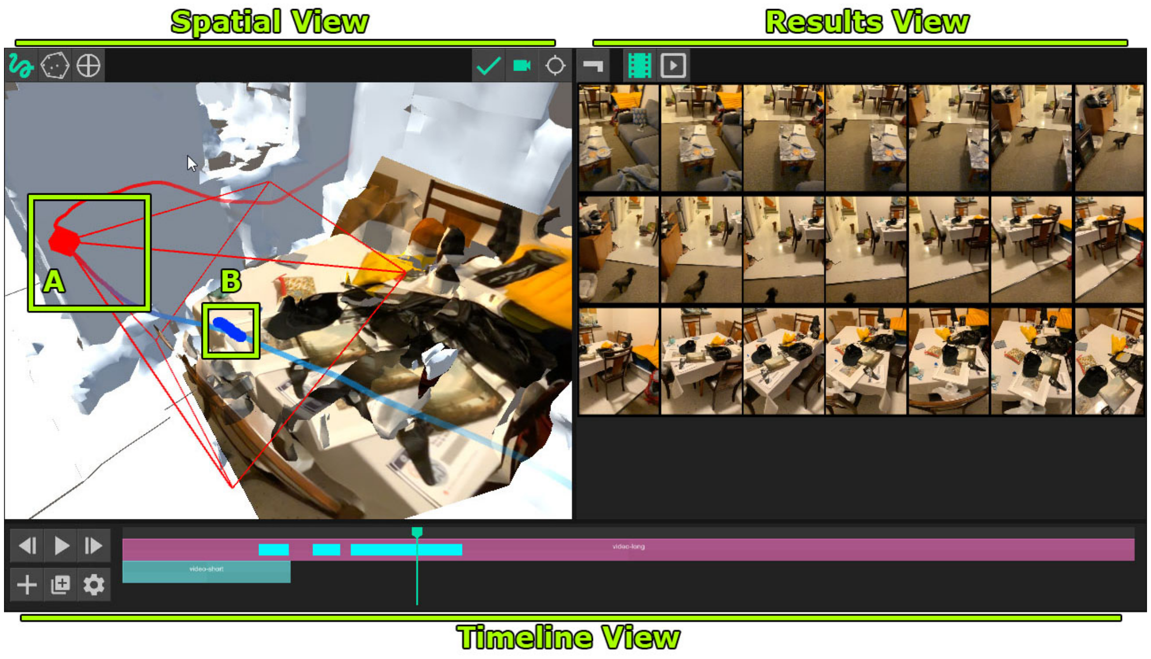

Fig. 2. AR-Video Query interface. Top left shows the spatial view, top right show the results view, and bottom shows the temporal timeline view. Callout A shows the camera position rendered in the spatial view. Callout B shows an example paint stroke query.

Additionally, by enabling a user to select a subset of such regions of interest, a user can define and capture spatial relationships between different parts of the environment such as objects, people, walls, etc.

3.2.2 *Temporal*

We aim to provide a user a clear understanding of where each frame of each video is situated in time relative to the totality of the synchronized videos. To that end, we create a temporal interface consisting of timelines for each video which are synchronized and then stacked on top of each other into a single coherent timeline.

When the spatial query interaction is invoked, the temporal interface seeks to represent the salient times and time intervals in which the queried parts of the 3D mesh are present in camera view.

Meanwhile, we also enable the standard user functions of playing, pausing, and scrubbing through different time frames in the overall timeline. These effectively serve as temporal queries that again flow into visualizations in the other two states (spatial and results). Part of our design of these UI components was inspired by Adobe Premiere Pro's interface.

3.2.3 *Results*

In the results section of our interface, we strive to give a user a 2D results panel for understanding the outputs of spatial and temporal queries. The panel will, in certain cases, fill up with some number of videos concurrently to reflect the query or current time. This visual format proves convenient for a user to keep track of the videos playing all at once. In other situations, the panel will reflect a mosaic of individual thumbnail images with certain time frames within

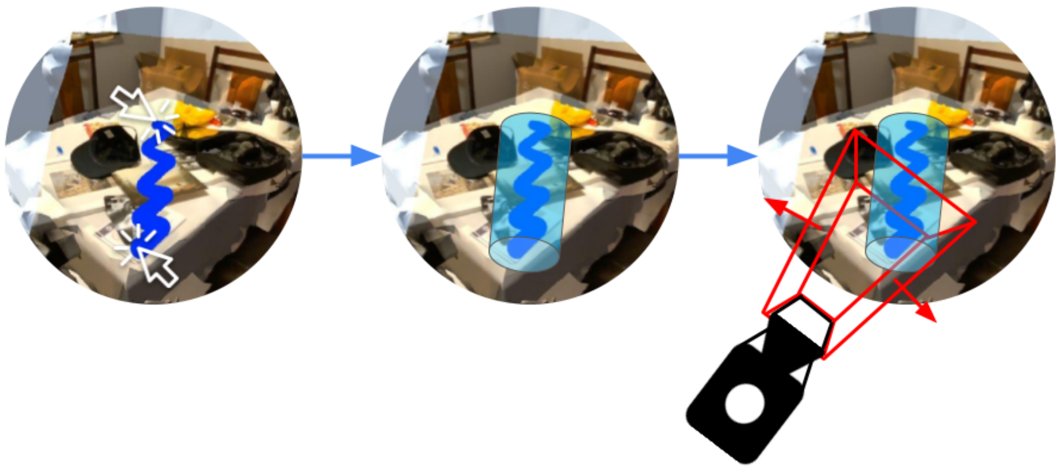

Fig. 3. Example interaction in the spatial interface. First the user scribbles a line over the object of interest. A bounding hull is computed for the paint stroke. Finally, the bounding hull is checked for visibility against all camera view frustums for all frames of video.

intervals that a user is focusing on. This alternate visualization design serves as a helpful visual preview tool for a user that summarizes what is occurring in certain spatial regions or time intervals.

4 System Implementation

4.1 Input Capture

The system pipeline implementation starts with a basic Xcode project written in Swift that interfaces with an iOS device. It then extracts and computes the recorded video (mp4), 3D mesh (obj), and camera parameters (JSON). Any iOS device can be used to capture the video data with its camera and the camera localization data with Apple's ARKit. Meanwhile, although older iOS device generations can still utilize ARKit's basic plane detection, only the iPhone 12 Pro, iPhone 12 Pro Max, and the iPad Pro have the new LiDAR (Light Detection and Ranging) sensor to make full use of ARKIt 4's depth API and SLAM (Simultaneous Localization and Mapping) for generating the full 3D mesh data.

4.2 Query User Application

Spatial

For user navigation, we make use of Unity's fly-through mode to lend a user maximum flexibility in flying around the 3D mesh environment with 6DOF (Degrees of Freedom). The specific keyboard-mouse instructions consist of: right click + drag for rotation, right click + WASD for forward/left/back/right, and right click + QZ for up/down.

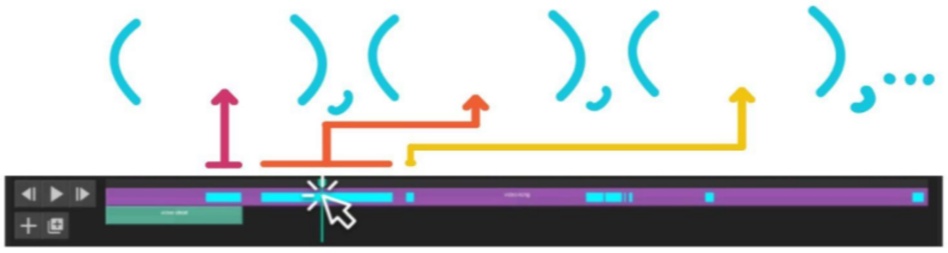

Fig. 4. Example showing how result intervals are displayed in the timeline view. Intervals represent the range of frames for which a query object is visible. These intervals are then colored on the video timelines to signify their location in the video.

Within this 3D mesh environment, we implement the rendering of real-time textures of the parts of the mesh in view of the video cameras via a projection shader. The shader renders a 3D frustum out from the video camera, checks when it clips any areas of the mesh, and changes the texture of the clipped areas to be that of the video frame output.

To enable the user spatial query interaction, we implemented a "painting" tool for the user to paint areas of interest in the environment with their mouse's left click + drag. For this paint interaction, we provide two modes - line rendering and convex hull. In line rendering mode, our paint tool renders small 3D spheres that trace the path of the user's mouse in 3D space. This is represented by interaction A in Figure 6, where a zigzag is painted on the table, which is then reflected in the timeline with all blue intervals containing video frames with that zigzag visible. In convex hull mode, our paint tool again renders the same collection of small 3D spheres, but then computes the 3D convex hull that surrounds those points and overlays this generated shape onto the scene. The resulting line or convex hull are referred to as a query object.

After painting a query object, we use it to perform a query on all video frames.

For every frame of video, we compute a world space view frustum for the space that the video can see. The individual points in the query object can then be checked against this frustum using a series of dot products. This reduced the query time to about 100ms for a 1.5 minute long video.

Results of a spatial query are in the form of a sequence of intervals for each video. These intervals represent the frames of video in which the query object was visible to the camera

Temporal

For the temporal interface, we created a timeline section of a fixed width that contains all of the video timelines that have

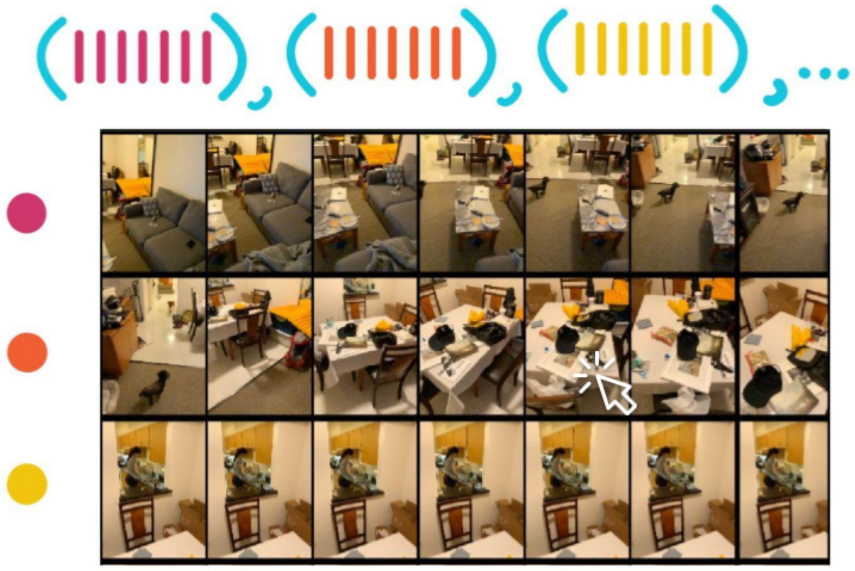

Fig. 5. Example of how the results are displayed in the Results View. Intervals are sampled uniformly, then images at those sample times are extract and placed together as a strip of thumbnails. One thumbnail strip per results interval.

been time-synchronized. These individual video timelines are implemented with color-coded strips of widths relative to their length. We decided on this implementation to give a user a strong visual sense of the videos' relationships to each other without the friction of having to scroll around an interface to uncover obscured sections.

To represent the results of queries from the spatial state, the temporal state portrays small icons and strips of a different color that are overlaid on top of the timeline strips at the result intervals.

We create icons and buttons to enable query interactions within this timeline section. The play, pause, and frame forward buttons can be left-clicked with a mouse to execute those functions. A user can also left-click and drag the timeline slider around to seek to a specific time that they are interested in. This is represented by interaction B in Figure 6, where clicking on part of the timeline translates to moving the camera position in the spatial view, which therein changes which part of the 3D mesh is textured.

Results

The results view allows us to render out the results of the query as a strip of thumbnail images from the result intervals. Thumbnails are sampled uniformly across each interval, and an image is taken from the video at each sample point to provide a visual indicator of all of the points in space and time in which the query object is visible. These sampling transitions are represented by transitions 1 and 2 in Figure 6, where the 2nd interval is sampled uniformly 7 times and the 5th thumbnail reflects the 5th sampled time. The thumbnails in these strips can be clicked to synchronize the camera to the corresponding point in space and time to facilitate easier searching and contextualization of results. This is represented by interaction C in Figure 6, where clicking on the 5th thumbnail in the 2nd row brings the user back to the correct point in space and time.

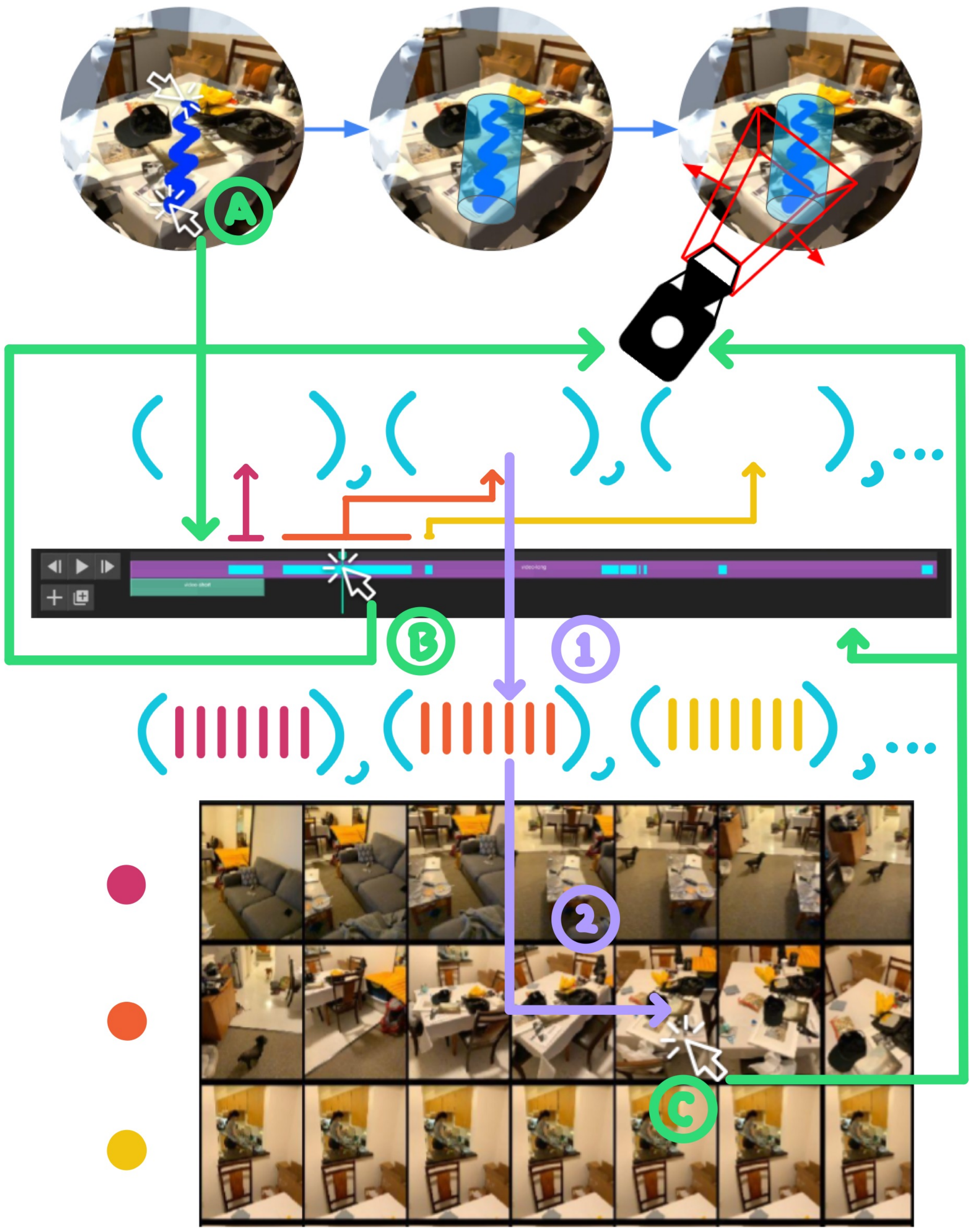

Fig. 6. AR-Video Query interaction flow diagram. Interactions in the different views affect other views to allow for interactive exploration of the results. Interaction A: Spatial paint query whose results are reflected by the light blue intervals of video timelines in the temporal state. Interaction B: Temporal timeline query whose results are reflected by the video camera position and texturing in the spatial state. Interaction C: Results thumbnail query whose results are reflected by the video camera's position in the spatial state. Transition 1: Demonstrates the transition from valid intervals to uniformly sampled times. Transition 2: Demonstrates the transition from uniformly sampled times to visual thumbnails of those video frames

5 Discussion

In implementing these interactions, we are able to pilot what the user's experience would feel like when trying to perform video queries. A number of improvements were identified.

For the spatial interface, it would be very useful to be able to specify logical operations on multiple paint strokes. Currently the system only handles the OR operation (any of the paint strokes can be in view), meaning that all paint strokes are included in the query object. We identified a few cases where it would be very useful to be able to AND (all paint strokes must be in view) or XOR (any single paint stroke is in view) to be able to generate a more complex set of outputs.

When query objects enter the camera's view, there is often a bouncing effect that happens due to the noise in the AR tracking information, as well as any shake due to the fact that the camera is hand held. This manifests in the temporal view as a sequence of short intervals where the object is computed as being in view and then out of view. We believe a simple debounce algorithm would suffice to solve this issue.

For the results view, while the thumbnail strips do provide a good representation of the results output, we found that it would be nice to also play the video represented by the result intervals. We imagine this working by allowing the user to play the timeline, but if a result interval is present, to only play the frames from those intervals. The videos will be rendered in the results area as a video player for each loaded video.

6 Evaluation Plan

When the work from the discussion section is complete, we believe the interface will be at a point that we can run a user study. This section describes our plan for such a study.

Study Groups

As we are comparing this new interface to existing solutions, we will test 3 groups in our study. The first group will attempt to complete each task by looking at the recorded videos directly. We could either provide this as a folder full of files, or in an interface that has the videos loaded so they all could be played at once. Our second group will use a purely time synchronized approach described in the Rashomon project. The third group will use the AR-Video Query software described in this paper.

Tasks

We need to consider what types of benefits this software provides when thinking about task design. The main benefit over something that is purely time synchronized is the fact that we also have spatial synchronization. Therefore, the types of tasks we will ask the user will be of the form "find when this object enters a given space" or "find all intervals of video in which this action occurs". Additionally, we hypothesize that with some combination of logical operations added to queries, we will also be able to generate diverse sets of training data for training neural networks. Another task to test this would be to ask the users to gather a specific set of data to train a convolutional neural network.

Resulting Data

Performance of each condition can be measured in a number of ways. We can record the amount of time it takes the user to complete their task, as well as record the number of actions the user has to take to perform the action. We would also ask users to complete a questionnaire aimed to determine which interfaces they found confusing, attempting to determine if spatio-temporal queries are something that users can understand easily or if it is more challenging than the other approaches.

7 Conclusion

Leveraging the historic emergence of video querying and mobile AR tracking capabilities, we created AR-Video Query to enable spatio-temporal queries on data captured by a user's smartphone. The software has strong potential applications in augmenting sousvellance tools and collecting computer vision network training data. We chose to design a system pipeline that would empower a user to capture, query, interact, and visualize multiple streams of data. We successfully implemented such interfaces with fluid interactions among the spatial, temporal, and results states. Lastly, we plan to conduct user studies to quantitatively and qualitatively demonstrate the benefits of our software system; this would involve reaching out to certain groups of people and asking them to take part as a volunteer in our carefully planned evaluation study.

References

[1] [n.d.]. http://rieff.ieor.berkeley.edu/rashomon/

[2] Xue Bai and Guillermo Sapiro. 2007. Distancecut: Interactive Segmentation and Matting of Images and Videos. In *2007 IEEE International Conference on Image Processing*, Vol. 2. II – 249–II – 252. https://doi.org/10.1109/ICIP.2007.4379139

[3] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. 2008. Video Browsing by Direct Manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 237–246. https://doi.org/10.1145/1357054.1357096

[4] Jinha Lee, Alex Olwal, Hiroshi Ishii, and Cati Boulanger. 2013. SpaceTop: Integrating 2D and Spatial 3D Interactions in a See-through Desktop Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 189–192. https://doi.org/10.1145/2470654.2470680

[5] Steve Mann, Jason Nolan, and Barry Wellman. 2003. Sousveillance: Inventing and Using Wearable Computing Devices for Data Collection in Surveillance Environments. *Surveillance & Society* 1 (01 2003). https://doi.org/10.24908/ss.v1i3.3344

[6] 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*. 541–550.

[7] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 181–190.

[8] Lawrence A. Rowe and Ramesh Jain. 2005. ACM SIGMM Retreat Report on Future Directions in Multimedia Research. *ACM Trans. Multimedia Comput. Commun. Appl.* 1, 1 (Feb. 2005), 3–13. https://doi.org/10.1145/1047936.1047938

[9] Klaus Schoeffmann, Mario Taschwer, and Laszlo Boeszoermenyi. 2010. The Video Explorer: A Tool for Navigation and Searching within a Single Video Based on Fast Content Analysis. In *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems* (Phoenix, Arizona, USA) *(MMSys '10)*. Association for Computing Machinery, New York, NY, USA, 247–258. https://doi.org/10.1145/1730836.1730867

[10] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. 2014. Global Localization from Monocular SLAM on a Mobile Phone. *IEEE Transactions on Visualization and Computer Graphics* 20, 4 (2014), 531–539. https://doi.org/10.1109/TVCG.2014.27

[11] Juyong Zhang, Yuxin Yao, and Bailin Deng. 2021. Fast and Robust Iterative Closest Point. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1. https://doi.org/10.1109/tpami.2021.3054619