A Pursuit Method for Video Annotation

by

Zoltan Foley-Fisher

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

November 2012

Abstract

Video annotation is a process of describing or elaborating on objects or events represented in video. Part of this process involves time consuming manual interactions to define spatio-temporal entities - such as a region of interest within the video.

This dissertation proposes a pursuit method for video annotation to quickly define a particular type of spatio-temporal entity known as a point-based path. A pursuit method is particularly suited to annotation contexts when a precise bounding region is not needed, such as when annotators draw attention to objects in consumer video.

We demonstrate the validity of the pursuit method with measurements of both accuracy and annotation time when annotators create point-based paths. Annotator tool designers can now chose a pursuit method for suitable annotation contexts.

Preface

A pursuit method for video annotation has been a collaborative research effort as part of the MyView project at the Human Communication Technologies Laboratory by Professor Sidney Fels, Dr. Gregor Miller, Visiting Professor Daesik Jang, Ph.D. student Abir Al-Hajri, Master students Michael Ilich and Matt Fong, visiting students Nicolas Pajnic, Loïc Duron, Manual Fernandez, and me.

Some of the work reported in this thesis resulted in the following demonstration, for which I designed and developed the cross-platform annotation software framework reported in Appendix G:

- Gregor Miller, Sidney Fels, Abir Al Hajri, Michael Ilich, Zoltan Foley-Fisher, Manuel Fernandez and Daesik Jang. 2011. MediaDiver: Viewing and Annotating Multi-View Video. In Proceedings of ACM CHI Conference on Human Factors in Computing Systems Extended Abstracts. ACM, New York, NY, USA.

Professor Fels and Dr. Miller actively supervised the research project and contributed ideas on the development and evaluation of the annotation framework. All content reported in this thesis was part of my research during my participation in the project, including the implementation of the annotation software framework and prototyping of the interaction techniques, the design and execution of all pilot studies and formal user studies, and the analysis of all data that resulted from the experiments.

Permission to reproduce some images in this thesis was kindly granted by YouTube users iPix966, chandlerwantsahorse and SoccerXAcademy.

The MyView research project was funded by the Natural Sciences and Engineering Research Council of Canada. All experiments reported in this thesis have been approved by the UBC Behavioural Research Ethics Board (Certificate Number H08-03006 and H09-00066).

Table of Contents

List of Tables

List of Figures

Acknowledgements

My supervisors Prof. Sidney Fels and Dr. Gregor Miller challenged me to contribute to the field of Human-Computer Interaction and provided valuable insights, practical advice, proofreading and encouragement.

Thank you to Dr. Chris Oram, Abir Al Hajri and the HCT lab, Rock Leung, Russ Mackenzie and the HCI Grad Research Forum for invaluable feedback. Many thanks to Prof. Rodger Lea, Prof. Karon MacLean, and Prof. Joanna McGrenere for important feedback and support. Thanks also to Kenji Okuma, Wei-Lwun Lu and the LCI lab; Idin Karuei, Vincent Levesque, Louise Oram and the MUX lab; and my friends Vincent Tsao, Johnty Wang and the MAGIC lab. Thank you Victoria Jeffries for encouraging me to pursue graduate studies.

This research would not have been possible without the support of my family, particularly my parents Nigel and Barbara, and my wife Françoise - you give me confidence, help me to understand, and reassure me that life continues merrily.

Chapter 1

Introduction

Each month on YouTube, people watch over three billion hours of streaming video [4]. Every minute, mobile users upload three hours of video to the service from handheld devices.

Finding people and events in this torrent of video is difficult. Photo sharing services, like Facebook albums, provide quick techniques to identify people and so organise media collections [86]. Yet simple equivalent techniques to organise video media are not yet commonplace.

Adding information to objects in video - such as tagging people as in Figure 1.1 - is known as annotation. Annotation enables a range of exciting opportunities such as finding interesting videos or sharing thoughts about video objects.

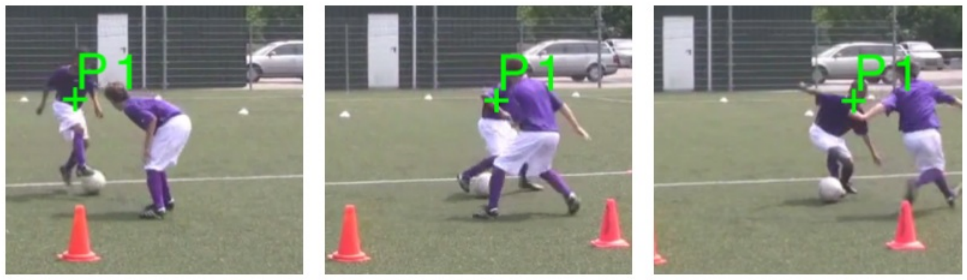


Figure 1.1: A point-based annotation to draw attention to a video object.

Contemporary annotation of consumer video is typically a slow process. Some automated annotation methods require time-consuming video preprocessing [38]. Other methods hamper annotators by interrupting video playback during annotation [8].

However, annotation presents a speed and accuracy trade-off in time and space. The more dense an annotation - spatially and temporally - the more time needed to create the annotation. At one extreme, tracking a single point of interest reduces the time needed to accurately specify an annotation

- although at the cost of accurately specifying a region of interest.

Furthermore, for many annotation applications, a point of interest is sufficient. For instance, annotators can draw attention to screen-based objects with a crosshair or a fixed-sized shape centered at a point [50]. Video effects or interactive elements can be centered on point-based paths [38]. Annotators can also position points to identify objects, in a similar fashion to fixed-sized Facebook photo tags. Additionally, a point of interest may also be combined with computer supported region growing to provide fast, accurate region of interest video annotation.

This thesis provides evidence that, with the appropriate interface, tracking a point of interest can be performed sufficiently accurately in realtime, and - in some cases - faster than realtime. Thus we demonstrate that a pursuit approach for annotating single points of interest is a viable approach for fast video annotation.

First, we present a predictive model of the accuracy of annotation paths created with a pursuit annotation method. We derive the model for a hard case of synthetic video object movements - when objects reverse direction. We also assumed a pursuit with a mouse pointer, when objects change direction near the limit of human reaction times.

Our model indicates that the speed and size of video objects explain a substantial portion of the variability in the peak error during a pursuit. Additionally, the model indicates a linear relationship between speed, size and peak error.

Secondly, we present measurements of pursuit accuracy of more realistic video object annotations. We compare the performance of the pursuit method with an alternative video annotation technique - known as Hold-Pause - that interrupts video playback during annotation.

Measurements of annotation path accuracy in the second study exhibit the same linear relationship between speed, size and peak error. We show that when video objects move at 1mm/s, pursuit interactions produce annotations as accurate as the alternative technique (for video objects 1.25 mm in diameter). But pursuit interactions produce the annotations faster, requiring just 1.05 seconds of annotator time for each second of video - a 12% reduction in annotation time (with a standard deviation of 5% reduction to annotation time).

Moreover, we show how our model can be incorporated into a annotation system so that by slowing down the video, an annotator can achieve an arbitrary accuracy with a pursuit method. Alternatively, if objects are moving slowly, the annotator can speed up video playback and the method will produce predictably accurate paths - and faster than realtime.

At first glance, video annotation designers may be sceptical that tracking a video object would be a viable annotation technique. We present quantitative evidence of the validity of a pursuit method for video annotation, including measures of video annotator satisfaction. System designers can now choose a pursuit method as a viable annotation method - and help people to stay afloat in the rising tide of online video.

1.1 Contributions

The contributions of this work are

- A new manual method for video annotation. The method does not require video to be paused during annotation. We provide evidence that a pursuit method is suitable for tracking a point of interest on a video object. For consumer video, the methods produce satisfactory point-based paths with an accuracy that compares favourably with an alternative method.
- A model of how pursuit accuracy is related to video object speed and size, based on the results of a controlled experiment. With our the model, we report how to both adjust video playback rate and time-shift the paths annotators create so interaction designers can best apply a pursuit method for video annotation.

Chapter 2

Related Work

Digital annotations were first conjectured in 1945 when Bush described a futuristic table known as the Memex [18]. He imagined that the device would assist a researcher to add marginal notes and comments to a trail of documents. At the Stanford Research Institute in 1968, Engelbart demonstrated a working annotation system for knowledge workers - the oNLine System (NLS) [33]. He joined a group of remote programmers to write software and annotated blocks of code.

More recently, Harrison describes video annotation as simply "note taking" [42]. But Goldman elaborates that "video annotation is the task of associating graphical objects with moving objects on the screen" [38]. In this thesis, we adopt a definition of video annotation similar to Goldman: a type of digital annotation to associate spatio-temporal entities with events and objects in video.

A pursuit method sits within a broad field of video annotation research. To contextualise the method, we first introduce video annotation representations and detail a particular type of annotation - the point-based path. We then illustrate the space of methods to create and correct annotations, providing a vocabulary for discussing annotation methods and identifying some of the shortcomings of existing methods.

2.1 Video Annotation Representations

Four key elements comprise a video annotation: (1) video media, (2) semantic entities, (3) spatio-temporal entities and (4) metadata. In this chapter, we introduce video media, semantic entities and spatio-temporal entities. Annotation metadata is discussed in Appendix A. The elements are based on video annotation representations described in two taxonomies: MPEG-7 and SMIL.

The Moving Picture Experts Group (MPEG) developed the highly-detailed taxonomy of video annotation representations known as MPEG-7 [62]. Simpler taxonomies of graphical content, such as the Library of Congress Thesaurus for Graphic Materials, only represented a limited range of video annotations [69]. MPEG-7 represents a broader range of annotations at different "levels" of annotation with a Description Definition Language (DDL). The levels bridge a "semantic gap" - low-level features, such as the edges of objects, are encoded with low-level Descriptors, while high-level features such as events and objects are encoded with other high-level Descriptors.

The Synchronized Multimedia Integration Language (SMIL) is a simpler alternative taxonomy to MPEG-7 [16, 46]. Developed as a format for interactive multimedia presentations, the language groups video annotation representations into modules of functionality. The Media Object module details the playback and positioning characteristics of images and video. The behaviour of "area" elements are described in the Linking module. The Metadata module details the structure of information attached to multimedia elements. The simpler SMIL format is used for mobile applications such as multimedia text messaging (MMS).

2.1.1 Image and Video Media

A video annotation is always relative to video media. Media can be (1) copied, like a digital music file, (2) stored, like an x-ray or (3) transmitted, like a newspaper story. Video media are the digital substrates that store or communicate events and objects recorded in video.

In this thesis, we assume that an image is a two dimensional grid of spatial elements, known as pixels. Each pixel has a colour that is uniformly rendered by a display across the pixel area.

We further assume that video is a two dimensional grid of pixels - like an image - but that the colour of each pixel is time dependent. A frame is an image of pixel colour values for a particular time period. Frame pixel colours are defined for discrete periods - determined by the video frame rate. During video playback, a display renders pixel colour uniformly during the frame period.

2.1.2 Semantic Entities

Semantic entities should not be confused with video media - when someone annotates a soccer player, the object they are interested in is usually the person, rather than the pixels portraying the player. MPEG-7 semantic entities are the significant events or objects in video.

- Video Objects: Any entities that occupy space, such as a soccer ball or a pool of milk. Objects usually change over time - a tree grows slowly and sways in the wind. An object may be composed of smaller parts, like a swarm of bees. Video objects with distinct boundaries, such as people or vehicles, are particularly suited to annotation with a pursuit method.

- Video Events: Abstract entities, such as a volcano eruption, a Christmas sale or a car theft. The start and end of an event may not be distinct - such as when a life starts and ends - but events can be ordered and measured relative to each other. Here, we assume that an event has both a start and an end, relative to other events.

Some annotation tools specialise in annotation of particular objects, such as the TrackMarks tool that only annotates the locations and identities of people [27]. Other annotation tools, such as the Family Video Archive [5], EVA [60], VANNA [42], Marquee [93] or CEVA [24] only support event annotations.

Sometimes the semantic entities are ambiguous: when a newspaper editor underlines text, the semantic entity may be either details of a story or a grammatical error. Given a compound video object, such as a person wearing clothes, the semantic entity may be the person or their clothes, so an annotation viewer plays a role in disambiguating an annotation.

2.1.3 Spatio-Temporal Entities

MPEG-7 region descriptions are spatio-temporal entities that correspond to semantic entities. By spatio-temporal, we mean that the entites are defined in units of space and time, relative to the spatial and temporal dimensions of video. Spatio-temporal entities can be imagined as sausage-like extrusions over time (Figure 2.1).

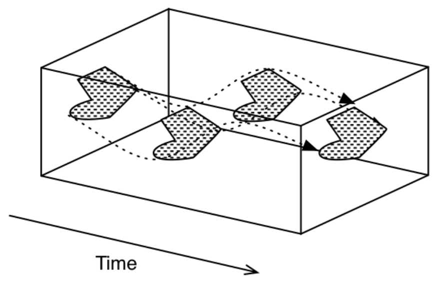


Figure 2.1: A spatio-temporal entity that can be imagined as a sausage-like extrusion of a changing shape through time.

Spatio-temporal entities have a *correspondence* to semantic entities, and a spatio-temporal entity should correspond to exactly one semantic entity. The definition of the spatio-temporal entity should be accurate enough so that the correspondence is clear. We define an annotation error as an unsatisfactory correspondence between a spatio-temporal entity and a semantic entity. However, spatio-temporal entities can inaccurate - such as when a newspaper editor draws a line under a sentence - and human perception determines both the extent of the spatio-temporal entity and the corresponding semantic entity.

Measures of spatio-temporal entity accuracy are often based on the degree of overlap between a *ground truth* - known good entities - and annotator created entities. The CLEAR MOT measure is an example of an overlap accuracy measure [14]. However, in the case of point-based paths, annotation accuracy can be measured with frame-wise geometric error between ground truth paths and annotator paths.

Spatio-temporal entities have a visualisation independent of their definition. The visualisation determines how well people perceive the correspondence between a spatio-temporal entity and a semantic entity. Sometimes, a visualisation will "enclose" a semantic entity to indicate a correspondence. For example, Facebook visualisations are located to identify faces in photos, but rarely fit faces well [1].

There are a number of types of spatio-temporal entity, with different *densities*, relative to the quantity of media data annotated. Dense spatio-temporal entities have detailed definitions. In comparison, sparse spatio-temporal entities have more rough definitions and are quicker to create and transmit, and require less information storage space. Figure 2.2 arranges three different spatio-temporal entities in terms of their temporal and spatial density.

Purely Temporal Periods

Spatio-temporal entities can be purely temporal when corresponding to events, such as to correspond to a song played during a concert. MPEG-7 represents purely temporal entities with Video Segment Descriptors [62]. Purely temporal entities are found in video annotation tools for ethnographers, such as EVA [60] or VANNA [42] (see also Appendix C) and have two different representations:

- Instantaneous. Those temporal entities that occur simultaneously with a corresponding event, but without a duration, so relying on human

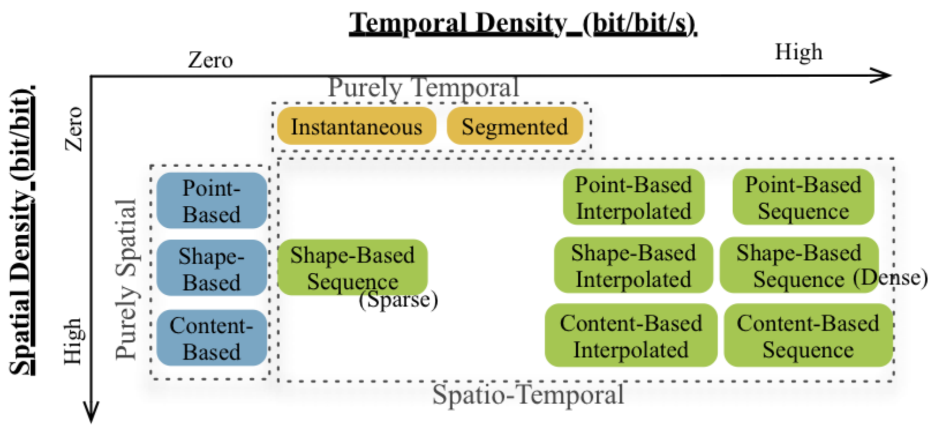


Figure 2.2: Comparison of three spatio-temporal entities - Purely Temporal, Purely Spatial and Spatio-Temporal - in terms of the information density of the entities. Information density is relative the quantity of media data annotated. A pursuit method typically produces sequence or interpolated point-based paths.

perception to infer the duration of the event. For example, a match time can refer to a soccer goal even though the event involves game play before and after the ball crosses the goal line.

- Segmented. Those temporal entities that enclose events, although perhaps with rough boundary start and end times. In comparison with instantaneous entities, they define an event duration, so are more dense.

Purely Spatial Areas

Similarly, spatio-temporal entities can be purely spatial when corresponding to objects in images. Spatial entities have three different representations, with increasing information density:

- Point-based. These entities only specify positions, such as (1) the location of fixed shape visualisations like the Facebook face area [1], or (2) simple point visualisations like a laser pointer dot or a pin location on a map.

- Shape-based. Entities formed from parameterised shapes that regularise areas. The shape is typically simpler than the corresponding video object shape, so is a "lossy" representation of complex video object shapes such as tree canopies. However, shape-based entities correspond to objects better than point-based entities, and can be rotated and scaled. Different shapes have a range of information densities:

  - Simple forms. Ellipsoid or rectangular entities, with variable parameters such as radius or side length. SMIL represents rectangular shape-based entities with an area element [16].
  - Vertex-based. Vertices may be joined with simple straight lines or with more complex paths, such as as Bezier paths [12]. MPEG-7 represents vertex-based entities with Polygon Region Locator Descriptors [62].

- Content-based. The most dense entities, typically formed from groups of pixels. Alpha "mattes" are images that define groups of pixels - sometimes created by chroma-keying - and can have "soft boundaries" [22]. MPEG-7 represents content-based entities with Region-Based Shape Descriptors [62].

Spatio-Temporal Volumes

A combination of spatial and temporal entities is typically required to correspond unambiguously to semantic entities moving in video. Visualisations of spatio-temporal entities can be both (1) spatial for particular times, and (2) temporal for particular spaces (the basis of time-bars in video timelines). Spatial and temporal entities are combined in two ways:

- Sequences. The most dense representations of spatio-temporal entities are sequences of spatial entities explicitly defined for specific video times. MPEG-7 represents these entities with groups of spatial entities. Sequences of content-based mattes form the densest spatio-temporal entities [22].

  However, some sequences are sparse, so that entities are defined and visualised for only a few frames of a video clip. Sparse sequences of shape-based entities are used to index video, such as with the Semantic Video Annotation Tool (SVAT) [75] and VideoAnnEx [56].

- Interpolated. Representations that are less dense than sequences, and that rely on rules for entity definition between "keyframes". The technique is similar to animation keyframing, where junior artists define shapes for frames between keyframes drawn by senior keyframe artists [17]. Interpolation improves the correspondence of a spatio-temporal entity with a video object, with little increase of entity density other than parameters to define the interpolation.

Point-based spatial entities can be interpolated into spatio-temporal paths. MPEG-7 represents spatio-temporal paths with Spatio Temporal Locator Descriptors [62]. Content-based spatial entities can be positioned on spatio-temporal paths - similar to MPEG-4 motion vectors [22].

Shape-based spatial entities can be interpolated by making shape parameters time dependent. MPEG-7 represents such parameter interpolations with the TemporalInterpolation Descriptors [62]. By using different interpolations for shape-based entity parameters, entities can change shape as well as appear to move with time. Some content-based spatial entities can be interpolated with skeleton techniques [17].

Different interpolation schemes result in different entities. Interpolation with simple linear functions introduces discontinuities in the apparent shape change and movement of entities. More complex interpolation with parametric curves results in smoother spatio-temporal entity change with time, but at the cost of increased entity density.

Keyframes and interpolation are found in most video annotation tools such as GALATEA [71], MediaDiver [63] and YouTube Annotations [3], to name a only a few.

2.2 User Interfaces To Create Point-Based Paths

A pursuit method creates a point-based path - a specific type of spatio-temporal entity for video annotation. We arranged spatio-temporal entities by spatial and temporal density and noted that point-based paths have a reduced spatial density.

Creating and correcting spatio-temporal entities is a particularly time consuming task. However, methods to produce spatio-temporal entities with a reduced spatial density should also have reductions to annotation time, and so produce point-based paths quickly.

In this section, we compare a variety of methods to create point-based paths. Even with a reduced spatial density, point-based paths are useful for a number of reasons:

- Semantic Entity Identification: semantic entities are "tagged" for identification in subsequent viewings, such as with the Agora tool [89]. Point-based paths must only correspond unambiguously to video objects, and do not match the objects shape exactly.

- Attention Direction: annotation to draw attention to video objects, such as in classrooms [11]. Point-based paths to draw attention are typically of low spatial density and direct attention to one object in a collection of objects.

- Video Navigation: annotations for subsequent navigation with video, such as hyperlink anchoring [63]. Navigation point-based paths are typically of low spatial density, since the paths are the basis for simple interaction elements.

- Semantic Entity Analysis: annotation so video object behaviour can be analysed, such as for player movement analysis [83]. Some analyses can be made with point-based paths, such as horse racing performance.

- Video Effects: Point-based paths for can be the basis of highlights or animations that follow video objects [38].

However, the MPEG-7 and SMIL taxonomies only address representations of video annotations, and don't consider the process of creating or correcting annotations. To arrange methods to create point-based paths, we draw on two taxonomies of methods to create spatio-temporal entities. We extend these taxonomies to describe user interfaces to create and correct point-based paths for a wide variety of video annotations.

2.2.1 Smith's Taxonomy

Smith developed a taxonomy of methods to create video hyperlinks [82]. Video hyperlinks can be considered as video annotations to support navigation among multimedia resources via "linkage areas" or "hotspots" of video. Smith arranged methods to create hyperlinks along three axes: (1) manual vs. automatic generation, (2) server vs. client-side generation and (3) realtime vs. offline video. We consider methods to create spatio-temporal point-based paths along the two most relevant axes - degree of automation and creation time.

Table 2.1: Interfaces to create point-based path annotations arranged by degree of automation - either manual or automated.

	Manual	Automated
Example	Hold-Pause [47]	TurfTrax [83]
Method	An annotator stops video playback by holding a mouse button down. When the mouse button is released, a point is added to an interpolated path and video playback resumes.	Before annotation, technicians attach positioning devices to objects such as race-horses. The annotation system then aligns object paths detected by the positioning devices with video records of the objects.
Strength	Handles consumer video that is difficult for automation	Reduces number of frames an annotator manually edits
Weakness	Very slow relative to automatic methods	Often customised to particular video context

Degree of Automation

Automated object tracking systems often create point-based paths. The TurfTrax system creates annotations of racehorses in broadcast video with radio positioning devices aligned with video recording equipment [83]. Each horse caries a transmitting tag to send a radio-frequency "chirp" to receivers around a race course track. The receivers relay details of the chirps to a central server that produces position measurements of the horses for video statistics and visualisations. While the system estimates horse position and speed with similar accuracy to Differential Global Positioning System (DGPS) devices, aligning tracking devices with video is time consuming and expensive, and different tracking technology is needed for different sports contexts.

Some automated systems create point-based paths from video data alone. Amisco is a contemporary video system to generate coaching statistics of the movements of soccer players with overhead camera video [80]. The Noldus Ethovision tool measures animal movements with video recorded during experiments in well-lit cages [84].

Automated systems to create point-based paths based on video analysis alone often assume that objects move against an unchanging background in recordings from stationary cameras. In complex video, such as consumer video, automated methods introduce a variety of errors when creating point-based paths due to challenging lighting conditions or unpredictable content.

In contrast, completely manual techniques can cope with more challenging video conditions, but with longer annotation times. Hold-Pause is a completely manual technique to create point-based paths, based on an interaction technique to improve moving target selection [47]. Using the technique, the annotator navigates to a video time with a playhead and timeline. The video starts playing automatically once the annotator has specified the start location for a path with a mouse. Any subsequent mouse button press pauses the video so that the annotator can accurately acquire a moving target and define additional points on the path - interpolation between these annotator defined points forms the path. The path ends when the annotator manipulates video playback controls such as the video pause button or the video playhead.

Creation Time

Realtime annotations are created in time or very soon after video capture. Alternatively, offline annotations, such as hyperlinks generated for police surveillance video, are created sometime after the video is captured.

Annotators use some ethnographic tools to mark events in-time with video capture, such as EVA [60], VANNA [42], Marquee [93] or CEVA [24]. The temporal annotations are useful to support navigation within video, and can be refined into more accurate annotations. Pursuit methods extend these temporal only methods to introduce a spatial dimension and create point-based paths in-time with video capture.

A pursuit method produces temporally dense point-based paths in-time with video playback. An annotator tracks the movement of video object in time with playback. The method is described in detail in Chapter 3. The method differs from a typical spatial brush method since a brush tracing interaction is self-paced. With a pursuit method the annotator is under a constraint to trace the path within a limited time. A pursuit method is fast to create point-based path without stopping video playback, perhaps suitable for "first-passes" for correction later.

Offline annotation - while video playback is paused - allows for a variety of techniques to create spatial entities for annotations, including methods to specify points for point-based paths (see Appendix B). Spatial entities can be merged into a single spatio-temporal entity that corresponds to a video

Table 2.2: Interfaces to create point-based path annotations arranged by creation time - either realtime or offline.

	Realtime	Offline
Example	A pursuit method 3	Particle Tracking [38]
Method	Track video object with pointing device in time with video playback to produce temporally dense point-based paths.	Video is pre-processed offline to determine groups of particles. An annotator selects a group of particles to grow a spatio-temporal path so that the path is on the center of the group.
Strength	Faster to produce paths by avoiding interrupting video playback	Allows time to modify existing paths while the video is paused
Weakness	Some video could be annotated faster than realtime	Annotators under less time-pressure to complete annotations. Needs additional time to manipulate video playback controls.

object more accurately. However, annotators typically spend more time to produce more accurate spatio-temporal entities.

Some tools cut annotation time by reducing the number of video frames available for annotation, so annotators only create a spatial entity for a single frame for a thirty second video clip (VideoAnnEx [56, 82], SVAT [75]). These tools could potentially create sparse point-based paths in realtime, even though video playback is interrupted - but at the cost of annotation temporal density.

2.2.2 Wills' Taxonomy

Wills developed a general taxonomy of 2D selection methods [94]. His taxonomy includes both "data-dependent" or "data-independent" and "memory" or "memory-less" methods. We extend the taxonomy to consider methods to create and correct spatio-temporal point-based paths, and illustrate the taxonomy in Figure 2.3.

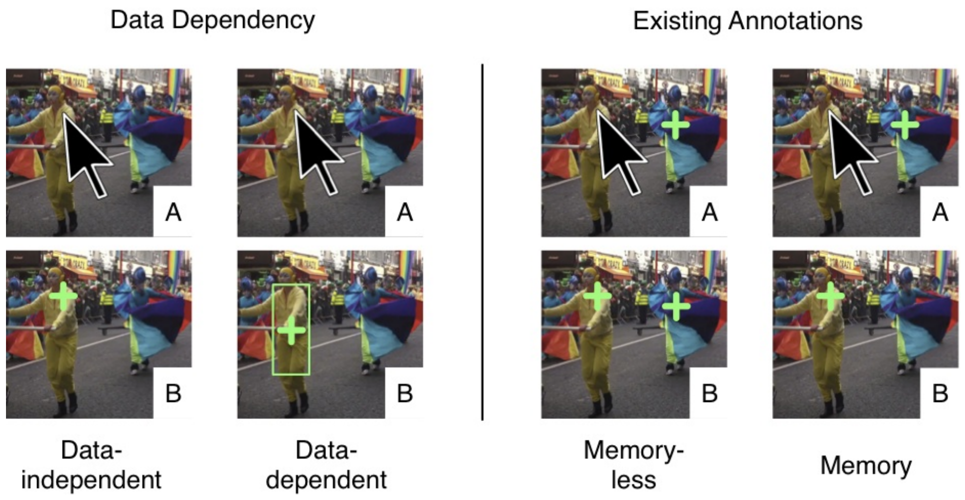


Figure 2.3: Comparison of four methods to create a point in a single frame for a point-based path. In A, the annotator positions the mouse pointer and pushes the mouse button down. When the mouse button is released in B, four different points result. A data-independent method simply places the point at the mouse pointer position. A data-dependent method - in this case based on pixel colour - places a point at the center of an area expanded around similar pixels. A memory-less method also places a point at the mouse pointer, but has no effect on existing points. A memory method updates an existing point to lie at the mouse pointer position.

Data Dependency

Data dependent methods define point-based paths based on properties of the underlying media data. Text selection is a well-known data dependent method for defining areas - by dragging a pointer, areas are constrained to conform to the positions and sizes of characters in the text.

Table 2.3: Interfaces to create point-based path annotations arranged by data dependency - either dependent or independent.

	Data-Independent	Data-Dependent
Example	Mark Corrections [12]	Magic Wand [8]
Method	Over a number of frames, animators update an existing path by recreating a new path. The update is based on parameterised curvatures of the paths.	Points for the path are based on the centers of a sequence of regions of interest selected by an annotator. The regions of interest are based on pixel colours detected in video frames.
Strength	Rough accuracy paths may be good enough for some annotation contexts	Reduces the number of frames an annotator manually edits
Weakness	Doesn't save time when paths could be detected from video	Annotator must review annotations to detect errors.

Magic Wand is a data-dependent method to expand a point into a content-based region [8]. To grow an area, an annotator selects a point on a video object image and an algorithm expands a content-based area around the point. The algorithm can grow areas based on (1) the presence of edges detected on the video object, (2) the presence of similar pixels to the selection pixel (in colour or intensity), or (3) a hybrid approach that fills "pools" within the image. Points of a path can be snapped to the grown area - such as the center of gravity of the area.

In contrast, Khan et al. explored the affordances and design characteristics of a data-independent "digital spotlight" to direct attention to objects in realtime [50]. Without any dependency on the underlying media, the spotlight was automatically stabilised on a large screen display - without the "wobble" of a laser pointer. On wall-sized screens, viewers found objects illuminated by the spotlight up to 3.4 times faster than when attention was directed to the object with a mouse cursor alone. While their evaluation only considered illumination of stationary objects, pursuit methods could be considered as an extension of their technique to follow moving objects.

Existing Annotations

Memory-less methods define point-based paths independently of previously created paths. In contrast, memory methods perform operations such as path intersection, addition or replacement.

Table 2.4: Interfaces to create point-based path annotations arranged by existing paths - either memory-less or memory.

	Memory-less	Memory
Example	Digital Spotlight [50]	Constrained Optimisation [9]
Method	Conference participants position a spotlight visualisation with pointing movements on a large screen display. As objects on the screen move, the spotlight is repositioned. Each repositioning is independent of previous repositioning of the spotlight.	An animator adjusts the interpolation between keyframe points by inserting points into the path between keyframe points. The path between the points is interpolated based on how well the interpolation follows edges in the intervening frames.
Strength	Rough accuracy paths may be good enough for some annotation contexts	Corrections to errors on a path can be quicker than recreating the path
Weakness	Even if path contains a small error, must recreate path on mistake	No time saving when there are no existing paths

Baudel describes a technique to modifying existing paths with successive paths [12]. Although developed for spatial paths, his technique could be applied to spatio-temporal paths. His technique is similar to adjusting a sketch by repeating pen strokes - the technique transforms Bezier curves according to successive curves drawn on an original. The advantage of the technique is natural, fast and direct interaction with paths, instead of manipulating “low-level” control points of a path.

Points of a path can be defined based the center of a group of particles - image features that correlate with video object movements. Goldman developed a data-dependent method to select spatio-temporal volumes detected by a particle tracking algorithm [38]. The research assumed that video is pre-processed offline to determine groups of particles. To create a path, the annotator selects a particle group and the center of the group defines the path in each video frame. His work mentions situations when the particle tracking "breaks down" - when the particle group no longer corresponds well to video objects - but doesn't investigate methods to address the situation.

Data-dependent also includes automated memory methods that are constrained by existing annotation paths. Agarwala et al [9] reframed rotoscoping - an animation technique to trace the outline of objects represented in video - as a semi-automated process similar to video annotation. With the technique, an animator specifies waypoints in two keyframes, and an algorithm creates paths by both interpolation between the waypoints and optimising measures of how well the interpolation follows edges in the video.

2.3 Discussion

2.3.1 Annotation Method Comparisons

We have arranged methods to produce point-based paths and noted the trade-offs between the methods.

Although fully automatic annotation systems reduce the number of frames an annotator manually edits, they are often customised to particular video contexts. Manual systems can handle consumer video that is difficult for automation, although the methods are slow relative to automatic methods.

Realtime annotation systems avoid interrupting video playback. However, offline systems have the advantage of allowing annotators time to modify existing paths.

Data-independent methods can be useful in situations where rough accuracy paths are good enough. In contrast, data-dependent methods often reduce the number of frames an annotator manually edits, although they may still require some supervision by annotators.

Similarly, memory-less methods can produce paths with sufficient accuracy for some applications. But if more accurate paths are needed, memory methods have the advantage of correcting errors on a path quickly rather than recreate the path.

In this thesis, we chose to focus on completely manual methods. However, manual methods can be combined with automated techniques. For instance, point-based paths created by manual methods can constrain automated methods or select the results of automated methods. But as an initial step to validate pursuit methods, we consider a completely manual method, leaving work to combine pursuit methods with automation as interesting research for the future.

2.3.2 Input Device Considerations

Although some automated systems assume specialist tracking hardware, most methods to create and correct point-based paths typically rely on pointing or tracking interactions. However, we note that our taxonomy does not consider the range of input devices for methods to create point-based paths in detail.

Some devices will significantly affect video annotations interactions. For instance, touch screen devices typically do not combine interaction states - such as defined with multiple buttons - with position measurements. In contrast, a computer mouse has buttons to control the properties of annotation methods. An annotation system with a mouse based interface could add a new path while the left mouse button is depressed, or modify an existing path while the right mouse button is depressed.

Additionally, while is often performed in a quiet environment with a desktop computer, annotation on mobile devices is become increasingly common. In mobile situations, some device specific issues impact video annotation accuracy and the time to produce annotations - such as finger-touch occlusions, distractions or the limited processing capabilities of both the device and the annotator.

Although video annotation on mobile devices is an exciting avenue for research, for most of this thesis, we consider annotations created with a computer mouse. Mice devices are widely available when creating annotation systems and many people are familiar with the device, making our results relevant to wide audience.

2.4 Conclusion

Video annotation is often a slow process. The process should be quick to support realtime video annotation, and quicker still if large archives of video are to be repeatedly annotated for different annotation contexts.

To speed up the annotation process, we focused on methods to create and correct point-based spatio-temporal paths. The paths trade spatial density for reduce annotation time. The paths are useful for a number of reasons, such as “first-pass” annotations, or to draw attention to video objects.

We organised methods to create point-based paths so annotation tool designers can choose an appropriate method for their annotation context. Additionally, our arrangement supports researchers by illustrating the annotation method space, provides a vocabulary for discussing annotation methods and identifies some of the shortcomings of existing methods.

We considered a pursuit method as a realtime, manual, data-independent and memory-less method - although the method could be combined with automation in an semi-automated system. In the next chapter, we define the pursuit method in more detail.

Chapter 3

Pursuit Method Definition

A pursuit method quickly creates point-based paths for video annotations in-time with video playback. Other methods to create point-based paths typically interrupt video playback. Additionally, other methods require additional annotation time to control video playback and navigate between video frames. Furthermore, some automated methods struggle with consumer video conditions that are well supported by the manual pursuit method.

In this chapter, we clearly define the interactions of a pursuit method. We illustrate how a pursuit method can produce paths for typical uses of video annotations. We also present feedback from annotators using a pursuit method prototype.

3.1 Pursuit Method Interactions

Most video annotation methods typically rely on pointing interactions to define paths. A pursuit method is based on a well known pursuit interaction, previously studied for weapons research [52] and as a psychological performance measure [77], although previously unexplored for video annotation. Poulton describes pursuit interactions with two aspects: [72]:

- Target acquisition: Manual movements to reach a moving object, such as reaching to touch a box on a conveyor belt.
- Target tracking: Manual movements to match the time dependent position of a moving object, such as following an aircraft with a searchlight. Poulton notes that pursuit is distinct from a self-paced tracing interaction.

Video annotation methods to create a point-based path must satisfy two requirements: (1) specify start and end times and (2) define a sequence of points between the start and end times. Here we describe how a pursuit interaction can satisfy these requirements, with illustrations in Figure 3.1.

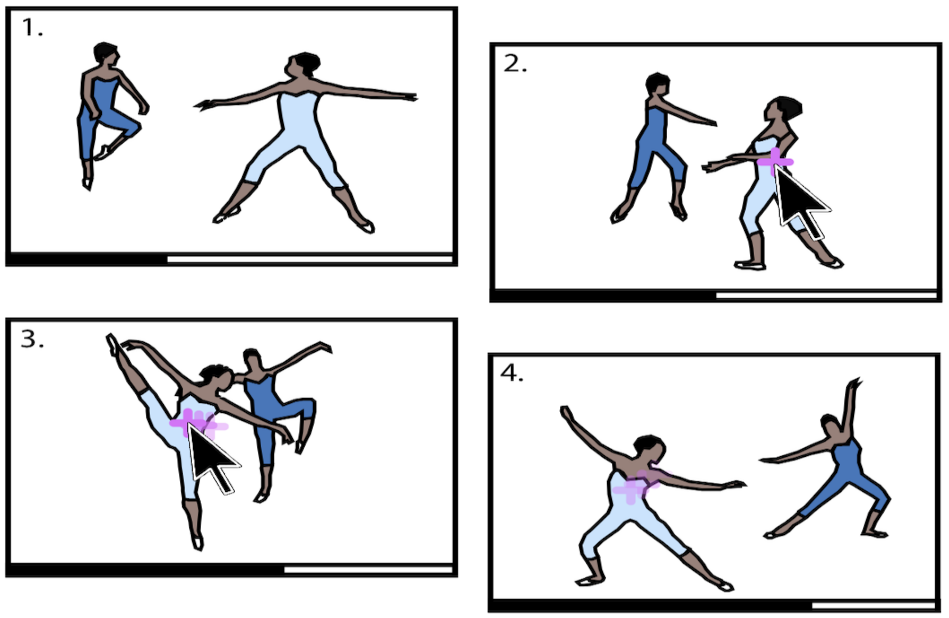


Figure 3.1: Video annotation with a pursuit method. (1) Two figures are moving in video. (2) A sequence of points starts when the annotator depresses the mouse button. (3) The sequence follows the mouse as the annotator pursues the video object (points in previous frames are illustrated for clarity with onion skins). (4) The annotator releases the mouse button to end the path.

3.1.1 Specify Start and End Times

To specify the start time for a point-based path, an annotator uses a mouse button action - the start of the path coincides with the currently displayed frame when the mouse button is depressed.

To more accurately specify the start time of the path, video playback can be paused. The annotator can navigate to the start time with a video timeline and playhead. On depressing the mouse button, video playback can resume for the pursuit.

Alternatively, video playback can be uninterrupted, and the annotator can chase and acquire a video object before indicating the path start time by depressing the mouse button. When video playback is not interrupted, pursuit interactions would be suitable for realtime video annotation.

To specify the end time of the path, an annotator releases the mouse button. On releasing the mouse button, video playback can be paused to allow annotators to review or correct the path.

3.1.2 Define a Sequence of Points

Point locations are defined by sampling the position of the mouse during the pursuit. The sampling rate can be less than the video frame rate if interpolation provides locations between samples.

Ideally, the video object should be acquired before point sequence is started. Pausing the video as described above allows the annotator to more accurately acquire the video object.

The point is positioned on a portion the video object, such as the head or torso of human object. The portion of the object typically has a complex shape, although many annotation applications simply require that points of the annotation path lie within the portion. This simplifies pursuits of large objects, since the annotator may not have to frequently reposition the mouse.

Annotators can handle object occlusions by positioning the pointer at their estimate of the object position while occluded, or by ending the path when the object disappears and resuming the path when the object reappears - similar to situations when the object leaves the field of view.

The pursuit can be started with a mouse up event so that mouse movements are sampled while the mouse button is released. However, such an interaction would be unsuitable for touch screen devices.

3.2 Pursuit Method Examples

3.2.1 Handling Challenging Video Object Movements

We demonstrate how manual pursuit methods are useful in video conditions that challenge automated methods.

Occlusion is difficult for automated methods because when the object reappears, (1) the object may be difficult to recognise as the same object and (2) the object can reappear in an unexpected location.

In the example presented in Figure 3.2, as two soccer players pass each other, they also change direction. To make matters worse, the soccer players have a similar appearance. However, an annotator following the play and using a manual annotation technique can often distinguish the object movements and track the player with a pursuit method. In situations where the occlusion is confusing, such as when team players quickly intersect and

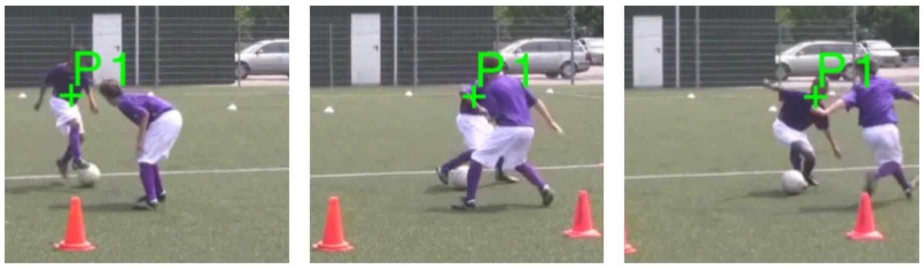


Figure 3.2: An example of video object occlusion that automated methods find difficult. Both players are wearing similar clothing. As one of the players is occluded, the players change direction. We manually annotated one of the players with a pursuit method.

separate, the video playback could be slowed down to allow annotators additional time to distinguish the players.

Shape changes are also difficult for automated method since an object can (1) divide - such as when a person removes clothing or (2) merge - such as when a person climbs into a car.

In Figure 3.3, a "compound" video object is composed of a horse and rider with similar colour characteristics. As the rider dismounts, she changes pose from sitting to standing and her shape changes. Additionally, the rider moves away from the horse. An annotator using a manual annotation method is primarily interested in tracking the person rather than the horse and can easily distinguish the two objects.

3.2.2 Integrating With Other Annotation Tasks

We demonstrate a pursuit method as part of a prototype annotation system in Figure 3.4. The example also illustrates a number of the uses for point-based paths.

To identify the video object once the point-based path has been created, the annotator selects the spatio-temporal entity - by double-clicking in this example - so that the point-based path forms an interactive element. A popover is presented so the annotator can enter metadata for the video object, including hyperlinks to additional information. Video object identifiers are rendered to follow the video object while the object is selected.

The example also illustrates how the point-based paths can support subsequent navigation within video. The timeline has been shaded to indicate

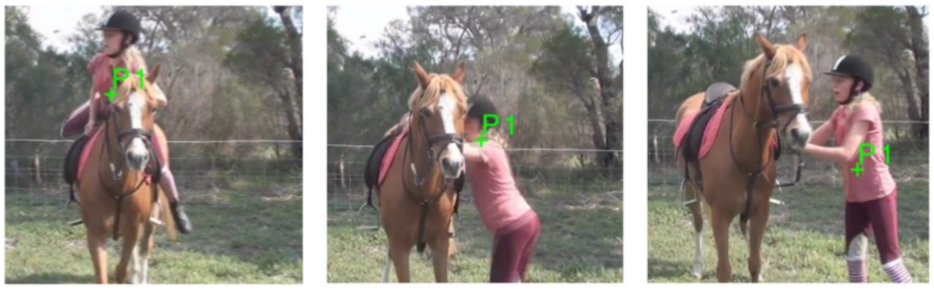


Figure 3.3: An example of video object shape change. To begin, the rider is mounted on the horse. As the rider dismounts, her size and shape changes. We manually annotated the rider with a pursuit method. Given that the rider should be annotated, if the point remained inside the shape of the horse, the distance between the point and the shape of the rider would constitute an error, since the point did not lie inside the rider shape.

the presence of an object of interest, so an annotator can quickly jump to the annotation time where the object appears with the playhead slider. Like some other methods, a pursuit method provides a "top-down" approach to annotation, so that paths can be quickly created for refinement later (Appendix H.

The path created by a pursuit method can be offset from the pointing device, as illustrated in Figure 3.5. Additionally, video playback rate during pursuits can be adjusted - both because objects are moving too slowly or too quickly (Figure 3.6).

3.3 Informal Prototype Evaluation

We implemented both mouse and touchscreen interactions for the prototype illustrated in Figure 3.4 and described in Section 3.2. Although our informal evaluation of the prototype was on a touch device, annotators currently have access to more mouse-based annotation systems and the remainder of this thesis focusses on analysis of mouse-based pursuits. Although the findings of our touch-based evaluation may not be directly comparable with results in the remainder of the thesis, they served to refine the pursuit method and raised important research questions.

We implemented pursuit and the Hold-Pause video annotation interaction described in Section 2.2. We automatically slowed down the video

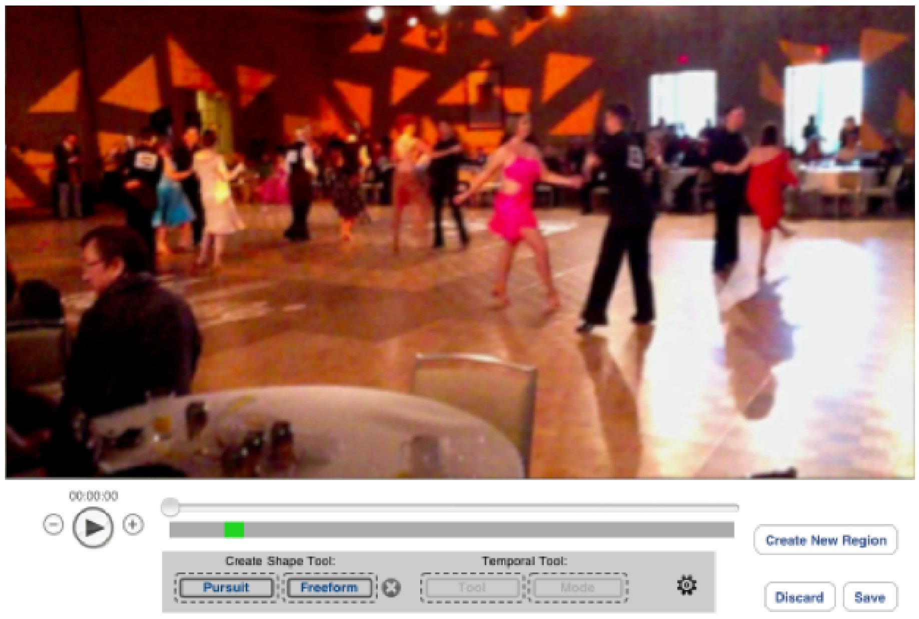


Figure 3.4: A prototype interface to combine video object identification with a pursuit method for video annotation. In this case, an annotator creates point-based paths to correspond to dancers. The interface indicates the presence of the object in video segments with a green bar below the playhead. Infrequently used functions such as metadata editing are normally hidden and accessed from the presence bar below the timeline. In addition to playback and timeline navigation, we provided frame-by-frame navigation controls. The video shown was recorded from a hand-held consumer camera.

during the pursuit interaction. We also included a third interaction - Auto-Advance, similar to YouTube Annotations to create shape based spatio-temporal volumes [3]. Auto-Advance automatically advanced the video between the equally spaced keyframes in which an annotator created a shape, but otherwise paused video playback while the annotator created shapes. Hold-Pause and Auto-Advance are alternative manual annotation methods to create spatio-temporal entities.

We demonstrated each of the interactions to four computer interaction experts (unaffiliated with the research) and asked the experts to use each interaction in turn to position shapes to draw attention to specific dancers in video from a hand-held consumer camera. After they annotated with finger

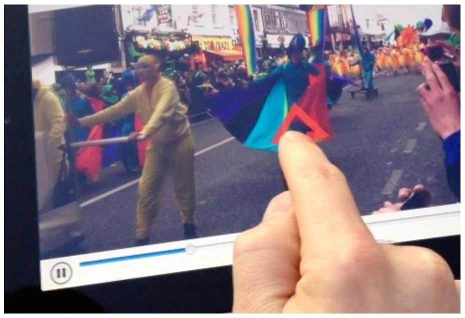


Figure 3.5: An annotator draws attention to a video object with a touch screen pursuit. This example illustrates an "offset tracking" mechanism to avoid finger occlusion issues.

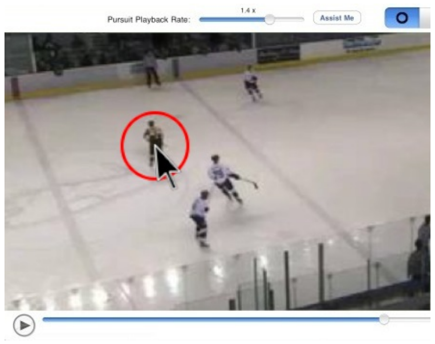


Figure 3.6: A video object pursuit illustrating adjustment to video playback rate. In this example, the annotator is placing a circular visualisation on the point-based path.

on a touch-screen, we asked for their preference between the interactions.

Two of the experts preferred the pursuit interactions. One explained that the pursuit interaction had a greater sense of control, although he cautioned that keeping his finger down during a long pursuit could get tiring. One expert preferred Auto-Advance, and another preferred Hold-Pause, explaining that the interactions seemed to create paths as accurate as the other interactions, but required "less work".

All of the participants were experts in the field of computer interaction, and would use computer interfaces on a daily basis - it is likely that they would learn to use the interaction toolset quicker than novice computer users. Additionally, participants preference for methods is subjective. In the following chapters, we present more objective studies to measure the time and accuracy of paths created by pursuit methods. However, we note that none of the experts considered the annotations they created with a pursuit method as inferior to annotations created by the other methods.

3.4 Discussion

We have defined and illustrated pursuit interactions for annotation tool designers. In comparison with other interactions, a pursuit method supports realtime path definition. A pursuit method usefully combines video playback control with simple interactions to quickly define spatial properties of a point-based path.

Like some other methods, a pursuit method provides a “top-down” approach to annotation, so that paths can be created for refinement later, perhaps with the curvature based merging described by Baudel [12]. Also like other methods, they can “offset track” objects on touch screens.

Paths created by a pursuit method can both constrain automated annotation methods to create content-based paths - in conjunction with region growing techniques like Magic-Wand [8] - or select from automatically generated path alternatives - like Goldman’s particle tracking system [38]. However, in this thesis we focus on a manual pursuit method and leave computer vision assisted pursuit as an exciting area for further research.

The method could position a shape on the point-based path. However, when positioning a shape, the annotator is more concerned with how the shape encloses the video object, rather than where the point-based path for the shape lies on the video objects. As a first step to demonstrating the validity of pursuit methods, we focus on how accurately annotators position points on video objects.

We prototyped the methods for a group of computer interaction experts. They used both a pursuit method and Hold-Pause to create annotations of dancers in consumer video. The experts found the pursuit method useful, even preferring the method in some cases.

However, the best method for a point-based path definition task will depend on a number of video content factors - such as the movements of video objects or scene lighting conditions. Ideally, annotators should choose a method for a particular video content. In the following chapters, we present quantitative data to support designers choosing a pursuit method. Additionally, results from our studies can be incorporated into toolsets that automatically recommend and configure methods according to detected video content.

In particular, the movements of video objects must be compatible with manual tracking movements for a pursuit method to be effective. We also note that movement speeds of a video object are related to video playback rate. In the next chapter, we develop a model to explain how pursuit accuracy is related to video object speed. With our model, we can both determine the feasibility of a pursuit method and also calculate how to adjust playback rate to modulate pursuit accuracy.

3.5 Conclusion

We have defined and illustrated a pursuit method for video annotation. A pursuit method for video annotation differs from other manual methods for video annotation because video playback is not interrupted during annotation. A pursuit interaction simply requires an annotator to track a moving video object in-time with video playback.

Although annotation tool designers may expect a pursuit method to be inaccurate for typical video object movements, our informal prototype evaluation suggests that the methods are useful for video annotation in consumer video contexts.

However, the evaluation raises some questions:

- Do the methods produce paths with similar accuracy to alternative methods? Although annotation in-time with video playback may dissuade the annotator from superfluous refinements, pursuit may prove too challenging for realistic video conditions.
- What video object movement factors affect path accuracy? Although the methods may not be suitable in some conditions, it may be possible to adjust video conditions so the methods become suitable. Additionally, how should video playback be adjusted to best support the pursuit method?
- Will annotators actually like a pursuit method? Even if the method is viable, the method may be onerous for annotators.

In the next chapters, we provide empirical evidence to answer these questions and demonstrate the viability of a pursuit method for video annotation.

Chapter 4

Peak Error Model for Pursuit Interactions

To investigate the important factors in pursuit accuracy, we followed a system identification approach [57]. Full details of the experimental setup and procedure are described in Appendix F. For clarity, we present a summary and discussion of the experiment in this chapter.

Following a system identification, researchers first choose suitable impulses for the system. We generated two impulse stimuli representative of a hard pursuit condition - representative of situations when a video object completely reverses direction abruptly and unpredictably.

Next, we examined the responses of the system. By assuming that the system is linear, the responses of the system to more complex stimuli can be expressed in terms of the responses to simple impulses. We present graphs of the responses to simple impulses in Figure 4.2.

Finally, we attempted to deduce a dynamic model of the system from the impulse responses. With such a model, the responses of the system to a wide range of stimuli can be predicted. Unfortunately, we failed to identify a simple dynamic system to explain the responses. However, there are predictable characteristics of the responses, and we present one of the characteristics - peak error - in Figure 4.3.

Robinson examined impulse responses of eye pursuit movements [77]. Additionally, researchers have examined moving target acquisition for pointing movements [41]. However, researchers have not yet investigated the effect of target size on impulse responses of human *pursuit movements* with a manual pointing device. In this chapter, we present a model for peak error during a pursuit that indicates that target speed and size explains a substantial portion of pursuit accuracy.

4.1 System Impulses

We created two system impulses with target movements on two similar paths - (1) a synthesised path and (2) a derived path based on hockey player movements in video. The derived path is illustrated in Figure 4.1. We transformed or otherwise randomised the paths so that target movements along the paths were unpredictable. We report full details of the procedure in Appendix F.

Video objects following these paths are hard to pursue since human movements to track a moving target must decelerate and accelerate a limb and a pointing device. An abrupt direction change to follow a target that completely reverses direction requires more tracking force compared to path changes in other directions.

Additionally, unpredictable movements are more difficult to pursue that predictable movements. When the direction changes are unpredictable, annotators cannot prepare for the change.

Both paths contain an abrupt direction change, but otherwise a target would move smoothly along the paths at a constant speed. The abrupt direction change reverses the target direction and is effectively an acceleration impulse at the time of the direction change.

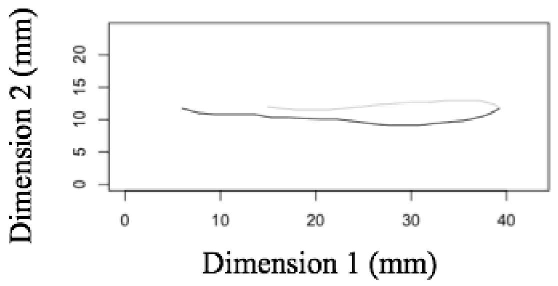


Figure 4.1: Derived video object movement path from video on hockey player movements.

4.2 Impulse Responses

We assume that human pursuit processes can be described as a linear time-invariant system. Other researchers often assume a linear time-invariant

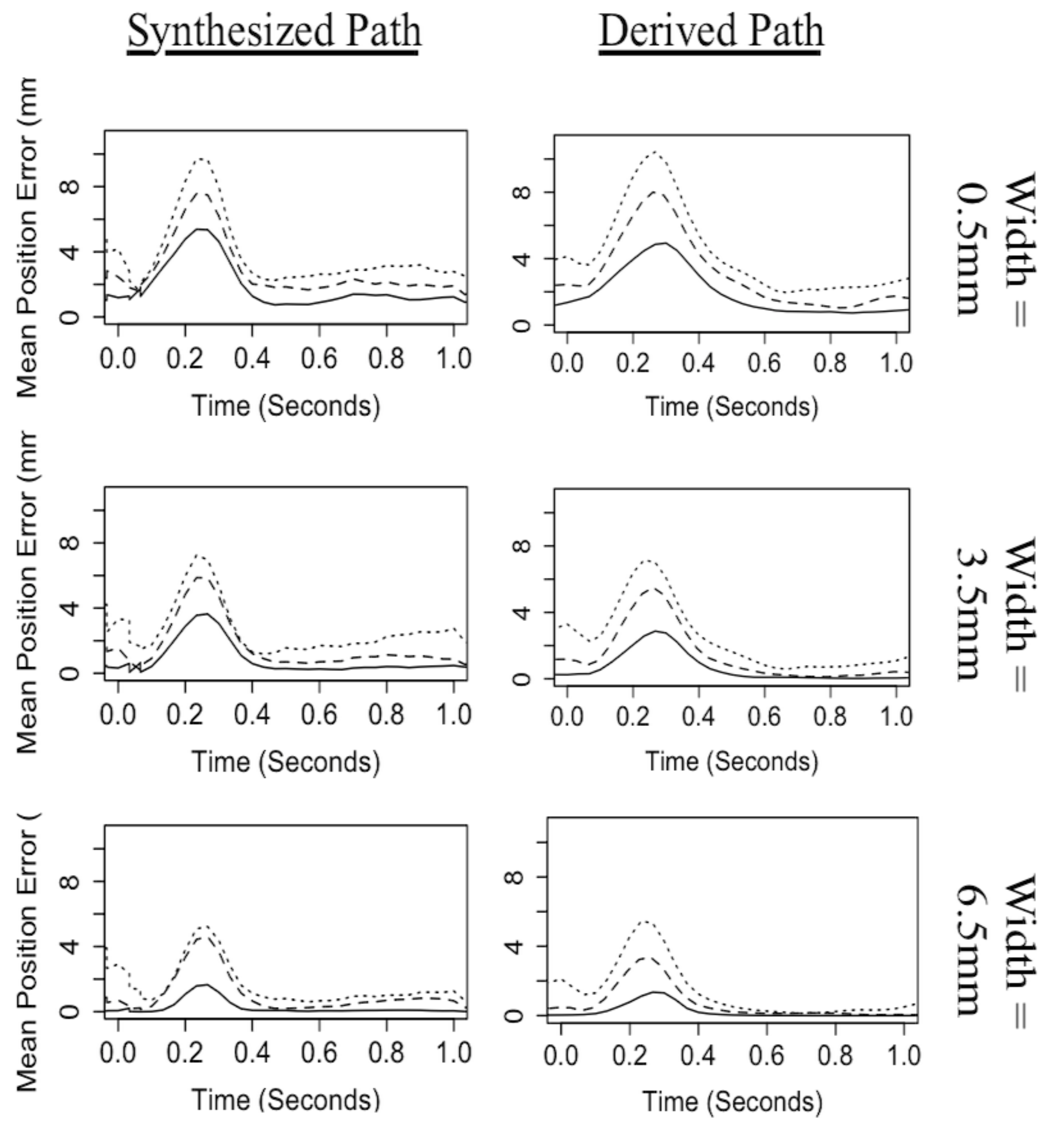


Figure 4.2: Pursuit impulse responses for three object sizes and three object speeds (dotted line: 35mm/s, dashed line: 25mm/s, solid line: 15mm/s). The distances are in terms of "input space" - the distances moved by the users hand. The pursuit target changes direction at $Time = 0$. Pursuit error appears to change linearly with speed and target size.

(LTI) model of human motor systems. Plamondon argues for a log-normal impulse response model for rapidly aimed hand movements [70]. Navas investigated LTI models of rotary hand movements [66]. Even when researchers caution that motor systems are non-linear, they also suggest that linear models describe target acquisition movements well [44].

We gathered measurements of error between a mouse pointer and a video object from ten subjects (described in Appendix F). Each subject pursued video objects moving at three different speeds and with three different sizes - nine conditions that were each repeated ten times. Averaged pursuit error impulse responses for each condition of speed and size are presented in Figure 4.2. Approximately sixty samples per condition are presented in the synthetic case due to our experiment configuration.

The measurements are plotted in terms of ”input space” - the distances moved by the users hand when positioning the mouse pointer. The distances are similar to those reported in studies of steering task performance [6].

After a direction change, the annotator initially “loses” the target. When annotators make a choice for a new pursuit movement, the error falls as they match the new video object direction (our measurement of error is described in the next section). This is consistent with the average reaction times for choices with few options as modeled by Hicks [43].

With increasing speed, the peak pursuit error also increases. With increasing size, peak pursuit error decreases. The responses are not simple first of second order responses. However, they do appear similar to log-normal responses reported by Plamondon [70].

4.3 Peak Error Model

Peak error is the largest deviation of the tracking pointer from the pursuit target. The measure is a more “honest” measure of accuracy than average error during a pursuit since it captures the worse behaviour of a pursuit method. We made a number of measurements per subject and condition so that a particularly bad pursuits are balanced out.

To investigate the worst cases of pursuit error, we took the largest error measurement between 0.2 and 0.3 seconds after the direction change and present these peak errors in Figure 4.3.

With an analysis of variance, we confirmed that (1) as speed increases, peak error increases ($p < 0.05$) and (2) as size increases, peak error decreases ($p < 0.05$, reported in Appendix F). Although our experiment did not have power to shed light on an interaction effect between target speed and size,

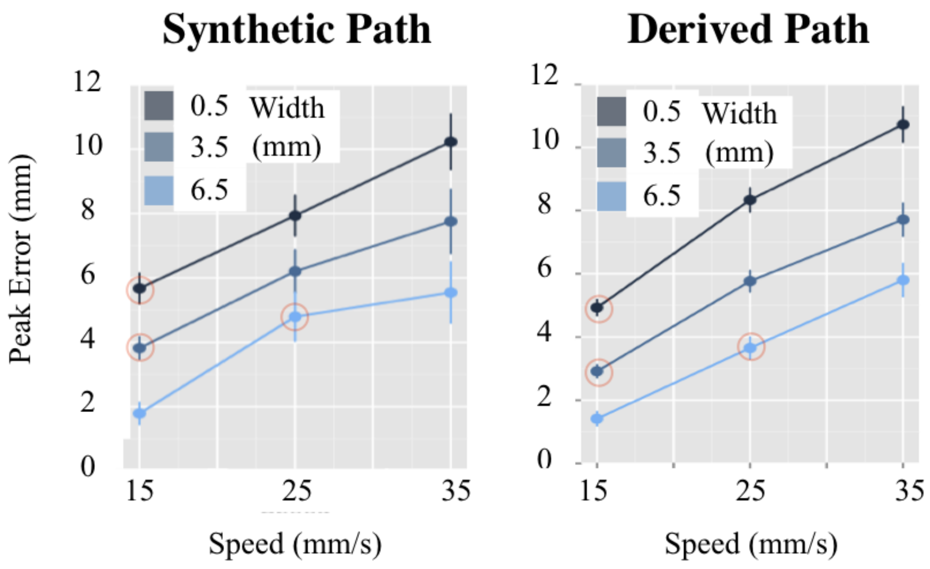


Figure 4.3: Peak error measured between 0.2 and 0.3 seconds into the pursuit impulse response, with 95% confidence intervals. Differences between speed and size conditions within both paths of the experiment are statistically significant. Only three conditions are significantly different between paths of the experiment (indicated with red).

Figure 4.3 suggests the effect of an interaction would be small relative to the main effects.

Bonferroni post-hoc comparisons revealed significant differences among all conditions (p <0.05) within paths. With Welch's t-test to accommodate uneven sample sizes, we determined that measures of peak error were significantly different between experiment paths for only conditions of $Size$ = 0.5 & $Speed$ = 15 (t = -2.71, df = 76.4, p <0.05), $Size$ = 3.5 & $Speed$ = 15 (t = -4.46, df = 89, p <0.05), $Size$ = 6.5 & $Speed$ = 25 (t = -2.87, df = 83.8, p <0.05).

We hypothesised a linear model for the relationship between peak error and video object speed and size. Assuming a linear human movement system, increases to target speed will linearly increase peak error. Similarly, increases to pursuit target size will linearly reduce peak error - a larger target size will simply reduce the distance from the edge of a target to the pointing device:

$$PeakError = k1.Speed + k2.Size + c \tag{4.1}$$

We fitted the linear model with a least squares technique to the data and report the regression coefficients in Table 4.1. The coefficients are significantly different from zero, indicating that target speed and size have a significant effect on peak pursuit error. The R-squared values indicate that speed and size explain 79% of the variance of peak pursuit error when within subject variability is excluded.

For a target moving with speed 20 mm/s and with width 2 mm, our model predicts peak pursuit error will be $0.23 \times 20 - 0.69 \times 2 + 2.60 = 5.82$ mm on average. Additionally, when the speed of a the target increases by 1 mm/s, our model predicts that peak pursuit error will increase by 0.23 mm, assuming target size is held constant.

4.4 Discussion

For a challenging target movement path, we have shown how peak pursuit error (1) increases with target speed and (2) decreases with target size. A linear model of peak pursuit error indicates that target speed and size have a significant role in peak error.

However, the estimate standard error indicates there is substantial variability between model predictions and likely peak errors, relative to the sizes of targets in our experiment - 95% of pursuits will have peak error within $2 \times 1.32 = 2.6$ mm of the predicted peak pursuit error. The source of the variability may be the differences in starting position during the trials since such an effect would both increase and decrease peak error.

Additionally, the regression indicates large intercepts relative to typical peak errors. The source of the constant error may be variability in subject reaction time to direction changes since such an effect would only increase peak error. Furthermore, a Breusch-Pagan test revealed that the measures of peak error were heteroscedastic (varying variance for different conditions) - peak error varies more with increasing speed.

We also caution researchers that our study does not isolate confounding effects of target speed and acceleration. During the experiment, although we varied target speed, we also simultaneously varied target acceleration - a confound that could be addressed in future studies.

Our model is only defined over a limited range of sizes and speeds - further studies could extend the model to larger movements with studies of pursuits on touch screens. For larger movements, reaction times will

	R^2	Estimate Standard Error (mm)			Coefficient Standard Error
Subject Trials Not Averaged					
Synthesized	0.45	2.59	$k1$ (1/sec)	0.20	0.01
			$k2$	-0.66	0.05
			c (mm)	3.21	0.41
Derived	0.64	2.03	$k1$ (1/sec)	0.25	0.01
			$k2$	-0.73	0.03
			c (mm)	2.01	0.24
Combined	0.56	2.27	$k1$ (1/sec)	0.23	0.01
			$k2$	-0.70	0.02
			c (mm)	2.43	0.21
Subject Trials Averaged					
Synthesized	0.77	1.29	$k1$ (1/sec)	0.20	0.02
			$k2$	-0.66	0.06
			c (mm)	3.22	0.48
Derived	0.81	1.33	$k1$ (1/sec)	0.25	0.02
			$k2$	-0.73	0.06
			c (mm)	2.01	0.49
Combined	0.79	1.32	$k1$ (1/sec)	0.23	0.01
			$k2$	-0.69	0.04
			c (mm)	2.60	0.35

Table 4.1: Parameters for predictive model of pursuit accuracy. When subject trials are not averaged, the prediction explains 56% of the variability in the data, but contains a substantial error relative to the sizes of the objects considered in our experiment. When subject trials are averaged, the prediction explains 79% of the variability in the data, suggesting that within subject variability is a substantial source of variability.

still be similar, but movement accuracy variability increases with movement distance as modeled by Schmidt [79].

Some video object paths may leave the annotation screen and then reappear - in our experiment, we only examined pursuit once a video object is acquired. Other research has addressed target acquisition, including moving target acquisition [41].

Some subjects reported fatigue during the trials, but our experiment could not shed light on how this affected pursuit method accuracy. Further studies could establish how prolonged pursuits use may improve with practice or decline with fatigue. We conducted our study with practiced mouse users, but the regression coefficients would likely be larger for older annotators or those with movement impairments.

Lastly, the study was for a particularly hard video object movement. For video objects on smoother and more predictable paths, peak pursuit error should be less than our model predicts.

4.5 Conclusion

Our study has shown that target speed and size has a significant effect on peak pursuit error. Additionally, over the range of speeds and sizes we considered, there is a linear relationship between target speed, size and pursuit error.

Our model does not explain the full complexity of human pursuit movement processes. Instead, the model indicates the substantial contribution of target speed and size to pursuit peak error - an honest measure of pursuit accuracy.

However, we raised a number of research questions:

- Is typical peak pursuit error acceptable for video annotation? Is a pursuit method actually viable for realistic speeds and sizes of video objects?

- Will the same relationship between video object speed, size and pursuit accuracy be present for more realistic video conditions? Is the pursuit model useful to adjust video playback speed?

In the next chapter, we develop a second study to demonstrate the validity of a pursuit method for a range of typical video object speeds and sizes.

Chapter 5

Pursuit Method Performance

We demonstrate the validity of a pursuit interaction for annotating typical consumer video with a controlled laboratory experiment.

Consumer video conditions vary considerably. For instance, in some video objects occlude each other, while in other video, the objects move on clearer paths. To control video object variability during other computer interaction experiments, some video interaction researchers create synthetic video [32]. However, we adopt another popular approach - we choose sequences of video with some consistent and controlled properties, but with video objects that are more realistic than synthetic video [48].

We chose video of a basketball game as a compromise between controlled and realistic video conditions. Objects in the video frequently occlude each other. Furthermore, the video combines object movement with camera movement. However, the video contains objects of roughly consistent size, and the objects travel with similar speed on average - we focussed on players either walking or slowly running.

In this chapter, we show how pursuit methods produce viable point-based paths for these consumer video conditions. We report both (1) pursuit performance for typical video conditions and (2) pursuit performance relative to a similar alternative video annotation technique.

5.1 Hypotheses

We address four research questions:

- Is a pursuit method viable for typical speeds and sizes of video objects?
- Is the pursuit model from our earlier study useful for adjusting video playback speed?
- How does a pursuit interaction compare with an alternative video annotation technique?
- Will annotators actually like a pursuit method?

We compare our interaction with the Hold-Pause method that requires video pausing, introduced in Chapter 2. We chose Hold-Pause as a method that creates similar point-based paths, with similar mouse based interactions to pursuit. We represent the basic interactions of Hold-Pause in Figure 5.1. However, Hold-Pause interrupts video playback and the path between points is linearly interpolated.

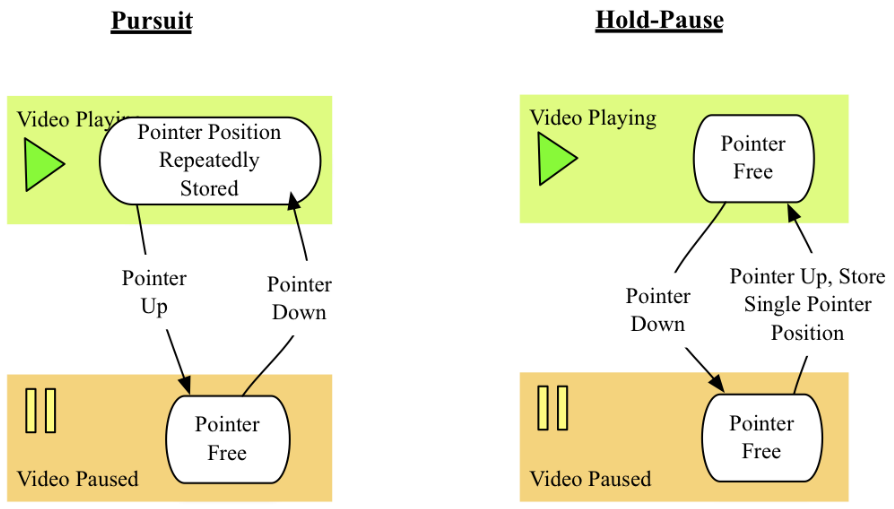


Figure 5.1: Interaction diagrams for both the pursuit and Hold-Pause methods. The position of the pointer is repeatedly sampled while the video plays in the pursuit interaction. In contrast, only a single position point per pointer up event is stored for the Hold-Pause method - the individual positions are joined together into a point-based path with a linear interpolation. Since the video is paused while points are positioned with the Hold-Pause method, the points can be positioned more accurately - although the linear interpolation may not follow the object movements well. Furthermore, the Hold-Pause method interrupts video playback, slowing down the annotation interaction.

- (H1) Pursuit interaction will be quicker than Hold-Pause for all conditions.

  We hypothesize that since pursuit will not interrupt video playback during annotation, the method will be quicker than Hold-Pause.

- (H2) Both interactions will have the same annotation accuracy.

For slow video object speeds, we hypothesize that the corresponding accuracy of the pursuit method will be low and sufficiently accurate to compare favourably with Hold-Pause. At faster speeds, both Hold-Pause and pursuit accuracy will suffer as annotators fail to match object movements. Additionally, for video object movements combined with camera movements, objects paths created with the Hold-Pause method may not be captured well with a linear interpolation.

- (H3) Annotators will find the annotations created by both methods equally satisfactory.

  For typical video object speeds, we hypothesize that video annotation accuracy with a pursuit method will be satisfactory. Additionally, we hypothesize that annotations with a similar accuracy created with Hold-Pause also be satisfactory - but annotators may be less satisfied with the time needed to create the annotations.

- (H4) Annotators will prefer the pursuit interaction that does not interrupt the video playback.

  We hypothesize that pursuit will not be too challenging for realistic video conditions. Annotation in-time with video playback may dissuade the annotator from superfluous refinements, and they may find they create annotations more quickly than with Hold-Pause.

5.2 Experiment Design

5.2.1 Subjects

Ten novice annotators, two female, average age 28.8 years. All subjects were right handed, with normal or corrected to normal vision, and none reported colour deficient vision. Two subjects reported playing computer games with a mouse more than an hour a day, two reported less than an hour a week and two others less than an hour a month. Eight subjects reported using a computer mouse more than an hour a day. We provided a training period and regular breaks. Subjects were compensated CAD$10 for their participation and encouraged to annotate quickly and accurately.

5.2.2 Independent Variables

Conditions of interaction type (x2) crossed with video object speed (x3) and video object size (x3) and video object path (x4). We refined speeds and

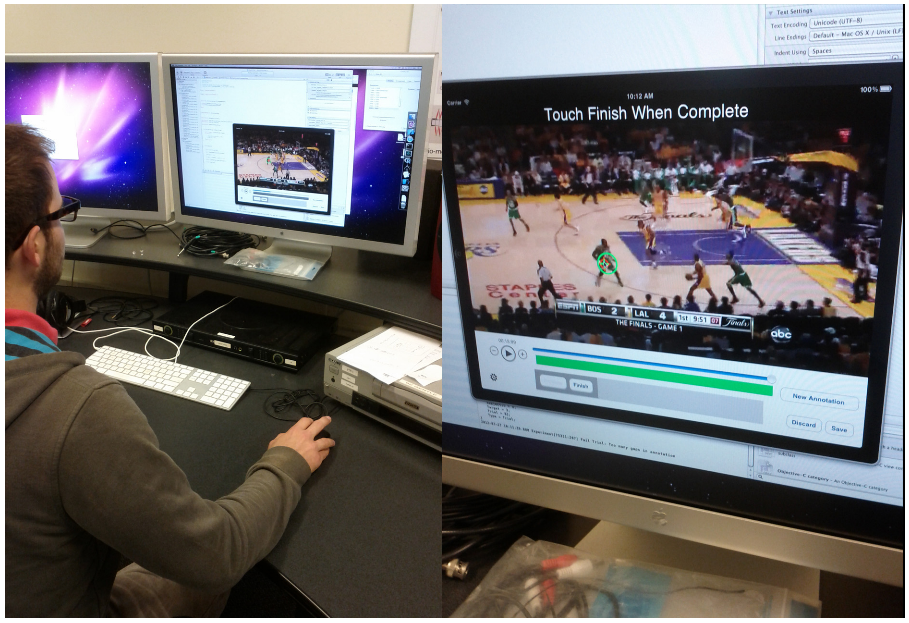


Figure 5.2: Experiment subject sitting a typical distance from the experiment apparatus. We also illustrate a typical screen image during the experiment.

sizes in pilot studies to realistic values for typical video object movements. Sizes were 0 mm, 0.625 mm and 1.25 mm, speeds were 1.075 mm/s, 2.15 mm/s and 3.225 mm/s. Subjects completed a block of trials for a single interaction, but with random speed and size condition order per measure block. We counterbalanced interaction order between subjects.

We chose four different video object paths so that (1) objects on the paths stay within the video frame for the duration of the annotation and (2) the paths that were similar in duration, each eight seconds long. The objects appearance and movements were similar, moving at 2.15 mm/s on average when video playback speed was normal. We adjusted video playback rate to 0.5 and 1.5 times normal to control video object speed.

Our video had manually created ground truth annotations describing bounding boxes surrounding each player from a college basketball game. To control for the size of the video objects, we overlaid circle visualisations of three sizes on the ground truth path so that the visualisations coincided with the basketball players - for the 0 mm condition, the visualisation was a point. Subjects were instructed to keep their annotation paths within the circles as accurately as they could. We also collected data for a number of trials when no visualisations were provided.

5.2.3 Dependent Variables

We gathered two repeated measures of annotation time and annotation accuracy from each subject for each experiment condition (144 measurements).

We derived ground truth paths from the bounding boxes centers so that objects on the paths lay on the midriffs of players in the video. We measured annotation accuracy with the error between the annotation path and the ground truth - from annotation path points to the edges of the visualisations centered on ground truth bounding box centers - sampled at video frame rate.

We measured time to create the paths as the time from the first mouse event of the trial until the last mouse event before the annotator indicated they had completed the trial.

5.2.4 Task

The task for annotators was to create point-based paths for different interaction conditions. Annotators were instructed to position a single point inside video object circle visualisations, or as close as possible to the "zero size" objects (objects with a point visualisation centered on the object, see above).

To correctly start each trial, the annotation point had to lie near the video object. Subjects repeated trials that did not provide annotation for at least 95% of the video. At the end of a trial, if these requirements were not met, the apparatus would prompt the subject to repeat the trial. Subjects required consistent experiment times - repeating approximately three trials per experiment.

Each interaction type was clearly distinguished with a different cursor style. We gave experiment subjects two second previews of the video paths before annotation to indicate the object they should annotate and help improve annotation when an object started moving.

5.2.5 Apparatus

Subjects sat approximately 75 cm from an Apple Cinema display 68.8 by 54.3 cm, with monitor resolution 2560 by 1600 pixels (Figure 5.2).

Subjects used a Microsoft Comfort mouse, mouse acceleration was disabled and the control-to-display ratio was set to 0.092. This ratio is considered low, but provided acceptable control of the pointer and is similar to ratios examined by Accot [6]. Like the measurement in the previous chapter, error and distance measurements are plotted in terms of ”input space” - the distances moved by the users hand when positioning the mouse pointer.

Enough desk space was provided to subjects to avoid mouse “clutching”. We wrote the experiment software with UIKit for iOS 4.3. We hosted the experiment on an iPad simulator on a Mac Pro with 16 GB of RAM, running dual quad-cores at 3.2 GHz, and containing a NVIDIA Quadro FX 5600 with 1.5 GB of video RAM.

5.3 Results

Figure 5.3 presents error profiles for annotations created with the pursuit interaction for video object paths in the study. Each of the profiles is an average from the paths of all the subjects, each subject following the path twice. Although the error profiles are not as “clean” as in the previous study, peak error increases with video objects speed and decreases with video object size.

To illustrate the relationship between peak error and video object speed and size, for both pursuit interactions and the Hold-Pause keyframe interaction, we derived measures of the peak error for each path. In the analyses that follow, each path is divided into two and we calculated the peak error of each half for each trial. This yields 16 measures for peak error (4 paths x 2 halves x 2 repeated measures) for each condition for each subject. We plot the average peak errors in Figure 5.4.

5.3.1 Peak Error Model

We analysed the relationship between peak error and video object speed and size for annotations created with a pursuit interaction (retaining between subject variability in the peak errors). In Table 5.1, we present the results of a linear regression with a model

$$PeakError = k1.Speed + k2.Size$$

.

The coefficients are significantly different from zero, indicating that target speed and size have a significant effect on peak pursuit error. The

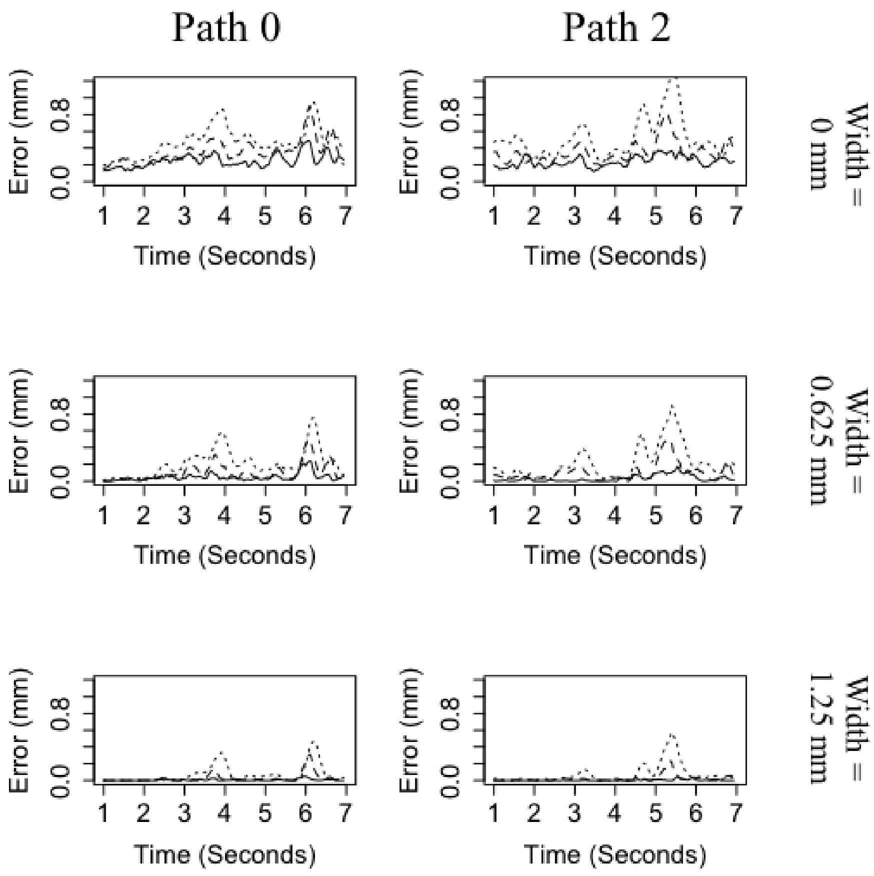


Figure 5.3: Error profiles for annotations created with a pursuit method for the two of the video object paths in our study. The dotted lines are for video objects moving at 3.225 mm/s, dashed lines for video objects moving at 2.15 mm/s and solid lines for video objects moving at 1.075 mm/s.

R-squared values indicate that speed and size explain between 88% of the variance of peak pursuit error, so we accept our model as a good description of the data observed.

Null hypothesis: there is no difference between the correlation coefficients in Chapter 4 and Chapter 5.

With Welch t-tests, with corrections to the degrees of freedom as described in Glass [37], we determined no difference between coefficient for

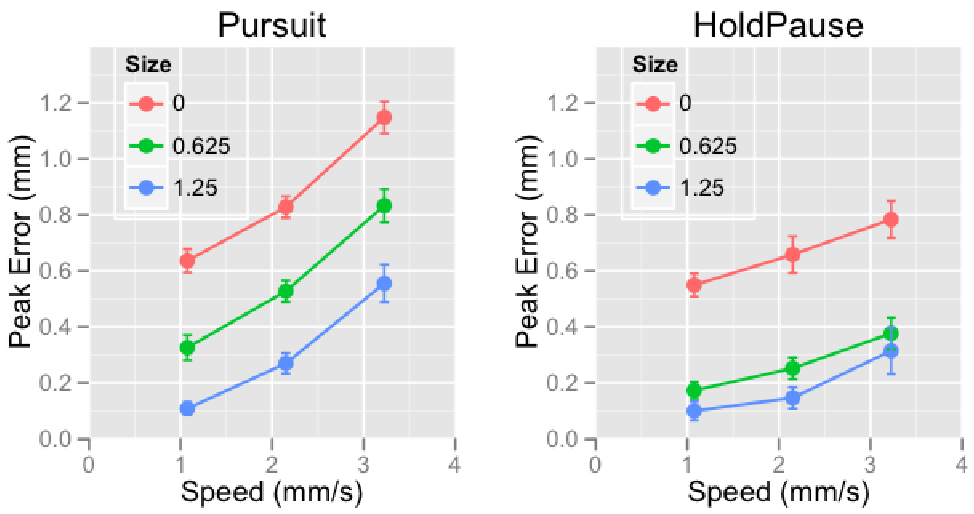


Figure 5.4: Average peak errors among annotators for both pursuit and Hold-Pause interactions (with 95% confidence intervals). These are not RMS errors, and are in terms of input space (user hand movements). There is a significant effect of interaction type on accuracy ($p < 0.05$) with Hold-Pause producing more accurate annotations for all conditions except the 1.25 mm, 1.075mm/s condition ($p > 0.05$).

speed (df = 232, t = 0.1, $p > 0.05$). However, we rejected the null hypothesis for coefficient for size (df = 244, t = 5.33, $p < 0.05$) and the intercept (df = 179, t = 6.34, $p < 0.05$).

Additionally, estimate standard error is lower in this second set of measurements, possibly because targets speeds were lower so reaction time to direction changes had less effect. For targets, 95% of pursuits will have peak error within $2 \times 0.11 = 0.22$ mm of the predicted peak pursuit error. (Although, like the previous results, the variance in peak error is heteroscedastic, so the prediction accuracy will vary with speed.)

5.3.2 (H1) Annotation Time

Null hypothesis: there is no difference between annotation times for both methods.

Repeated measures analysis of variance between the two conditions revealed a significant effect of interaction type (df = 1, F = 106.87, $p < 0.05$),

R^2	Estimate Standard Error (mm)			Coefficient Standard Error
		$k1$ (1/sec)	0.23	0.01
0.88	0.11	$k2$	-0.45	0.02
		$c(mm)$	0.38	0.03

Table 5.1: Regression coefficients for the peak error model developed in Chapter 4. The coefficient value for speed is similar to the previous study ($p > 0.05$), but coefficients for size and the intercept are different ($p < 0.05$).

and based on the confidence intervals illustrated in Figure 5.5, we accept our hypothesis (H1) that pursuit interactions are quicker in all conditions. (We assumed equal variances in annotation time for both interactions.)

For the slowest and largest video objects, we note that every second of video annotated with a Hold-Pause method required 1.20 seconds of annotator time (on average). In contrast, similar annotations created with a pursuit method required just 1.05 seconds of annotator time - a 12% reduction in annotation time (with a standard deviation of 5% reduction to annotation time). This relationship between interaction method and annotation time is illustrated in Figure 5.5.

5.3.3 (H2) Annotation Accuracy

Null hypothesis: there is no difference between accuracy of annotations made with both methods.

With a repeated measures analysis of variance test between the two conditions of interaction, we revealed a statistically significant effect of interaction type (df = 1, F = 61.98, $p < 0.05$), so we reject the null hypothesis.

Based on the confidence intervals and effect sizes in Figure 5.4, we reject our hypothesis (H2) that both pursuit and Hold-Pause will have the same annotation accuracy. However, with a t-test (assuming equal variance between conditions of interaction with repeated measures of accuracy), we determined that measures of peak error were not different for conditions of $Size = 1.25$ & $Speed = 1.075mm/s$ (t = -0.3832, df = 318, $p > 0.05$).

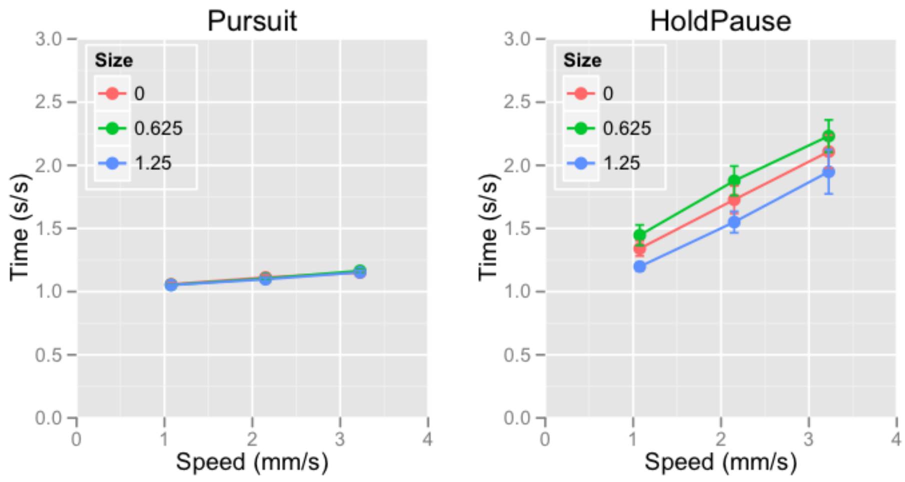


Figure 5.5: Average times to produce annotations for a second of video. The video object path was 8 seconds long for video objects moving at 2.15 mm/s, 16 seconds for 1.075 mm/s, and 5.33 seconds for 3.225 mm/s.

5.3.4 (H3) Annotators Satisfaction

We asked annotators "How satisfied were you with the accuracy of the annotations you made with the pursuit/Hold-Pause interaction?".

On a seven point scale from "Very Dissatisfied" to "Very Satisfied", subjects were "Somewhat Satisfied" with both interactions (pursuit 4.9, Hold-Pause 5.5, standard deviation in paired observations 1.65).

Null hypothesis: there is no difference between mean satisfaction for either method ($\mu_{PursuitSatisfaction} = \mu_{HoldPauseSatisfaction}$).

Based on a pairwise t-test, there is no significant difference between measures of satisfaction (df = 9, t = 1.09, p >0.05), although the power of this test is low at 0.57. Nevertheless, our hypothesis (H3) that annotators find the annotations created by both methods equally satisfactory is tenable.

5.3.5 (H4) Annotation Method Preference

We asked annotators "Which interaction did you prefer?".

Six subjects preferred a pursuit method. They described how the pursuit interaction required less work in order to produce annotations, possibly because they did not have to consciously decide when to create keyframes.

Some subjects preferred Hold-Pause because they felt it produced more accurate annotations, despite needing more time to produce the annotations.

Null hypothesis: there is no difference in preference between method.

With a Chi-squared test for categorical variables, we determined that there is no difference in preference for each method (df = 1, $\chi^2 = 0.4$, p >0.05). Unfortunately, our data does not provide evidence that annotators will prefer the pursuit interaction.

5.3.6 Additional Results

Repeated measures analysis of variance failed to show an effect of repetition on accuracy (df = 1, F = 0.1, p >0.05) or time measures (df = 1, F = 2.37, p >0.05). Additional tests failed to show an effect of condition order on accuracy (df = 1, F = 3.26, p >0.05). However, there is an effect of condition order on time measures (df = 1, F = 7.35, p <0.05). Closer examination revealed that annotators who used the pursuit method first were 0.16 seconds faster on average with the Hold-Pause method.

For the slowest video objects, repeated measures analysis of variance indicated an effect of visualisation on accuracy (df = 1, F = 25.0, p <0.05). In the absence of visualisations, annotators are approximately 0.1mm less accurate relative to our ground truth data. However, the analysis failed to detect an interaction effect of method type on accuracy - annotators are no more accurate without visualisations with Hold-Pause than pursuit (df = 1, F = 0.026, p >0.05).

When creating annotations with the pursuit interactions, seven subjects described either following or predicting target movements. Five subjects mentioned concentrating to better track the video objects.

With the pursuit interaction, two subjects felt that the camera movement affected their performance - the camera movements felt unpredictable and rapid. One subject felt fatigue affected their performance, and three others that they learned the player paths as the experiment progressed. Two subjects explained that the type of player movement affected their performance - sudden direction changes were harder to follow.

After using the Hold-Pause interaction, six subjects described pausing the video when the target is moving quickly. Three subjects described how pausing was unnecessary when the target was moving slowly. One subjects described how they would repeatedly click regardless of movements, whereas another explained how they clicked when the target was "stable".

With the Hold-Pause interaction, one subject also felt fatigue affected their performance. Another subject felt the Hold-Pause interaction required more “effort”. Two subjects mentioned the target movement also affected their performance - mentioning the camera movement again, and direction changes.

5.4 Discussion

Our goal was to gather evidence that pursuit interactions are suitable for quickly creating paths for consumer video annotations. We designed a controlled lab study to (1) compare pursuit interactions with an alternative interaction and (2) quantify the performance of pursuit interactions.

Our results provide additional support for the relationship between pursuit accuracy and video object speed and size from Chapter 4. The results suggest that by slowing down video playback, pursuit accuracy can be predictable modulated.

However, in comparison with the study in Chapter 4, our results show a different relationship between video object size and pursuit accuracy for typical video object movements. This may be explained by the irregular shapes of video objects - particularly since the visualisation and non-visualisation conditions differed in accuracy.

A pursuit interaction compares favourably with an alternative video annotation technique. A pursuit interaction outperforms Hold-Pause for 1.25 mm video objects moving at approximately 1mm/s. However, for smaller video objects, or video objects moving faster than 2mm/s or 3mm/s, the Hold-Pause technique produces more accurate annotations, although with longer annotation times.

Although we have measured the performance of pursuit methods for more realistic video conditions to Chapter 4, we have not accounted for other video factors such as changing lighting conditions or changing video object size. Further studies could introduce additional sources of variability into the video and use our results in a comparison.

Annotators considered pursuit to be as satisfactory as Hold-Pause. However, satisfying annotations in one context may not be satisfying in another. For instance, when many objects are in the video, point-based paths may have to be more accurate to correspond unambiguously to an object. In this instance, annotation tool designers could compare the expected peak error for pursuits with the average distance between objects to determine if the methods are suitable.

Although there is no clear preference for either interaction, annotators considered pursuit a valid video annotation method. However, our subjects were practiced mouse users, so further studies should investigate how a pursuit method fares with people who use mice less than an hour a day.

5.5 Conclusion

The pursuit interaction created point-based paths faster than the alternative method in all cases. However, the faster annotations came at the cost of annotation accuracy - Hold-Pause produces annotations more accurately than pursuit in all but one condition.

However, for 1.25 mm video objects moving at about 1 mm/s, a pursuit method not only produces paths as accurate and satisfactory as Hold-Pause, it does so 12% faster. In these situations, a pursuit method is a viable method for video annotation, and outperforms the alternative keyframe technique.

Additionally, the results in this chapter provide further evidence of a predictable relationship between peak error and video object speed during a pursuit. In the next chapter we demonstrate how annotators can use this relationship to modulate the accuracy of a pursuit method for video objects.

Chapter 6

Design Guidelines

In light of our model of pursuit accuracy, in this chapter we first compare the performance of pursuit interactions with Hold-Pause interactions and report our insights into the two methods.

Given that a pursuit method shows promise, we then present two guidelines on how to integrate the pursuit method into an annotation system.

The first guideline addresses situations when annotators want to slow down video playback to improve pursuit method accuracy. The second guideline is an optimisation for video annotation systems to address the effect of annotator reaction time on the accuracy of a pursuit method.

6.1 Choosing Pursuit Methods

6.1.1 Pointing Interaction Speed-Accuracy Trade-Off

Hold-pause trades annotation time for accuracy in two ways: (1) by allowing annotators time to perform accurate pointing movements at their own pace and (2) allowing annotators to refine path properties with a sequence of separate movements.

Annotators using Hold-Pause typically specify path properties with pointing movements. Although pointing movements are composed of a sequence of sub-movements, only the final sub-movement determines the accuracy of the entire pointing movement. However, the time for the pointing movement is the total time for all the sub-movements of the sequence.

Video playback is interrupted while points are specified with Hold-Pause. Annotators then specify path points with a number of movements per frame. Since video playback is interrupted, annotators are not under any time constraint to specify path properties as quickly as possible.

6.1.2 Pursuit Interaction Speed-Accuracy Trade-Off

A pursuit method trades annotation speed for accuracy directly and predictably. For video objects moving quickly (or small video objects), annotators have less time to react to movements - the relationship modeled in the previous chapter.

Since the method has a predictable trade-off between video object speed and path accuracy, the accuracy can be modulated by adjusting video playback rate. By reducing video playback rate, video objects can be slowed down so annotators have more time to react and follow video objects.

Even if video playback rate is adjusted, a pursuit method doesn't interrupt video playback, and so the flow of video review is maintained. Although the path accuracy may be rough, the method is suitable for realtime annotation of slow video objects.

6.2 Choosing Video Playback Rate

For fast video objects, annotation accuracy could be improved by slowing down the video playback. The playback rate should not be too slow that annotators spend too much time creating needlessly accurate annotations. With an appropriate playback rate, annotators can annotate in time with video playback and avoid interruptions to video playback.

A guideline on how to choose playback rate would eliminate trial and error choices of video playback rate, particularly for novice annotators.

For times when a pursuit method is not appropriate at all - perhaps video objects are moving too unpredictably and at high speed - annotators could be directed to alternative methods.

Additionally, annotators could spend less time annotating slow video objects by speeding up the video playback.

6.2.1 Playback Rate Formula

With the model presented in this thesis, we can estimate the peak error of mouse pursuits of video objects. If we assume that video objects are moving with constant speed, we can control the video object speed with video playback rate, and so modulate the accuracy of pursuits. Given a requirement for peak error, annotators should adjust playback rate according to the following formula, with constants ($k1$, $k2$ and c) from Chapter 5:

$$AveragePeakError > Rate.k1.Speed + k2.Size + c \tag{6.1}$$

For example, consider a designer creating a tool for annotation with a mouse-based pursuit method (with similar mouse characteristics as in the previous chapter). If video annotated with the tool should contain less than

0.4 mm peak error for at least 50% of pursuits, and the video contains objects of at least 0.625 mm in diameter, moving at 3.225 mm per second, then the designer uses the following calculation:

$$0.4 > \mathit{Rate} \times 0.23 \times 3.225 - 0.45 \times 0.625 + 0.38 \tag{6.2}$$

In this case, video playback should be reduced to 41% of the original playback rate and a pursuit method will produce acceptable results. Every second of video annotated will require 2.44 seconds of annotation time, comparable to the Hold-Pause technique in similar conditions.

In a similar case, but with objects of 2 mm in diameter, the video playback can be increased to 124% of the original playback rate and a pursuit method will produce acceptable results. Moreover, annotators will produce the paths quicker than if the video is playing at the original rate.

6.2.2 Prototype Example

To provide speeds and sizes of video objects for the formula above, an annotation system could pre-process video to extract approximate values. Alternatively, the annotator could set the speed and sizes values explicitly.

Here, we describe a mixed-initiative [10] approach to prompt the annotator to specify relevant object speeds and sizes. Such prompts could be in response to the system detecting new video content or scene changes.

With our prototype, annotators adjust the pursuit playback rate either explicitly with (1) a slider element or (2) by responding to prompts from a playback rate assistant (invoked manually in this prototype). Based on sizes and locations provided by the annotator, the assistant sets a playback rate so that 50% of pursuits have better than a required peak error. A precise video playback rate is indicated numerically above the slider.

The annotator specifies the size and location of an object of interest with a rubber-band bounding region in a single frame. The system estimates object size from the smaller dimension of the bounding region. The video is advanced one second, and the annotator is prompted to specify the location of the same object in a second frame. Based on the change in location between the two frames, the system calculates a simple linear estimate of object speed.

In an informal evaluation, annotators reported the interface was satisfactory for creating point-based paths to direction attention to objects in consumer video. However, estimates of speed and size the interface calculated could vary widely, so in practice, annotators may simply choose the

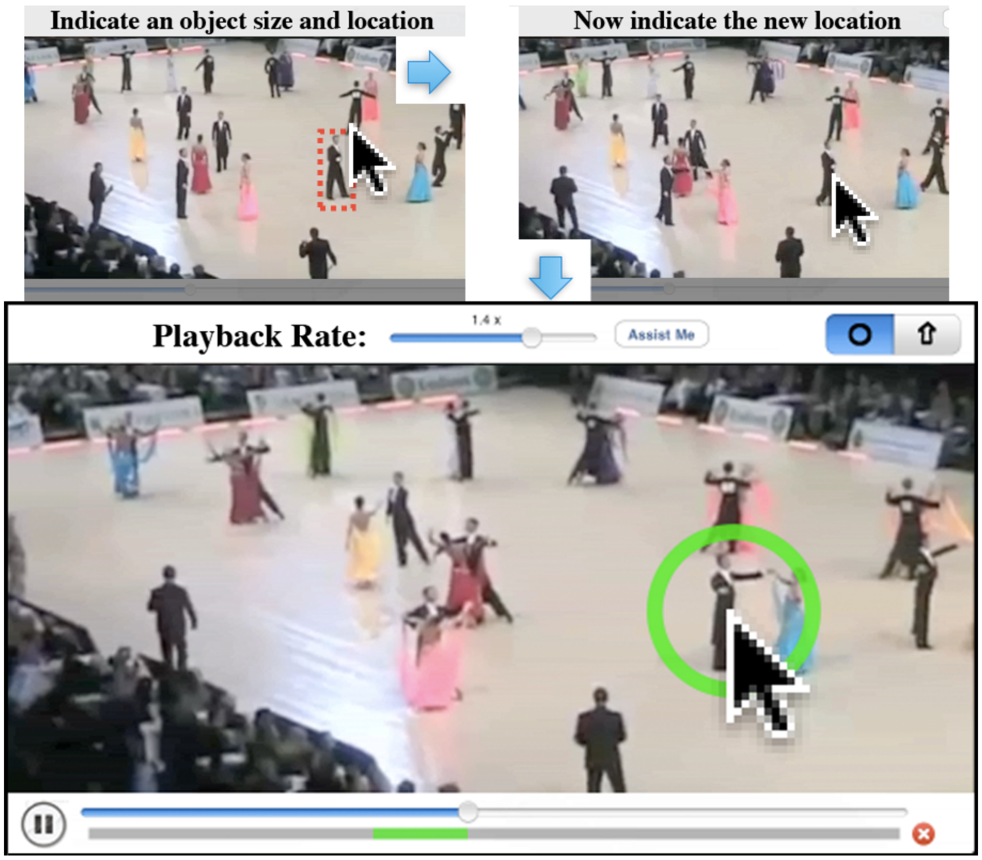

Indicate an object size and location
Now indicate the new location
Playback Rate:
1.4 x
Assist Me

Figure 6.1: Prototype pursuit method annotation interface to assist the annotator to choose a video playback rate. (1) The annotator identifies a video object size and location with a rubber band bounding region. (2) The interface advances the video one second and the annotator specifies the object's new location. With our model of human pursuit movements, the prototype sets the video playback rate so that 50% of pursuits have better than a required peak error.

playback rate directly once they become familiar with the playback rate settings.

6.3 Time-Shifting Paths

Annotator reaction time to video object movements is a significant factor in the accuracy of a pursuit method. Figure 6.2 illustrates how average peak error falls as the annotation paths created by with a pursuit method are shifted forward in time. An annotation tool designer can simply advance annotation paths by 0.1 seconds to improve the accuracy of a pursuit method.

The time shift to achieve a minimum error agrees with the range of perceptual processor times (100ms on average) reported by Card [21].

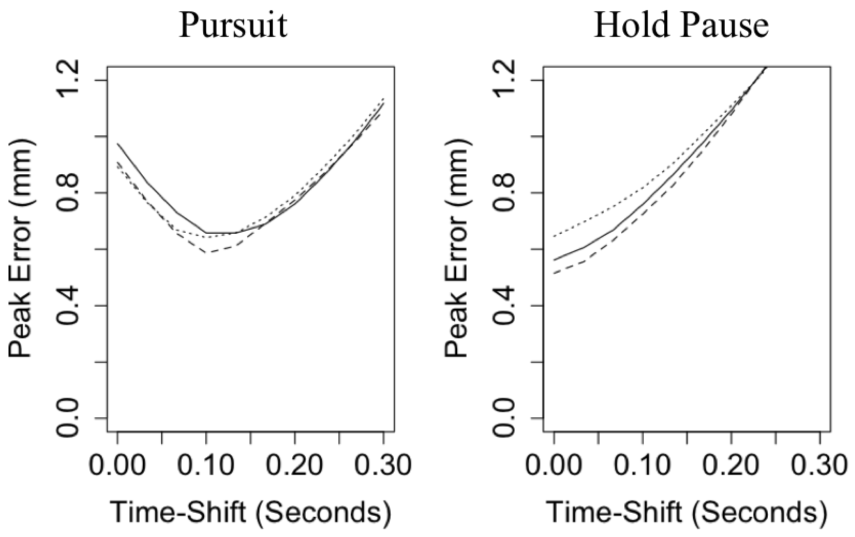


Figure 6.2: Peak error reduces when annotation paths created with a pursuit method are shifted forward in time. Hold-pause does not benefit from this adjustment. Only data from the 2.15 mm/s condition are presented. Dotted line: 1.25 mm objects, dashed line: 0.625 mm objects and solid line 0 mm objects.

6.4 Conclusion

A pursuit method combines video playback with interactions to define a point-based path. The predictable relationship between pursuit accuracy and video object speed and size allows designers to trade annotation accuracy for annotation time.

We showed how to adjust video playback rate to improve the accuracy of a pursuit method. If video is to be annotated with less than 0.4 mm peak error for at least 50% of pursuits, and the video contains objects of at least 0.625 mm in diameter, moving at 3.225 mm per second, video playback should be reduced to 41% of the original playback rate and a pursuit method will still produce acceptable results.

We demonstrated an interface to assist annotators choosing a playback rate. While the interface may reduce the trial and error of playback rate adjustment for novices, expert annotators will likely make quicker direct choices for video playback rate based on their experience of a pursuit method.

Additionally, we illustrated how peak error can be further reduced in video annotation systems by time-shifting annotation paths to reduce the effect of reaction time during a video object pursuit.

Annotation systems are likely to provide a number of methods in a toolset for annotators, and a pursuit method should not be overlooked as a viable option for creating point-based paths. The optimisations in this chapter show how designers can best incorporate pursuit methods into an annotation toolset.

Chapter 7

Conclusion

Video annotation can be a slow process, but by focusing on point-based paths, we aimed to develop faster annotation methods - particularly for consumer video. We noted that a completely manual annotation method - once validated - could be further combined with automation to improve video annotation.

A pursuit interaction has been explored in other contexts, but not yet researched as a method for video annotation. Researchers has explored other techniques to direct attention to stationary targets, and a pursuit method could be considered as expanding that research to consider moving targets.

The feedback to our prototypes suggested that with the right optimisations a pursuit method should compare well with alternative methods that interrupted video playback during annotation. However, gathering evidence to demonstrate the validity of video interactions is difficult since an annotator can encounter a wide range of video conditions.

We tackled this issue by gathered evidence from two different studies. Our first study was designed to isolate two important factors affecting the accuracy of pursuit methods - video object speed and video object size - by tightly controlling video object movements. But to provide evidence that a pursuit method would be suitable for typical consumer video, we designed our second study with more realistic video conditions.

We found that a pursuit method is viable for typical speeds and sizes of video objects. In some situations, a pursuit method even outperforms an alternative keyframe technique - for 1.25 mm video objects moving at about 1 mm/s, a pursuit method produces paths as accurate and satisfactory as Hold-Pause, and does so 12% faster.

Over a limited range of speeds and sizes, in both our studies we measured a predictable relationship between peak pursuit error and video object speed and size. This relationship allows designers to trade annotation accuracy for reduced annotation time, since path accuracy is related to video playback rate.

With our model, we showed how to adjust video playback rate to improve the accuracy of a pursuit method. If video is to be annotated with less than

0.4 mm peak error for at least 50% of pursuits, and the video contains objects of at least 0.625 mm in diameter, moving at 3.225 mm per second, video playback should be reduced to 41% of the original playback rate so a pursuit method will produce acceptable results.

Furthermore, we can time-shift the point based paths to further optimise the accuracy of the pursuit method. This time-shift optimisation, together with video playback rate adjustment, have been implemented in toolset that video annotators have found useful.

7.1 Future Research on Pursuit Methods

Our controlled experiments found evidence that a pursuit method is a viable technique for creating point-based paths. However, a number of shortcomings of our studies should be addressed in future work.

7.1.1 Extended Model Study

From our model experiment, the most prominent shortcomings are (1) large unexplained sources of variability and (2) the confounding effect of video object acceleration.

To address the unexplained variability, future studies should control the initial conditions of object movements more closely. Additionally, by varying the speed of targets before and after and acceleration impulse, future research could isolate the effect of video object acceleration. Tighter control of the experimental conditions would allow researchers to model the form of pursuit impulses, in a similar fashion to Plamondon [70].

Furthermore, to address mobile device interactions, future experiments should extend the range of experimental conditions to larger video object speeds and sizes suited to touch screen interactions.

7.1.2 Additional Video Conditions

In our second study, even though we assessed the performance of pursuit methods in realistic video conditions, future work should address a wider range of video conditions.

Performance measures for the VideoAnnEx tool were based on annotations of video "benchmarks" [56]. The benchmarks were of content typically found in news broadcasts, and did not describe the video in terms of the video properties, such as the number of objects in the video. However, future studies that report the performance of pursuit methods on the benchmark video sets would be more comparable between studies.

The differences between model coefficient in our two studies suggests that new factors from additional video conditions should be addressed in future models of pursuit method performance. Additionally, factors affecting pursuit method performance in new mobile annotation contexts - such distractions and vibrations - should be considered.

7.1.3 Semi-Automated Pursuit Methods Study

Although the pursuit method was less accurate in most conditions in the second study, future research should examine how automation can improve the accuracy.

A semi-automated pursuit method could select from spatio-temporal entity suggestions, or constrain automated segmentation algorithms with pixel values around the point-based path. Future research could use our model as a benchmark to assess the accuracy of semi-automated pursuits.

Additionally, automation could help configure annotations tools or choose the correct tool. Further research could examine how to detect when a pursuit method would be suitable, perhaps with optical flow measurements of object speeds.

7.1.4 Refinements to Pursuit Method Paths

A pursuit method provides a "top-down" approach to annotation, so that paths can be quickly created for refinement later (Appendix H). Even though annotators must revisit video frames to annotate multiple objects, once a path is created, it serves as a landmark to help annotators navigate within the video to add additional paths.

Also, corrections to some common mistakes, such as paths that "run long", can be performed quickly with targeted interactions. Other correction techniques could include direct manipulation to correct splines as described by Baudel [12] and could reduce the number of frames visited during annotation and allow correction in-time with playback.

Research on these corrections by pursuit methods would particularly relevant to distributed annotation contexts, where a number of annotators collaborate and refine the annotations of others.

Bibliography

[1] Facebook, 2011. `http://en.wikipedia.org/wiki/Facebook`. 2.1.3, 2.1.3, B.2

[2] Telestrator, 2011. `http://en.wikipedia.org/wiki/Telestrator`. A

[3] Youtube, 2011. `http://en.wikipedia.org/wiki/YouTube`. 2.1.3, 3.3, A, D.1, H.2

[4] Youtube, 2012. `http://www.youtube.com/t/press_statistics/`. 1

[5] G. D. Abowd, M. Gauger, and A. Lachenmann. The family video archive: An annotation and browsing environment for home movies. *Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, 2003. 2.1.2, A, C.1, H.2

[6] Johnny Accot and Shumin Zhai. Scale effects in steering law tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 1–8, New York, NY, USA, 2001. ACM. 4.2, 5.2.5

[7] P. Ackermann. Direct manipulation of temporal structures in a multimedia application framework. In *Proceedings of the second ACM international conference on Multimedia*, 1994. C.1

[8] R. Adams and L. Bischof. Seeded region growing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1994. 1, 2.3, 2.2.2, 3.4, B.1

[9] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *In ACM Trans. on Graphics (Proc. SIGGRAPH)*, 2004. 2.4, 2.2.2, D.1, H.2

[10] J.E. Allen, C.I. Guinn, and E. Horvtz. Mixed-initiative interaction. *Intelligent Systems and their Applications, IEEE*, 1999. 6.2.2

[11] Richard J. Anderson, Crystal Hoyer, Steven A. Wolfman, and Ruth Anderson. A study of digital ink in lecture presentation. In *Proceedings*

of the SIGCHI conference on Human factors in computing systems, CHI '04, 2004. 2.2

[12] Thomas Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, 1994. 2.1.3, 2.3, 2.2.2, 3.4, 7.1.4

[13] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2), 1987. B.3

[14] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *J. Image Video Process.*, 2008, January 2008. 2.1.3

[15] S. T. Bryson. Effects of lag and frame rate on various tracking tasks. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, 1993. F.1

[16] D. Bulterman, J. Jansen, P. Cesar, S. Mullender, E. Hyche, M. DeMeglio, J. Quint, H. Kawamura, D. Weck, X. Garcia Paneda, D. Melendi, M. Hanclik, D. Zucker, and T. Michel. Synchronized Multimedia Integration Language (SMIL 3.0), 2008. `http://www.w3.org/TR/SMIL/`. 2.1, 2.1.3

[17] N. Burtnyk and M. Wein. Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Commun. ACM*, 19(10), October 1976. 2.1.3

[18] V. Bush. As we may think. *The Atlantic Monthly.*, 1945. 2

[19] P. Bzier. Emploi des machines commande numrique. *Paris: Masson*, 1970. B.6

[20] D. Cabral, U. Carvalho, J. Silva, J. Valente, C. Fernandes, and N. Correia. Multimodal video annotation for contemporary dance creation. *Proc. CHI Conf. Human Factors in Computing Systems*, 2011. A, D.2

[21] S. K. Card, T. P. Moran, and A. Newell. *The psychology of human-computer interaction.* Routledge, 1983. 6.3, F.2.1

[22] T. Chen, C.T. Swain, and B.G. Haskell. An approach to region coding for content-based scalable video. In *Image Processing, 1996. Proceedings., International Conference on*, 1996. 2.1.3, 2.1.3

[23] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. Video matting of complex scenes. *ACM Trans. Graph.*, 2002. D.1

[24] A. Cockburn and T. Dale. Ceva: a tool for collaborative video analysis. *Proc. ACM SIGGROUP oup Work*, 1997. 2.1.2, 2.2.1, A, C.2, H.1, H.2

[25] Antonio Criminisi, Toby Sharp, Carsten Rother, and Patrick P'erez. Geodesic image and video editing. *ACM Trans. Graph.*, 29, November 2010. B.3

[26] E. R. F. W. Crossman. The information-capacity of the human motor-system in pursuit tracking. *Quarterly Journal of Experimental Psychology*, 12(1):01–16, 1960. F.1

[27] Philip DeCamp and Deb Roy. A human-machine collaborative approach to tracking human movement in multi-camera video. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, 2009. 2.1.2, A, D.2, H.2

[28] Nicholas Diakopoulos and Irfan Essa. Videotater: an approach for pen-based digital video segmentation and tagging. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, 2006. A, C.1, H.2

[29] Nicholas Diakopoulos, Sergio Goldenberg, and Irfan Essa. Videolyzer: quality analysis of online informational video for bloggers and journalists. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, 2009. A, C.1, H.2

[30] A. Dix, J. Finlay, G. D. Abowd, and R. Beale. Human-computer interaction. 1998. H

[31] D. Doermann and D. Mihalcik. Tools and techniques for video performance evaluation. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 2000. A, D.3, H.2

[32] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. Video browsing by direct manipulation. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, 2008. 5

[33] C. Engelbart, D. A conceptual framework for the augmentation of man's intellect. *Vistasin Information Handling.*, 1968. 2

[34] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 2003. D.2

[35] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstraction and reuse of object-oriented design. 1993. 2

[36] Andreas Girgensohn, John Adcock, and Lynn Wilcox. Leveraging face recognition technology to find and organize photos. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, MIR '04, 2004. D.3

[37] Gene V. Glass and Kenneth D. Hopkins. *Statistical methods in education and psychology, Volume 1.* Allyn and Bacon, 1996. 5.3.1

[38] D. R. Goldman. A framework for video annotation, visualization, and interaction. *PhD Thesis*, 2007. 1, 2, 2.2, 2.2, 2.2.2, 3.4, A, D.2, H.2

[39] Rudinei Goularte, Renan G. Cattelan, José A. Camacho-Guerrero, Valter R. Inácio, Jr., and Maria da Graça C. Pimentel. Interactive multimedia annotations: enriching and extending content. In *Proceedings of the 2004 ACM symposium on Document engineering*, DocEng '04, 2004. A

[40] J. Grudin and D. Bargeron. Multimedia annotation: An unsuccessful tool becomes a successful framework. *Communication and Collaboration Support Systems*, 2005. H.2

[41] Abir Al Hajri, Sidney Fels, Gregor Miller, and Michael Ilich. Moving target selection in 2d graphical user interfaces. In *Proceedings of Interact Conference on Human-Computer Interaction.* Springer, September 2011. 4, 4.4

[42] B. L. Harrison and R. M. Baecker. Designing video annotation and analysis systems. *In Proc. Graphics Interface 92*, 1992. 2, 2.1.2, 2.1.3, 2.2.1, A, C.2, H.1, H.2

[43] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 1952. 4.2

[44] Errol R. Hoffmann. Capture of moving targets: a modification of fitts' law. *Ergonomics*, 34(2):211–220, 1991. 4.2

[45] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185 – 203, 1981. D.2

[46] P. Hoschka. An introduction to the synchronized multimedia integration language, 1998. 2.1

[47] Michael Victor Ilich. Moving target selection in interactive video. 2009. 2.1, 2.2.1

[48] Thorsten Karrer, Moritz Wittenhagen, and Jan Borchers. Draglocks: handling temporal ambiguities in direct manipulation video navigation. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 623–626, New York, NY, USA, 2012. ACM. 5

[49] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988. B.6

[50] Azam Khan, Justin Matejka, George Fitzmaurice, and Gordon Kurtenbach. Spotlight: directing users' attention on large displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, 2005. 1, 2.2.2, 2.4

[51] Michael Kipp. Spatiotemporal coding in anvil. In *Language Resources and Evaluation*, 2008. C.1, H.2

[52] David L. Kleinman, Krishna R. Pattipati, and Arye R. Ephrath. Quantifying an internal model of target motion in a manual tracking task. *Systems, Man and Cybernetics, IEEE Transactions on*, 10(10):624 – 636, oct. 1980. 3.1

[53] G. E. Krasner. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. 1988. G

[54] Gordon Kurtenbach and Bill Buxton. Gedit: a test bed for editing by contiguous gestures. 1991. B.7

[55] Francis C. Li, Anoop Gupta, Elizabeth Sanocki, Li-wei He, and Yong Rui. Browsing digital video. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000. C.1

[56] C.-Y. Lin, B. L. Tseng, and J. R. Smith. Video collaborative annotation forum: Establishing ground-truth labels on large multimedia datasets. *TRECVID 2003 Workshop Notebook Papers*, 2003. 2.1.3, 2.2.1, 7.1.2, A

[57] Lennart Ljung. *System Identification.* John Wiley and Sons, Inc., 2001. 4

[58] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, 1999. D.2

[59] Huitao Luo and Alexandros Eleftheriadis. Designing an interactive tool for video object segmentation and annotation. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, MULTIMEDIA '99, 1999. A, B.5

[60] W. E. MacKay. Eva: an experimental video annotator for symbolic analysis of video data. *SIGCHI Bulletin*, 21(2), 1989. 2.1.2, 2.1.3, 2.2.1, A, C.1, C.2, H.2

[61] I. S. MacKenzie and W. Buxton. Extending fitts' law to two-dimensional tasks. *Proc. CHI Conf. Human Factors in Computing Systems*, 1992. B.1, B.2

[62] J. M. Martnez. Mpeg-7 overview (version 10). *International Organisation for Standardisation*, 2004. 2.1, 2.1.3, 2.1.3, 2.1.3, A

[63] Gregor Miller, Sidney Fels, Abir Al Hajri, Michael Ilich, Zoltan Foley-Fisher, Manuel Fernandez, and Daesik Jang. Mediadiver: Viewing and annotating multi-view video. *Proc. CHI Conf. Human Factors in Computing Systems*, 2011. 2.1.3, 2.2, A, D.1, H.2

[64] S. Mizobuchi and M. Yasumura. Tapping vs. circling selections on pen-based devices: evidence for different performance-shaping factors. *Proc. CHI Conf. Human Factors in Computing Systems*, 2004. B.1, B.4

[65] E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. *In ACM Trans. on Graphics (Proc. SIGGRAPH)*, 1995. B.4

[66] Fernando Navas and Lawrence Stark. Sampling or intermittency in hand control system dynamics. *Biophysical Journal*, 8(2):252 – 302, 1968. 4.2

[67] Merrill Noble, Paul M. Fitts, and Clause E. Warren. The frequency response of skilled subjects in a pursuit tracking task. *Journal of Experimental Psychology*, 1955. F.1

[68] Sile O'Modhrain and Ian Oakley. Touch tv: Adding feeling to broadcast media. In *in proceedings of the European Conference on Interactive Television: from Viewers to Actors*, pages 41–47, 2003. A

[69] B. Parker, E. and H. Zinkham. Thesaurus for graphic materials. *Washington, D.C.: Library of Congress.*, 1995. 2.1

[70] Rejean Plamondon and Adel M. Alimi. Speed/accuracy trade-offs in target-directed movements. *Behavioral and Brain Sciences*, 20(02):279–303, 1997. 4.2, 7.1.1

[71] M. J. Potel and R. E. Sayre. Interacting with the galatea film analysis system. *Proceedings of the 3rd annual conference on Computer graphics and interactive techniques*, 1976. 2.1.3, A, D.1, H.2

[72] E.C. Poulton. On prediction in skilled movements. *Psychological Bulletin*, 1957. 3.1

[73] Nathan C. Rahn, Youn-kyung Lim, and Dennis P. Groth. Redesigning video analysis: an interactive ink annotation tool. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, 2008. A, H.1

[74] G. Ramos and R. Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. *Proceedings of ACM symposium on user interface software and technology (UIST)*, 2003. A, C.1

[75] H. Rehatschek, W. Bailer, H. Neuschmied, S. Ober, and H. Bischof. A tool supporting annotation and analysis of videos. *Reconigurations. Interdisciplinary Perspectives on Religion in a Post-Secular Society.*, 2001. 2.1.3, 2.2.1, A, D.3, H.2

[76] Cameron N. Riviere and Nitish V. Thakor. Effects of age and disability on tracking tasks with a computer mouse: Accuracy and linearity. *Rehabilitation Research and Development*, 1996. F.1

[77] D. A. Robinson, J. L. Gordon, and S. E. Gordon. A model of the smooth pursuit eye movement system. *Biological Cybernetics*, 55:43–57, 1986. 10.1007/BF00363977. 3.1, 4

[78] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *In ACM Trans. on Graphics (Proc. SIGGRAPH)*, 2004. B.5

[79] R. A. Schmidt, H. N. Zelaznik, B. Hawkins, J. S. Frank, and J. T. Quinn. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86(5):415 – 451, 1979. 4.4

[80] D. Setterwall. Computerised video analysis of football - technical and commercial possibilities for football coaching. *Masters Thesis*, 2003. 2.2.1

[81] J. Sivic and A. Zisserman. Video Google: Efficient visual search of videos. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *LNCS*, pages 127–144. Springer, 2006. D.3

[82] J. R. Smith and B. Lugeon. Visual annotation tool for multimedia content description. In *SPIE 4210*, 2000. 2.2.1, 2.2.1, A, C.1, D.3

[83] A. J. Spence, H. Tan, and A. Wilson. Accuracy of the turftrax racing data system for determination of equine speed and position. *Equine Veterinary Journal*, 2008. 2.2, 2.1, 2.2.1

[84] A. J. Spink, R. A. J. Tegelenbosch, M. O. S. Buma, and L. P. J. J. Noldus. The ethovision video tracking system - a tool for behavioral phenotyping of transgenic mice. *Physiological Behavior*, 2000. 2.2.1

[85] Robert F. Sproull. *Principles of interactive computer graphics (2nd ed.)*. McGraw-Hill, Inc., New York, NY, USA, 1979. B.5

[86] Z. Stone, T. Zickler, and T. Darrell. Autotagging facebook: Social network context improves photo annotation. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –8, june 2008. 1

[87] I. E. Sutherland. Sketch-pad: a man-machine graphical communication system. *In Proceedings of the SHARE design automation workshop*, 1964. B.5, B.6

[88] Zenonas Theodosiou, Anastasis Kounoudes, Nicolas Tsapatsoulis, and Marios Milis. Mulvat: A video annotation tool based on xml-dictionaries and shot clustering. 2009. A, H.2

[89] M. Topkara, S Pan, J Lai, S. P. Wood, and J. Boston. Tag me while you can: Making online recorded meetings shareable and searchable. *IBM Technical Report.*, 2010. 2.2, H.1, H.2

[90] Timo Volkmer, John R. Smith, and Apostol (Paul) Natsev. A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, 2005. A, C.1

[91] C. Vondrick and D. Ramanan. Video annotation and tracking with active learning. *Neural Information Processing Systems (NIPS)*, 2011. D.2, H.2

[92] Carl Vondrick, Deva Ramanan, and Donald Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, 2010. A, D.1, H.1, H.2

[93] K. Weber and A. Poon. Marquee: a tool for real-time video logging. *Proc. CHI Conf. Human Factors in Computing Systems*, 1994. 2.1.2, 2.2.1, A, C.2, H.1, H.2

[94] G. J. Wills. Selection: 524,288 ways to say this is interesting. *Proceedings of the 1996 IEEE Symposium on Information Visualization*, 1996. 2.2.2

[95] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes. Elan: a professional framework for multimodality research. *Proceedings of 7th Language Resources and Evaluation Conference*, 2006. A, C.1, H.2

[96] J. Yuen, B. Russell, C. Liu, and A. Torralba. Labelme video: building a video database with human annotations. *In Proc of ICCV.*, 2009. A, D.1, H.2

[97] Shumin Zhai and Paul Milgram. Human performance evaluation of isometric and elastic rate controllers in a 6 dof tracking task. In *Proc. SPIE Vol. 2057 Telemanipulator Technology and Space Telerobotics*, pages 130–141, 1993. F.1

Appendix A

Annotation Metadata

Metadata are any information associated with annotation semantic entities. They can both describe and expand upon a semantic entity and are often structured in a particular format, such as a formal grammar. MPEG-7 represents metadata with a Semantic Description Scheme that structures the syntax and relationships between metadata [62].

Both static or dynamic metadata are useful. Goldman et al. describes two dynamic alternatives: (1) "scribbles" - pen strokes placed on spatio-temporal entities to move with semantic entities; and (2) "graffiti" - pen strokes that deform and conform to the video object [38].

Metadata are not necessarily in the same medium as the semantic entity. Sound annotations, such as choreographer commentary, may be associated with dancers in a dance video. Vibrations have been added to some interactive television annotations [68].

Annotators sometimes use metadata to help disambiguate an annotation. For example, if the metadata are instructions to a writer on how to reformat text, then the semantic entity is likely a textual object for correction, rather than an abstract idea in a document.

Metadata can be associated with more than one semantic entity, and a semantic entity can have multiple metadata. Associations between a semantic entity and metadata can be made or broken, be time dependent, and are independent of the "original" semantic entity. An association between metadata and a semantic entity is visualised in a number of ways:

- Inferred. An association based on metadata properties or content, without any special treatment applied to spatio-temporal entity visualisations, such as:
    - Proximity. Metadata spatially and/or temporally close to recognisable spatio-temporal entities, such as notes on the margin of a page near underlined text, or the concept tags near video clips in the EfficientVideoAnnotation system [90].

    - Gradient. Metadata that appear to emanate from a particular spatio-temporal entity, such as a sound or a smell that diffuses from a spatio-temporal entity.

- Shared Properties. An association inferred from properties of *both* spatio-temporal entities and metadata, such as:

    - Colour. Metadata with the same colour as a spatio-temporal entity visualisation - such as colour-coded tags for players in sports video or spatio-temporal entity border colours in LabelMeVideo [96].
    - Symbolic. Metadata sharing textual or numerical symbols with a spatio-temporal entity to indicate association - such as numbered references in an academic paper.
    - Activated. Metadata and spatio-temporal entities that change simultaneously to indicate an association. Goldman describes how video users can point to sensitive spatio-temporal entities to reveal metadata added to semantic entities [38].

- Explicit. An explicit visualisation of an association between metadata and a spatio-temporal entity, such as with the straight line connecting player names with a player in MediaDiver [63].

Metadata can be created and edited in multitude of ways. The methods can be arranged by the particular types of metadata they support, such as:

- Textual metadata creating and editing methods: Videolyzer [29], VideoAnnEx [56, 82], Vatic [92], VANNA [42], SVAT [75], Family Video Archive [5], MuLVAT [88], ELAN [95], EVA [60], LabelMeVideo [96], CEVA [24], Videotater [28]

- Audio metadata recording/editing methods: M4Note [39], CreationTool [20]

- Graphical metadata creating/editing methods: LEAN [74], Marquee [93], InteractiveInk [73], CreationTool [20], Telestrator [2], M4Note [39], MediaDiver [63]

- Methods to create/edit interactive metadata elements: YouTube [3], Goldman et al. [38]

- Minimal metadata (some annotation tools specialise in region creation and correction, with minimal metadata): TrackMarks [27], Vatic [92], Viper [31], VOSystem [59], GALATEA [71]

Methods to associate metadata and regions can also be organised into distinct categories based on their visualisation or display:

- Metadata Manipulation: direct manipulation methods such as dragging to position metadata visualisations "close" to regions to indicate associations. Such associations are similar to margin annotations of books - comments are associated with a section of text by the proximity of the comments to the text. Some video annotation tools, such as LEAN, implement both spatial and temporal positioning methods, so that metadata appear and disappear over time [74].

- Association Manipulation: methods to directly manipulate an explicit visualisation of the association, such as graphical lines dragged between visualisations of region and metadata in MediaDiver [63].

- Shared Property: methods to match metadata and region properties to indicate associations. For example, "tag paint" dragging interactions found in Videotater [28] and Marquee [93] matches the colours of regions with metadata.

- Synchronised: methods to associate metadata with the time of region creation, such as the multimodal techniques to record audio in time with region creation provided by the M4Note annotation tool [39].

Appendix B

Spatial Area Definition Methods

B.1 Selection Methods

To create a region with selection methods, an annotator selects a number of sub regions by pointing with a mouse or finger.

Region selection methods are usually faster than lasso methods on region definition tasks involving stationary sub-regions of three to six millimeters - an exception being definition of “low complexity” regions of closely packed sub-regions [64].

The time to select a two dimensional region is modeled by MacKenzie [61]. For rectangular regions, the time to “acquire” each region is well described as a function of the smallest dimension of the region and the distance of the pointing device to the region center.

Data dependent selection methods include seed-based region growing techniques, such as Magic Wand [8]. To grow a region, an annotator selects a point on an object representation and an algorithm expands a complex region around the point. The algorithm can grow regions based on (1) the presence of edges detected on the representation, (2) the presence of similar pixels to the selection pixel (in colour or intensity), or (3) a hybrid approach that fills “pools” within the image. Parameterised regions can also be fitted to the grown regions.

B.2 Placement Methods

Annotators place point-based regions - those inferred by a video viewer - by pointing to the intended center of the new region with a mouse of a finger.

Parameterised regions, such as fixed dimension “cookie-cutter” regions found in Facebook, can also be positioned with placement methods [1].

The time to place a region with a particular accuracy is modeled by MacKenzie [61].

To move an entire region, an annotator can drag the area inside the region. To rotate or rescale a region, additional draggable "handles" are often provided.

However, alternative methods such as rubber-band methods (see below) are typically used to adjust the shape or size of a region created with placement methods.

B.3 Brush Methods

Brush methods can be thought of as applying paint to form a region from painted sub-regions [13]. Instead of using a pointing technique, an annotator creates a region by tracing a path to select sub-regions.

The sub-regions can range in size, from as small as individual pixels, to larger regions formed by automatic sub-region detectors. The sub-regions can be regular or irregularly shaped. Regions can be corrected with brush modes that remove sub-regions from a larger region.

Data dependent brush methods include matting methods, such as Geodesic Segmentation [25]. To create a region with a matting method, an annotator specifies some foreground pixels, and perhaps some background pixels by tracing a path over the pixels. An algorithm then expands a region around the foreground pixels, by repeatedly fitting a model of both the foreground and background pixels colours until the models are well separated and match the image. The annotator can correct the expanded region - to some degree - by using more brush strokes to refine the model, although the annotator may have to reset the model.

Brush methods are useful to form a region with many small sub-regions, rather than a few large sub-regions better grouped with selection methods.

B.4 Lasso Methods

Lasso methods create regions by encircling sub-regions. Annotators create a region with a lasso methods by forming a complex region boundary along a "trail" traced from a start point. Region creation is completed when the annotator indicates an end point for the trail - typically, the end point is joined to the start point so that a region is formed inside the trail [64].

The region created can be smooth or parameterised. Unlike rubber-band methods, annotators cannot reposition the region while using a lasso method to create the region.

To create a region with a data dependent lasso method, an annotator could use "Intelligent Scissors" [65]. The annotator roughly traces a boundary in an image from a start point. However, the actual region boundary created adheres to edges in the image. An algorithm calculates paths through pixels in the image, with measures of path "length" based on whether neighbouring pixels form edges. The annotator effectively reveals optimal paths back to the lasso start point. To help an annotator create a path, Intelligent Scissors also "cools" older portions of the path so that they remain in place while the annotator traces the region boundary. Additionally, "training" techniques can be incorporated to help prevent the trail from jumping to strong nearby edges.

To correct a region with lasso methods, additional regions can be joined to an existing region or sub-regions can be removed. An annotator can also drag the boundary of a region to include additional sub-regions.To edit a region with the Intelligent Scissors, an annotator can drag the region boundary to force a path to pass through or avoid particular pixels.

These methods are usually slower than selection methods when sub-regions are large (relative to pixel sizes). But the methods are preferred when region editing cannot be modeled as a selection task, such as when sub-regions are as small as pixels.

Lassos allow annotators to specify region boundaries very precisely, but at the cost of increased interaction time.

B.5 Rubber-Band Methods

Rubber-band methods create parameterised regions from lines, circles or polygons. To create a region with a rubber-band method, an annotator moves vertices of the parameterised regions [85, 87]. As the annotator moves region vertices along a path, a region is dynamically created that deforms like a rubber-band. Once the annotator chooses the final position for region vertices, the region is completed.

Region vertices often define the corners of a rectangular region. But if the region is circular, the vertices can define the center and radius of the region. Polygonal regions can be formed from repeatedly adding line segments to a region perimeter.

To correct a region with a rubber-band method, an annotator can reshape the region by dragging on vertices or edges of a region. Vertices can also be added to or removed from polygonal regions.

Data dependent rubber-band methods are sometimes found in matting tools such as Grab-Cut [78]. To create a region with Grab-Cut, an annotator uses a rubber-band method to specify a rectangular seed region around an object. A foreground model is fitted to pixels inside the region, with the assumption that pixels immediately inside the seed region are background pixels. The region formed is based on fitting a colour model to image pixels, and can be adjusted with brush methods as described above.

Researchers have developed data dependent rubber-band methods similar to data dependent lasso methods. With VOSystem video annotation system developed by Luo et al., annotators create region boundaries by constraining an edge tracing algorithm with lines formed with a rubber band method [59].

Rubber-band methods are perhaps the oldest methods to specify regions, first developed by Sutherland for the SketchPad system [87]. Sketchpad was presented as an alternative to command based computer interaction predominant in the 1960s. Instead, diagrams and problems could be "communicated" to a computer with a light pen, although the communication was limited to two-dimensional topology.

Rubber band methods allow an annotator to adjust region vertices and see changes to the region in realtime. This helps the annotator to both position and reshape the region.

B.6 Shape-Parameter Methods

Similar to rubber band methods, but annotators create regions with shapes by specifying parameters of the region - instead of the positions of vertices composing the region. Such methods include Bezier-path techniques to adjust the curvature of region boundaries [19].

"Snakes" is a data dependent parameter method to fit boundaries to edges in an image [49]. An annotator traces a path in the image to specify a sequence of parameterised curves, and an algorithm moves the curves towards edges in an image, forming a region inside the curves. An annotator can correct the region by explicitly setting points that a curve must pass though, but the curves forming a region are typically "simpler" than the edges in the image. Parameterised regions are more compact region representations, but may not follow complex edges in images, such as around hair.

Sketchpad provided constraints to be specified on region parameters - rectangles could be constrained to be horizontal, for example [87].

B.7 Grouping Methods

Regions can be created with methods that only operate on groups of regions. Regions can be added to other regions or sub-regions can be removed. The regions added or removed could be specified with a variety of other methods, such as selection or brush methods.

Such grouping methods were provided in the GEdit tool [54]. Annotators manipulated regions with gestures formed with a pointing device. The tool recognised distinct gesture paths to manipulate groups of regions with lasso methods. The researchers pointed out that (1) operations have to be learned by annotators since they are not immediately apparent and (2) forming gestures with a mouse was more difficult than with a stylus.

Grouping methods also include methods to deform region boundaries so as to include or exclude sub-regions. An annotator could drag the boundary of parameterised region - with a rubber band method for example - to intersect it with another region.

Appendix C

Temporal Period Definition Methods

C.1 Spatially-Represented Methods

Temporal regions are often represented spatially, so annotators often use spatial region definition methods to create and correct temporal periods.

An early software framework developed in the early 1990s by Ackermann, provided direct manipulation of spatially-represented temporal periods with rubber-band methods [7]. Ramos et al. developed the LEAN video annotation system that provided lasso methods to manipulate spatially-represented temporal periods with a pen interface [74].

Kipp describes a speech and language research tool, called ANVIL, that provided grouping methods to manipulate spatially-represented temporal periods [51]. An ANVIL annotator used selection methods to specify periods to merge into larger periods, while still preserving the details of the original periods. The tool grouped periods from multiple media streams - in a similar fashion to early video annotation tools, such as EVA [60].

Data dependent temporal methods use semantic entity detectors to create period suggestions. Speech recognisers in audio annotation tool ELAN create spatially-represented temporal periods around words and phonemes in audio media [95]. Video analysis techniques detect scene changes within video to create temporal periods in the Family Video Archive [5], VideoAnnEx [82], Videotater [28], Videolyzer [29] and the EfficientVideoAnnotator [90].

Li et al. point out that special considerations must be made for spatially-represented temporal periods that are either very short and very long [55]. The researchers developed a digital video viewing tool with adjustable video playback speed to navigate more effectively within very long video sequences.

C.2 Synchronised Segmentation Methods

Synchronised segmentation methods are used to create periods without stopping video playback. While the video plays, annotators press buttons or perform other interactions in time with video playback to specify a period.

To correct periods created with synchronised segmentation methods, the periods are typically arranged spatially in lists or on a timeline, and annotators correct the periods with spatially-represented methods.

In the late eighties, most video was not digitised, and video annotation research focussed on methods to annotate events on analogue video cassette tapes. MacKay developed the experimental video annotator (EVA) [60] to annotate "media streams" - video recorded from different angles, audio recordings, scripts and button-press logs. The tool provided synchronised segmentation methods to create periods. During media review, an ethnographer pressed buttons corresponding to different event types to create instantaneous temporal periods for editing later. She observed that realtime video annotation was an intense process, but annotation time was reduced by annotating events without pausing the video.

Based on a task analysis of video annotation and interviews with annotators, Harrison and Baecker developed a similar ethnographic tool: the Video ANNotation and Analysis (VANNA) system [42]. The tool provided synchronised segmentation methods to create periods - annotators pressed customizable buttons for different predefined metadata in time with events during video playback. Richer features were provided for editing periods and metadata "offline". VANNA provided direct-manipulation spatially-represented temporal period methods, and annotation interface elements were presented alongside elements to control video cassette playback. The researchers found that system users adopted two distinctive annotation styles: (1) realtime annotation, with frequent "loopbacks" to review annotations and (2) slow motion annotation, with few loopbacks.

At Xerox Parc, Weber et al. developed a pen based video annotation tool - Marquee - that recorded temporal periods on a vertical note taking "scroll" [93]. The tool provided a synchronised segmentation method: to create a period known as a "timezone", the annotator drew a horizontal line across the scroll, in time with a video playback event. The video playback time coincident with the horizontal line defined the period start - the period end was either the end of the video or the start of another period. Handwritten metadata added below the line were associated with the period, and were typically unstructured, although they could contain predefined "tags" for navigation among events. Although period start and end times could be corrected later, the work does not address how to create a period “out of order”, to be inserted between other period.

Cockburn et al. noted that although video analysis is slow, it is often collaborative [24]. The researchers built the CEVA video annotation system to support synchronised segmentation methods by co-located annotators. The interface only supported temporal periods, representing them spatially, and allowed for multiple overlapping periods to correspond to threads of activity in the video. Metadata were either fixed iconic values or simple textual labels. The system provided an interface controlled by two mice - to be shared by either remote or co-located annotators. During an evaluation with four co-located post-graduate computer science students, Cockburn found that the students annotated silently during video playback, and avoided stopping the video since it would disturb their partner. The students did discuss their annotations during pauses in video playback. However, the researchers didn’t compare the accuracy of annotations created during interrupted and continuous video playback.

Appendix D

Spatio-Temporal Volume Definition Methods

D.1 Keyframed Methods

An annotator pauses video at two successive keyframes. The annotator encloses a video object with a spatial region in both frames to create waypoints. An interpolation algorithm generates spatial regions for times intervening the waypoints.

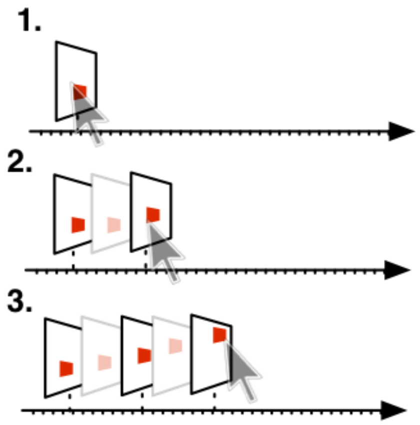


Figure D.1: Sequence of interactions to create a spatio-temporal volume with keyframed methods. The red square is a projection of the volume.

To correct a spatio-temporal volume, an annotator (1) corrects a waypoint region, and any interpolated projections are recalculated or (2) adds or removes keyframe regions to alter the interpolation. Corrections with keyframed methods change regions for times both before and after a keyframe - so annotators typically check the interpolations are accurate by reviewing the video around the keyframe.

YouTube spatio-temporal volumes, the basis of YouTube "speech bubbles", are created and corrected with rubber-band methods [3]. The keyframe regions are constrained to be rectangular and composed of horizontal or vertical line segments. The annotation tool simply extends the spatial region between two keyframes with a time-independent interpolation. Although YouTube spatio-temporal volumes are quick to create and correct, the volume has the same region in all frames it appears in, so the volume is stationary and doesn't follow moving objects.

Some annotation tools calculate volume extensions based on annotators corrections in previous frames [96]. Linear predictions of extensions don't follow objects that move in complex paths, however. To address this, the LabelMeVideo annotation tool, developed by Yuen et al., calculated extensions based on a three dimensional model of camera position for non-linear predictions that followed some objects more closely, although no measures of improved annotator performance were reported.

With keyframed methods, annotators typically adopt a "follow-and-check" approach during annotation (see E). They work on objects individually, creating waypoint regions in keyframes approximately every twenty frames (in thirty frames per second video). Annotators only return to correct interpolations when the object leaves the video. If they are satisfied with the spatio-temporal volume, they proceed to another object. Expert annotators using manual keyframed methods to annotate ten basketball players identities and sizes take approximately an hour for every twenty seconds of broadcast quality video.

Manual keyframed methods were provided in early spatio-temporal annotation tools, such as GALATEA [71], but are labour intensive. With interpolation, the methods may be suitable for some consumer applications, such as the MediaDiver [63], but to further reduce annotator labour, some tools provide data dependent keyframed methods.

Data dependent keyframed methods typically rely on algorithms that optimise how interpolated spatial regions fit video objects intervening the keyframes, subject to constraints such as that spatio-temporal volumes change smoothly. Although data dependent keyframed methods require fewer waypoints than data independent alternatives, annotators must still review video frames to check volumes correspond to moving objects.

Agarwala et al. reframed rotoscoping - an animation technique to trace the outline of objects represented in video - as an semi-automated process similar to a data dependent keyframed method. An animator specified spatio-temporal volumes by creating parameterised polygonal waypoint regions in two keyframes with a rubber-band method, and an algorithm interpolated projections between the waypoints to optimise measures of how well edges of the projections followed edges in the video. The researchers demonstrated the approach for three different video sequences, although no measures of rotoscoping performance are reported [9].

Chuang et al. developed a rotoscoping tool that also created spatio-temporal volumes with keyframed methods [23]. However, an animator using their tool specified two complex waypoint regions using a brush method. The regions contained pixel colours belonging to a "foreground". Spatial-temporal volumes were then interpolated between the waypoint regions by both (1) looking for similar pixels in frames between waypoints, and (2) accounting for the optical flow of pixel values between video frames. Although the approach was demonstrated on three sample videos, measures of rotoscoping performance were not reported.

Vondrick et al. developed Vatic, a crowdsourcing video annotation tool implementing keyframed methods [92]. They investigated how to best balance human and computer work when creating spatio-temporal volumes for players in a sports video. (In their case, waypoint regions were constrained to be rectangular.) Two different interpolation approaches were compared - (1) simple linear interpolation and a (2) error minimisation technique based on machine-learned weights for colour and edge features. The research revealed that, for easy sports video with few occlusions, error minimisation was accurate enough to reduce overall annotation costs. But for complex video, linear interpolation performed as well as error minimisation, in which case, it was more prudent to reduce automation costs, and spend the savings on human labour.

D.2 Track-Based Methods

Track-based methods are useful for situations where spatio-temporal volumes have already been created. An annotator creates a volume using a method such as a pursuit method or keyframed methods. Alternatively, automated methods can create volumes. The annotator uses track-based methods to correct spatio-temporal volumes by (1) splitting, (2) merging or (3) truncating the volumes.

Some data-dependent track-based methods use object-tracking algorithms to suggest volumes. To create a spatio-temporal volume with a data dependent track-based method, annotators first specify a purely spatial seed region corresponding to a video object. Tracking algorithms then compare pixels from inside the seed region with video pixels at times preceding and

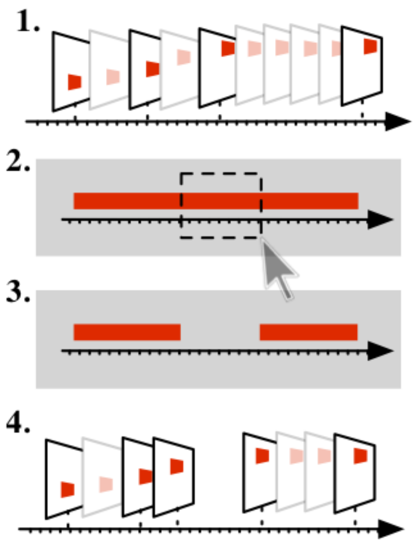


Figure D.2: Sequence of interactions to correct a spatio-temporal volume with track-based methods. The red square is a projection of the volume.

following the seed region and create spatio-temporal volumes based on optical flow [45] or feature tracking techniques [58]. Additionally, "top-down" constraints on the volumes, such as object motion models, can improve the accuracy of the volumes.

However, tracking algorithms struggle to follow objects that are obscured or that dramatically change shape and colour. Challenging lighting conditions and camera movements also introduce tracking errors. Cabral et al. reported that an annotation tool with data dependent track-based methods based on a continuously adaptive mean-shift tracking algorithm was "unsuited" to tracking handheld video of dancers [20]. (Mean-shift is a technique to adjust the size and location of a bounding box to maintain a consistent distribution of color features within the box.)

In another annotation context, DeCamp et al. reports some success with track-based methods optimised to deal with two common tracking issues: (1) when an object reappears after an absence from a video, or (2) when an object reappears after a momentarily occlusion [27]. The researchers developed the TrackMarks tool based on a realtime mean-shift tracker that generated "tracklets" (similar to spatio-temporal volumes). The tool provided progress of the realtime tracking process to annotators. The tool provided track-based methods to split and merge volumes. The researchers reported that skilled annotators take about 108 minutes to specify long volumes for a small set of objects in 60 minutes of video from stationary cameras monitoring child-caregiver interactions.

Goldman et al. developed data-dependent track-based methods to select spatio-temporal volumes detected by a particle tracking algorithm [38]. The research assumed that video is pre-processed offline to determine groups of particles - image features that correlate with object movements. To select a volume, an annotator uses a brush method to select particles while the video is paused, and a spatio-temporal volume is grown spatially and temporally to contain similar particles in neighbouring frames. The work mentions situations when the particle tracking "breaks down" - when the volumes no longer corresponded well to video objects - but doesn't investigate methods to address the situations.

Vondrick and Ramanan developed a tool, based on "active learning" techniques, that only displays volume suggestions that may need corrections [91]. Without any reference to a ground truth, their approach is to suggest frames to annotator where a change to the spatio-temporal volume will maximise change to the volume in other frames. Their approach doesn't suggest frames when there is certainty in the volumes, gracefully degrades when the tracker cannot localise objects at all, handles object deformation and occlusion, and doesn't suggest frames for stationary objects. On a benchmark surveillance video dataset, their dynamic programming algorithm outperformed keyframed methods, reducing annotator mouse clicks nearly ten-fold.

Fails and Olsen developed a volume detection tool based on brush methods [34]. Called Crayons, the tool allowed users to train a pixel classifier without requiring advanced programming or image processing skills. Once trained, the classifier suggested spatial regions in sequences of images that could be joined into a volume. The researchers chose a decision tree machine learning technique that provided rapid feedback of the classifier training state, even with a large number of features. The researchers also used "integral images" as fast and flexible features, although the features were not based on temporal measures. In evaluations, five users each took less than 10 minutes to train a classifier to detect volumes of an image set with 98% accuracy relative to a ground truth.

D.3 Sparse-Frame Methods

An annotator pauses video at a particular frame and encloses a video object with a spatial region method. The annotation tool searches for other frames containing similar video objects and presents suggested regions from a multitude of times before or after the initial frame. Suggested regions have been based on face detectors, text detectors, or distinctive image feature detectors. The annotator chooses a batch of suggestions to join into a larger spatio-temporal volume in a single operation. The annotator rejects false-positive suggestions that do not correspond to the correct semantic entity.

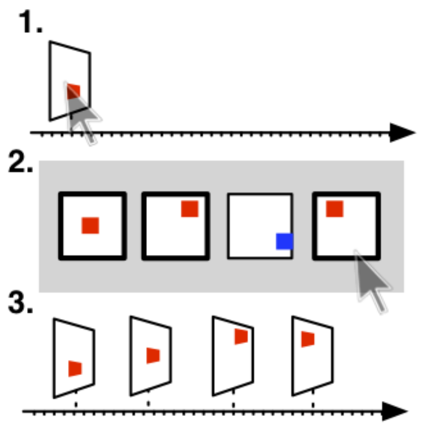


Figure D.3: Sequence of interactions to create a spatio-temporal volume with sparse-frame methods. The red square is a projection of the volume.

To correct the spatio-temporal volume formed suggested sub-regions, an annotator can remove erroneous sub-regions or add regions from additional searches. Region suggestions can be corrected with spatial region methods. Track-based methods could also be used to correct volumes created with sparse-frame methods. For example, a dancer may be wearing a shirt with two distinctive patterns on the front and back, so two separate searches may be merged to create a volume corresponding to the dancer.

Not all video frames need to be annotated with sparse-frame methods. With VideoAnnEx, videos were first split into "shots" by an automatic scene detection algorithm, and "representative frames" chosen from within the shots [82]. Representative frames were presented to the annotator in a scrollable horizontal list. An annotator used a rubber-band method to create a seed region for a "key object" in the frame. When the annotator viewed subsequent frames, the tool automatically extended the spatial region (at the same location), and annotators moved and resized the volume to better fit the key object as it appeared in subsequent frames. However, completely manual annotation using temporally-extended methods is slow - annotators took three hours on average to annotate key objects, scenes and events in sparse keyframes for 30 minutes of newscast video.

The methods work with sparse keyframes from video. Since suggested regions can be in distant frames, situations when a video object disappears and reappears later can be accommodated.

Sparse-frame methods don't work well with video of many similar semantic entities, such as video of sports teams, since the technique presents many erroneous suggestions to annotators.

Doermann and Mihalcik developed an annotation tool to detect regions containing faces or text [31]. Once detected, suggested regions corresponding to faces or text could be joined into a larger spatio-temporal volumes. The annotator could also adjust the shape of region suggestions to better fit video objects. Although the researchers discussed measures of annotation accuracy, they don't report measures of annotator performance that include annotation time.

Suggestions can be arranged in a time-saving order for annotators. The Rich Media Organiser tool developed by Girgensohn et al. grouped regions surrounding similar faces [36] in photos. A user specified a face as a "prototype", and the tool sorted suggested regions for other faces according to how similar the faces were to the prototype. The user then grouped multiple suggestions, instead of finding and grouping individual faces. Using a library 488 photos, the researchers showed how state-of-the-art face detection and recognition algorithms allowed a user to group suggestions containing 60% of the 63 appearance of a face with just four operations - instead of 38 operations (60% of the 63 faces) on individual unordered photos.

Suggestions for regions have been based on image feature classification algorithms. For analysis of theological video, Rehatschek et al. developed the Semantic Video Annotation Tool (SVAT) [75], based on Sivic and Zisserman's "Google Words" video search method [81]. Using SVAT, an annotator uses a rubber-band method to specify a seed region containing image edge features that form a "visual search word". The tool then suggests frames containing similar visual words by searching an index of keyframes for video shots. The annotator could join the suggestions to create a spatio-temporal region. The researchers don't report any measures of annotator performance.

Appendix E

Annotator Interviews

To better understand the tasks of a video annotator, we interviewed and observed two video annotators. Both annotators were doctoral students who had constructed a video annotation system who spent two months creating an annotation data set for a computer vision research. Neither annotator was compensated for their answers. We asked the following semi-structured questions to determine details of their annotation process:

- "What kind of video and video objects do you annotate?"
- "How long do you perform your annotation?"
- "Where do you perform your annotation?"
- "What tools do you currently use to annotate?"
- "What annotation tasks do you currently perform?"
- "What type of annotation regions are you interested in?"
- "What type of annotation metadata are you interested in?"
- "Can you describe a particularly troublesome annotation incident?"
- "What are the bottlenecks on your annotation process?"
- "What interface elements would you add to you annotation tool?"
- "What are the shortcomings with current annotation tools?"

Both annotators were male computer vision PhD students, aged 31 and 33. They used a custom-built annotation tool to annotate players in college basketball and ice-hockey. They annotated wide angle video from a single swiveling camera, taking about 1.5 to 2 hours to annotate 1000 video frames. Both annotators performed annotation in an office on a large computer monitor, although one described annotating on a plane to "kill time".

The annotators used keyframed methods to specify spatio-temporal regions for video objects. The annotators had tried track-based methods in an early version of the tool, but found track-based methods cumbersome - just as time consuming as keyframed methods, but without a fine degree of control.

Their system had a sophisticated technique to interpolate the positions and shapes of projections between keyframe waypoint regions. Before the annotators began their work, the tool ran an off-the-shelf object detector, the detection results were fed to a player recogniser tuned to shirt colours, and then a tracking algorithm tried to match players from frame to frame. When the annotators created regions, they defined a region around an object in a starting keyframe, then skipped to a second keyframe about 20 frames later to specify a region surrounding the same object, and their system used the offline tracking results to calculate interpolated projections between the keyframes.

Both annotators adopted a follow-and-check technique. Step 1: Add bounding boxes every 10 to 20 frames (sometimes 30 depending on the activity of the figures) and rely on interpolation to both fill in the other frames and resize bounding boxes. Step 2: Go back and visually check the interpolation results, correcting any errors, because there are no systems that can verify the work.

Both annotators used rectangular two-dimensional bounding box region, that could be resized both horizontally and vertically to closely fit to players. They used a jersey within the bounding box rule of thumb to make the boxes as compact as possible.

Both annotators were just interested in player IDs (full names) and team membership. They first used a play-by-play dataset from the video provider to determine who was on the court. They found that using a full name is easier than a player ID, but suggested that may just be a personal preference. They sometimes use temporary identities until the player is recognized. During a line change, its often not clear who is on or off the ice.

Both annotator complained that they found it time consuming to recognise players for annotation. One explained that a high level outline of events in the video would be useful to focus annotation effort on portions of the video. The other pointed out that recognising players is taxing work, unlike "brainless" adding bounding box work.

Autosaving in their tool could be improved. Although their tool has undo, one annotator wished it had a redo function. Their tool currently has to run computer vision detection, recognition and tracking algorithms offline. The tool is not web based.

Both annotators pointed out the problem of crowd-sourcing and cloud-sourcing annotation. It may be better to have five good annotators, rather than a thousand bad, since annotations must be manually verified. One annotator wondered what instructions he would give to less savvy annotators so they produced repeatable results. He thought that even if work could be distributed to other annotators, techniques to check the work of the other annotators would be needed.

An annotator is looking forward to a new feature they are developing that will map player movements onto an overview of the basketball court - he expects it will help him see errors in the annotations. He also expects that as computer vision techniques improve, he will have to specify waypoints in fewer frames.

Appendix F

Pursuit Model Study

Our study builds on other studies of pursuit task accuracy, but we go further by providing evidence of how target size affects pursuit accuracy and by examining pursuits of video object movements on typical video object paths - paths that that are unpredictable but not entirely random.

In both parts of the experiment, we examine movements during mouse pointer pursuits, since annotators will often use a mouse to annotate video. We report video object movements in terms of relatively small movement distances with the mouse - an experiment screen serves as a "magnifying glass" on the mouse movements and also presents video objects for pursuits.

There are two parts to the experiment. The first part tightly controls target movements in a one dimensional configuration to isolate the effects of target size and speed on pursuit accuracy and to provide a benchmark of accuracy. In the second part, we examine the effects of target speed and size on pursuits along realistic video object paths in a two dimensional configuration.

F.1 Related Work

Noble examined rotary one-dimensional pursuits of a target with a pointing device [67]. He describes a pursuit task as minimising an error $e(t) = o(t) - i(t)$ during a pursuit, where $i(t)$ is the position of a target and $o(t)$ is the position of a human controlled pointing device. As a measure of pursuit accuracy, he used a root mean square of $e(t)$ over the course of a pursuit. He measured increasing pursuit error with increasing target speed, but did not propose a model of the relationship. Additionally, subjects in his experiments pursued objects on predictable sinusoidal paths that do not necessarily require visually guided movement choices during pursuits.

Crossman modeled the processes of perception and movement during target pursuits as an information channel and tried to establish the capacity of the channel [26]. He controlled the predictability of target movements, by either showing or hiding future movements of a target. He found that in the absence of preview of target movements, the rate at which subjects transfer information from target movements to motor responses is limited by a process constrained to approximately 5 bits/second. However, subjects in Crossman's studies only pursued targets with a one dimensional steering wheel interface that simulated driving a car along a road.

Pursuit movements are used in contemporary studies of movements, such as Riviere's study of young, old and disabled subjects [76] or studies of device performance [15, 97]. However, the studies typically use pursuit accuracy to compare experimental conditions, and do not derive models of the pursuit process. The experiments also use customised metrics that are difficult to compare between studies. Furthermore, researchers often use predictable sinusoidal target movements and do not investigate the effect of pursuit target size.

F.2 Experiment Tasks

F.2.1 Part 1. Synthetic Paths

Targets with unpredictable but simple movements - infrequent direction changes, with few angles - provide an benchmark for subsequent studies of pursuit accuracy.

In the first part of our experiment, we constrain target movements to a single horizontal dimension. The target position $i(t)$ is synthesised so that the target follows a path that is (1) unpredictable, (2) of constant speed and (3) with controlled direction change frequency and (4) with controlled direction change angle of 180 degrees. Since the path is unpredictable, subjects must make visually guided corrections to their movements and don't learn the path during the experiment. Figure F.1 illustrates the target and cursor positions in the first part of the experiment.

The task for experiment subjects is to minimise one dimensional error $e_{1D}(t)$ from a mouse pointer to the edge of the target, although subjects can move the mouse pointer in two dimensions.

We controlled the target direction changes by selecting paths generated according to equation F.1 that contain three direction reversals. The direction changes are distributed uniformly over the trials by evaluating equation F.1 at intervals of 0.5 seconds. The maximum direction change frequency was chosen to be close to the limit of the human reaction times [21].

According to equation F.1, the target is more likely to return to the screen center, and when the target is at the center of the screen, the target is equally likely to travel right as to travel left. The likelihood of the target traveling towards the right of the screen at time t is given by the following

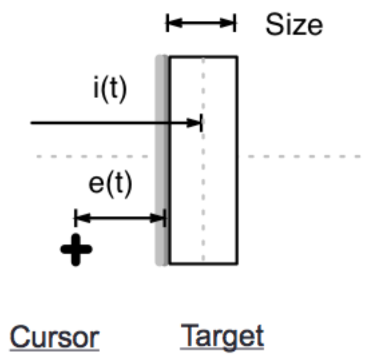


Figure F.1: The target position is $i_{1D}(t)$ and is constrained to lie on a horizontal line. The horizontal component of the distance between the human controlled mouse cursor and the target edge is $e_{1D}(t)$. When the cursor is inside the target, $e_{1D}(t)$ is 0.

equation, where $i_{1D}(t)$ is the position of the target relative to the center of the screen and W is a parameter refined in pilot studies to be a third of the screen width.

$$p = 0.5.(i_{1D}(t)/W) + 0.5 \tag{F.1}$$

F.2.2 Part 2. Derived Paths

Video objects typically move with complex direction changes (frequent, with various direction angle and magnitude change), but on two-dimensional paths that are not entirely random.

In the second part of the experiment, we adopt more realistic video object paths to examine human pursuits of targets on ecologically valid paths. However, to compare both parts of the experiment, we placed some restrictions on the paths. (1) They are approximately straight. (2) The paths contain three abrupt direction changes that occur infrequently (although the change is not an instantaneous 180 degree change as in Part 1). (3) The paths are transformed so they are unpredictable.

We simplify target shapes to circles. Figure F.2 illustrates the target and cursor positions in the second part of the experiment.

The task for experiment subjects is to minimise two dimensional error to the edge of the target.

We derived targets paths from hockey player movements in video illus-

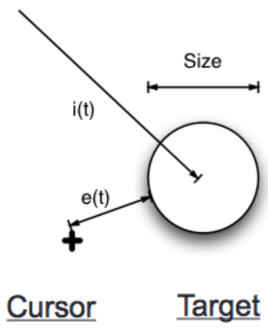


Figure F.2: The target position is $i_{2D}(t)$. The euclidean distance between the human controlled mouse cursor and the target edge is $e_{2D}(t)$. When the cursor is inside the target, $e_{2D}(t)$ is 0.

trated in Figure F.3. We chose a player path with movements that (1) remained inside the video frame, (2) changed direction three times (including starting "direction change"), and (3) otherwise remained smooth. We further smoothed the movements so that a target travelling along the path would move with constant speed. We centered a circular target on the path.

We randomly mirrored, rotated and translated the paths to make successive trials different. This transform prevents experiment subjects from learning the path - making the path unpredictable - but maintaining the abrupt direction changes and the direction change frequency.

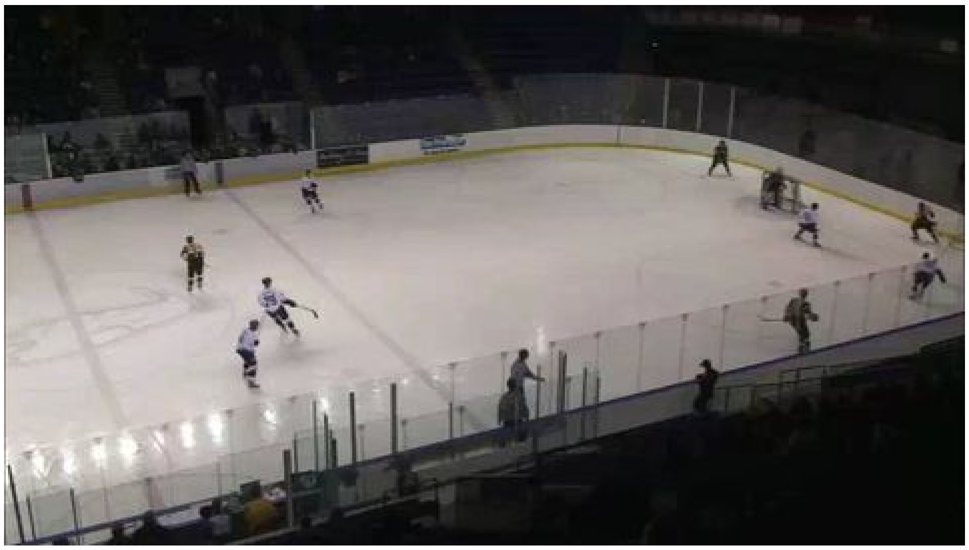


Figure F.3: Frame 3348 from a stationary camera recording of a college ice-hockey game.

	Part 1 - Synthetic Paths	Part 2 - Hockey Player Paths
Dependent Variable		
Error	Root mean square (RMS) of $e_{1D}(t)$ sampled every 1/30 second while the target is moving during experiment trials. We automatically terminated trials after four seconds. We calculate error during the last three seconds of each trial to allow subjects a period to adjust to the target speed and acquire the target. We measured $e_{1D}(t)$ to the edge of the target, $e_{1D}(t)$ is zero when the cursor is inside the target.	Root mean square (RMS) of $e_{2D}(t)$ sampled every 1/30 second while the target is moving during experiment trials. We automatically terminated trials after the target travelled 105 mm. We measured the euclidean distance $e_{2D}(t)$ to the edge of the target, $e_{2D}(t)$ is zero when the cursor is inside the target.
Independent Variables		
Speed	Three speeds (15 mm/s, 25 mm/s, 35 mm/s) chosen as typical for mouse based pursuits and refined in pilot studies.	Three speeds chosen to match the speeds of the target in the first part of the experiment. Targets follow the hockey player path with constant speed.
Size	Three sizes (0.5 mm, 3.5 mm, 6.5 mm) chosen to correspond to typical video object sizes (10, 70, 130 pixels) when displayed with a typical screen resolution, mouse tracking and control-to-display ratio. Also refined in pilot studies to spread the dependent variable mean.	Three sizes chosen to match the sizes of the target in the first part of the experiment.

Table F.1: Dependent and independent variables for both parts of the experiment

F.3 Subjects

We recruited ten subjects from a local university, one subject was female. Nine subjects reported using a computer mouse for more than an hour a day. When asked how long they spent playing computer games with a computer mouse, five subjects reported spending more than an hour a day, but another four reported spending less than an hour a month. All subjects were right-handed. No subjects reported colour deficient vision, but only eight subjects reported normal or corrected-to-normal eye-sight. Subjects were aged 22 years on average (standard deviation: 1.73 years).

An experiment session took less than sixty minutes. Subjects could stop their participation at any time. Subjects were compensated CAD$10 for participation whether they completed the experiment tasks or not. We presented subjects with printed instructions and clarified details if needed. Subjects completed a questionnaire detailed below.

In both parts of the experiment, subjects pursued the green target by keeping the purple tracking square inside the target as best they could. We informed subjects that the trials were challenging, that they should not be discouraged when it is hard to keep the square inside the target, and that they should try their best to keep the tracking square inside the target.

During an initial training period, subjects completed twenty-seven training trails with three repetitions of each condition of size and speed in random order. Subjects then completed ten analysis trials for each condition of size and speed in random order, yielding ninety analysis trials. The randomised order of each condition mitigated learning effects. Follow on studies to investigate learning effects or fatigue could administer trial blocks of nine trials each, so that each condition of speed and size appears in each block. Halfway through the analysis trails, subjects were encouraged to take a brief break, to mitigate fatigue and boredom.

Both parts of the experiment have a balanced designs, with ten repeated measures of each condition of speed and size within subject.

F.3.1 Synthesized Paths, Part 1

Subjects started each trial by holding a mouse pointer stationary inside a white start bar in the center of the screen. The start bar disappeared after the subject held the cursor stationary for one second, and a green moving target and a purple tracking square appeared. The start bar is illustrated in Figure F.4.

The purple tracking square was 10 pixels wide and centered on the cur-

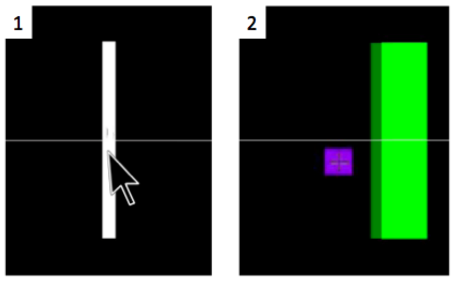


Figure F.4: 1. The trial starts when the subject holds the mouse cursor stationary inside the white start region. 2. The green target appears with a controlled speed and unpredictable movement path. The cursor is also replaced with a tracking cursor.

sor position to improve the visibility of the cursor and indicate the active tracking state. The moving target was a vertical green bar, 300 pixels tall, with controlled width and speed. The bar movements were confined to a horizontal axis. A horizontal line ran across the screen to help subjects align their mouse movements. The tracking square and moving target are illustrated in Figure F.4.

F.3.2 Derived Paths, Part 2

To mitigate learning effects, half subjects completed Part 2 before Part 1.

Subjects started each trial by holding a mouse cursor stationary inside a white start box in the center of the screen. The start box disappeared after the subject held the cursor stationary for one second. The start box is illustrated in Figure F.5.

The subject then moved the mouse cursor to a stationary green target and depressed the mouse button when the cursor was over the target. When the subject released the mouse button, a purple tracking square appeared at the cursor position and the green target started moving.

We measured error while the target was in motion. Subjects pursued the moving target for either 3, 4.2, or 7 seconds, depending on the speed of target, so that each trial required subjects to follow the target over the same distance of 105 mm.

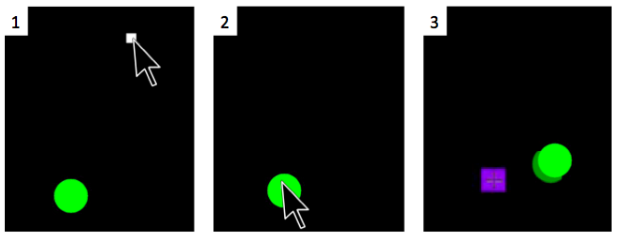


Figure F.5: 1. The trial starts when the subject holds the mouse cursor stationary inside the white start region. 2. The subject moves the mouse cursor to the green target and presses the mouse button. 3. When the subject releases the mouse button, the green target moves with a controlled speed and unpredictable movement path. The cursor is also replaced with a tracking cursor.

F.4 Apparatus

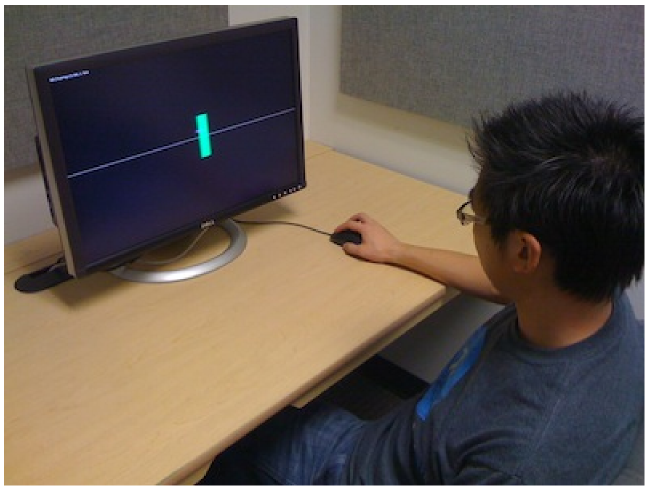


Figure F.6: The experimental setup with a subject sitting at a typical distance from the experiment screen.

We wrote our experiment software using the C++ annotation library described in Appendix G, and OpenGL and the OpenGL Utility Toolkit (GLUT). The experiment ran on Mac OS 10.6.8 on a Mac Pro with 8 GB of RAM and two 3 GHz dual-core Intel Xeon processors. The computer contained an ATI Radeon X1900 XT graphics card with 512 MB of video RAM and was connected to a Dell 2405FPW screen (518 mm x 324 mm), refreshed at 60 Hz, with a resolution of 1920 x 1200 pixels (0.270 mm/pixel pitch).

Subjects used a Microsoft Comfort 3000 mouse, with enough desk space to avoid "clutching". Mouse tracking was same for each subject - 1000 pixels for 50 mm of mouse movement. With the screen resolution described above, the control-to-display ratio is 0.185. Mouse resolution was 1000 increments per inch (DPI) or 39 increments per mm. Mouse acceleration was disabled for the experiment. A subject using the mouse and sitting at a typical distance from the monitor is illustrated in Figure F.6.

We verified records of mouse and target position records from the apparatus with movement test patterns.

F.5 Analysis of Variance

Hypothesis 1: As speed increases, peak error also increases. Hypothesis 2: As size increases, peak error decreases.

F.5.1 Synthesized Paths

There were approximately six repeated measures per condition and the following analyses were conducted with some missing measures. Additionally, we caution that subjects were not equally represented in the synthesized paths data set due to our experiment configuration.

We conducted Shapiro-Wilk's tests and noted that distributions were not normal ($p < 0.05$). However, histograms of the conditions indicated rougly normal distributions, and we assume that the results of the ANOVA tests that follow are robust to this deviation. We conducted Mauchly's sphericity tests and accepted that variances in error measures among conditions were equal ($p < 0.05$).

With a general linear model (GLM) repeated measures analysis, we determined there was a statistically significant difference among subject error measures for speed ($df = 2$, $F = 36.661$, $p < 0.05$) and size ($df = 2$, $F = 35.674$, $p < 0.05$). We rejected null hypotheses for part 1. Estimated power for effects of size and speed was 1, but fell to 0.292 for the interaction effect of speed and size.

Bonferroni post-hoc comparisons revealed significant differences among all conditions ($p < 0.05$). Figure 4.3 illustrates the differences between conditions. The trends illustrated lead us to accept both Hypothesis 1 and 2 for part 1.

F.5.2 Derived Paths

We conducted Shapiro-Wilk's tests and noted that distributions were not normal ($p < 0.05$). However, histograms of the conditions indicated slightly truncated normal distributions, but otherwise they appeared normal. We assume that the results of the ANOVA tests that follow are robust to this deviation. We conducted Mauchly's sphericity tests and accepted that variances in error measures among conditions were equal ($p < 0.05$).

With a second GLM analysis, we determined there was a statistically significant difference between conditions of speed ($df = 2$, $F = 107$, $p < 0.05$) and size ($df = 2$, $F = 82.1$, $p < 0.05$). We rejected null hypotheses for part 2. Estimated power for effects of size and speed was 1, but fell to 0.264 for the interaction effect of speed and size.

Bonferroni post-hoc comparisons revealed significant differences among all conditions ($p < 0.05$). Figure 4.3 illustrates the differences between conditions. The trends illustrated lead us to accept both Hypothesis 1 and 2 for part 2.

F.6 Questionnaire Analysis

F.6.1 Questions

Each subject completed a questionnaire after the experiment. For both parts of the experiment, the subjects answered the following questions:

- What strategy did you use to pursue the target? (Was there a particular technique you used to follow the target?)
- Did any factors other than target speed or size affect your performance during the experiment?
- How predictable was the path taken by the target? (100 = completely predictable, 0 = not predictable at all)
- What factors affected your choice of predictability? (Do not include speed or size of target)

F.6.2 Responses

Coded questionnaire results are reported in Table F.2, with examples of each code.

When asked what strategy they used to pursue the target, all the subjects described either predicting or reacting to target movements for both the first and second part of the experiment.

Subjects mentioned three different factors that affected their performance during both parts of the experiment - fatigue, direction changes and apparatus issues. One subject mentioned predictability as additional factor in the second part of the experiment.

Five subjects reported that the target paths during the second part of the experiment were more predictable than in the first part.

Subjects pointed out that the target is likely to return to the screen center when it approaches the edge of the screen in both parts of the experiment (3 in part 1, 4 in part 2). When asked what factors determined the predictability of the target path in the second part of the experiment, three subjects reasoned smoothness and four subjects reported familiarity. Three subjects reported the frequency of direction changes as a significant factor of path predictability during the first part of the experiment. One subject thought that two dimensional movements of the target in the second part of the experiment made the target path less predictable, while three subjects thought that the single dimension of movement in the first part was a factor.

	Code	Example
What strategy did you use to pursue the target?		
Part 1	Predicting (4)	“Anticipating the direction” “I pursued the target by seeing where it is going and trying to get my cursor to that place at the same speed”
	Reacting (5)	“I just tried to react to the movement of the target.” “Attempt to catch up with target rapidly, then adjust once within bounds.”
	None (1)	“None”
Part 2	Predicting (4)	“Tried to anticipate the changes in directions of the green circle (as it neared the edges of the screen).”
	Reacting (5)	“just followed said target as quickly as I could.” “For the smaller targets, it was more of a chase.”
	None (1)	“No”
Did any factors other than target speed or size affect your performance during the experiment?		
Part 1	Fatigue (3)	“Exhaustion and seeing the same thing over and over again”
	Direction Changes (2)	“The direction that the target moved affected my performance. It is easier to follow the target when it is not switching direction.”
	Apparatus Issues (3)	“Sensitivity of the mouse is not very good.” “Mouse sensitivity”
Part 2	Fatigue (2)	“My arm was aching”
	Direction Changes (1)	“The angle of the direction changes and the time between changes of direction affected my accuracy”
	Predictability (1)	“the boundary which a circle could go; the circles didn’t travel all the way in the screen.”
	Apparatus Issues (2)	“The mouse sensitivity.” “It’s harder to see the smaller circles when you’re overlapping it with the bigger square”

Table F.2: Codes derived from the questionnaire results with examples for each code.

What factors affected your choice of predictability?

Part 1	Edge of screen (3)	“When it is on the edge of the screen, I know that it has to go in the other direction.” “The rectangle seldom moved to the ends of the spectrum.”
	Change frequency (3)	“How much time it went in one direction before switching directions.” “The number of turns the pattern made.”
	Single dimension (3)	“The rectangle only moved on a set axis.”
Part 2	Edge of screen (4)	“It changes direction near the same area.” “When the green circle was close to the edges of the screen, it’d have to change directions.”
	Familiar paths (4)	“The number of turns it made, the distance which it travelled.” “Knowing that the bar would move to the polar opposite side after reaching a certain point.”
	Smooth paths (3)	“The target moves in similar looping path.” “It seemed a more natural form of movement.”
	Two dimensions (1)	“Multiple dimensions made it difficult to predict.”

Table F.3: Codes derived from the questionnaire results with examples for each code.

Appendix G

Video Annotation Software Framework

We developed a portable video annotation framework in C++. The annotation framework follows a Model-View-Controller design pattern [53] to separate the concerns of data modeling from interactions.

Annotation model elements are illustrated in the class diagram in Figure G.1. The framework retrieves and stores annotation model values in XML files. The key elements of the annotation model are:

1. Projection: A structure to represent a "slice" of a spatio-temporal region for a particular time.

2. AbstractRegion: An object to represent a spatio-temporal region. Concrete subclasses implement purely virtual function "ProjectionForTime" to return a Projection structure.

3. Shape: A polygonal primitive to represent a spatial-only region. Shapes are used to represent slices of regions in Projection structures.

4. InterpolatedRegion: The InterpolatedRegion is a concrete subclass of AbstractRegion. An InterpolatedRegion arranges Shape objects in a self-balancing binary search tree to efficiently insert and interpolate Projections for particular times. An InterpolatedRegion can also copy periods from other InterpolatedRegions.

5. ShapeAndTime: A structure to represent keyframes in an InterpolatedRegion. The structure represents four types of keyframe - start, continuous, disjoint and end - and can transform keyframes between the different types.

6. BindingModel: An object to manage dictionaries of regions and marks, keyed on RegionID and MarkID. The dictionary of regions represents a number of overlapping spatio-temporal volumes. The model also maintains a dictionary of bindings between regions and marks.

7. Binding: An object to represent a time-dependent association between a region and a mark.

8. Mark: An object to represent metadata to be associated with a number of regions. Marks are typically subclassed to add instance variables for particular information types.

Different user interface frameworks such as iOS UIKit can draw on the model. Here, view and controller elements for the Storm user interface framework (developed in the HCT lab) are illustrated in the class diagram in Figure G.2. The key elements of the annotation view hierarchy are:

1. FrameController: An object to coordinate annotation display and interactions for a particular video time. On video updates, the controller passes relevant Projection objects to an AnnotationLayer subclass for display. The controller also implements interaction logic to update the model on callbacks from the AnnotationLayer. For instance, the controller maintains a "working region" to accumulate region portions under construction before merging the newly constructed portions with existing region portions.

2. AnnotationLayer: An abstract object to display a set of Projection objects for a particular time. Concrete subclasses use a Flyweight pattern [35] to render Projection objects - Appearance objects are reused for Projections from the same region but different times. An annotation layer also captures mouse events but does not have a direct reference to the annotation model so relies on callbacks to the controller to create or modify regions. Mouse events are transformed into higher-level interactions such as shape creation or path creation with state machines.

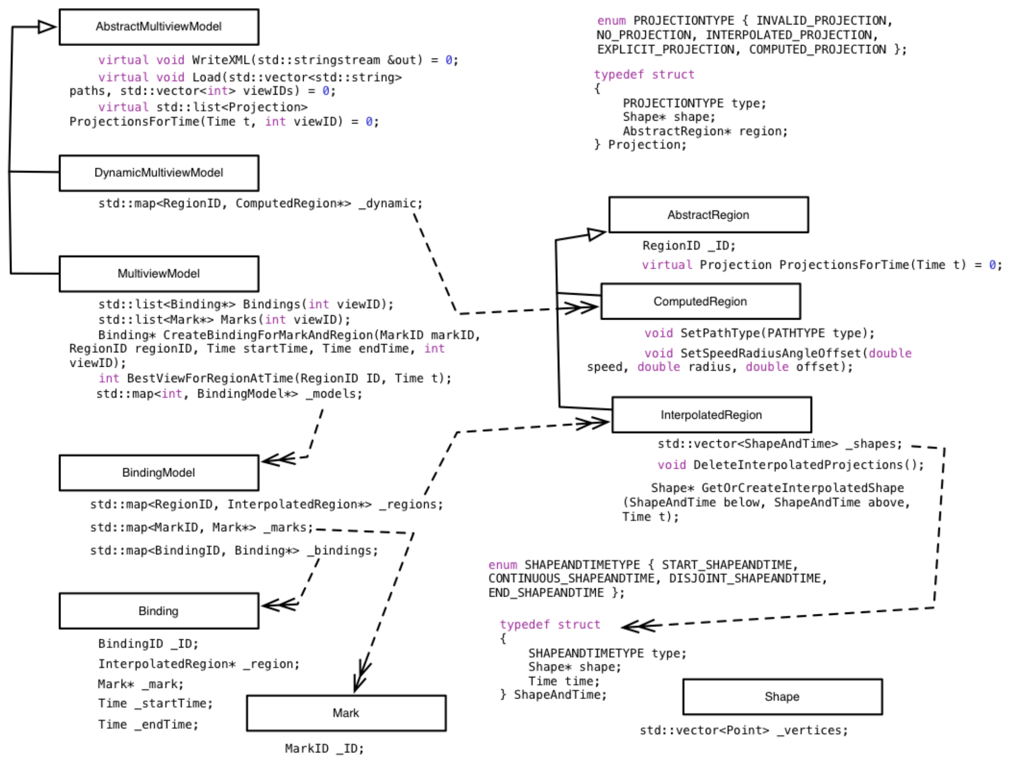
3. Appearances (SimpleProjectionUI or ProjectionUI): Flyweight objects to encapsulate interactions with Projections, such as dragging vertices of a polygonal Projection Shape. The objects also determine the style of Projection rendering, such as selected state colour and visualisation "tails".

To evaluate the performance of the framework, we conducted tests of keyframe insertion times and interpolation times on a MacBook Pro with an Intel Core 2 Duo processor running at 2.16Ghz, with 2GB of RAM. We report the results in Table G.1. Insertion times depart from a $log(N)$ relationship - future work should optimise the binary tree rebalancing.

Operation	Keyframes Present	Average Time (us)	Standard Deviation (us)
Insertion	1e5	7	5
Insertion	1e6	20	17
Insertion	1e7	115	87
Interpolation	1e5	5	1
Interpolation	1e6	9	1
Interpolation	1e7	16	2

Table G.1: Average insertion and interpolation times for the annotation framework. The times reported are based on ten insertions and interpolations for 10,000, 100,000 and 1,000,000 preexisting keyframes each containing a single point. We randomly spaced keyframes between 0 and 100 seconds to populate an InterpolatedRegion.

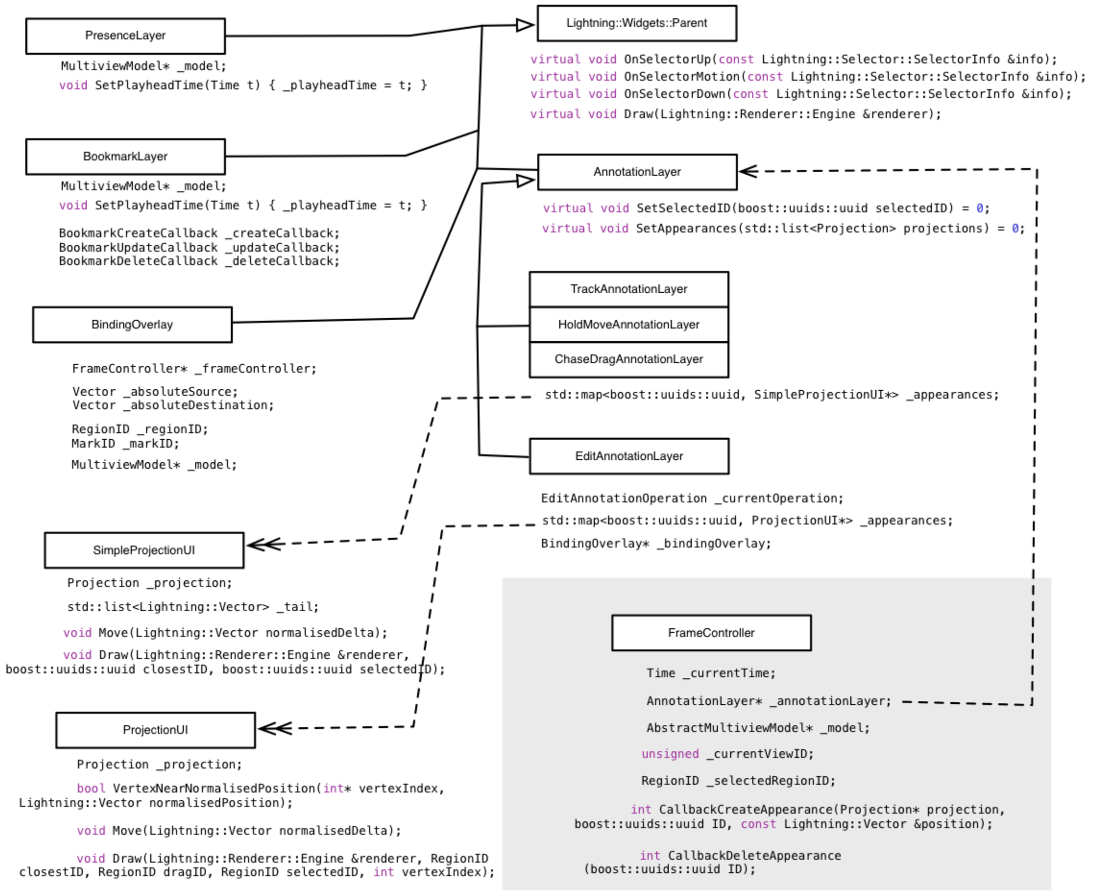
A known shortcoming of the annotation framework is the management of interpolated Projections by the InterpolateRegion class. Interpolated Projections must be explicitly deleted, typically by controllers - but the programmer must take care to avoid invalidating Projections that may still be referenced by other controllers and views. Future development could provide mechanisms to notify dependents when Projections are deleted.



```
AbstractMultiviewModel
    virtual void WriteXML(std::stringstream &out) = 0;
    virtual void Load(std::vector<std::string>
paths, std::vector<int> viewIDs) = 0;
    virtual std::list<Projection>
ProjectionsForTime(Time t, int viewID) = 0;

DynamicMultiviewModel
    std::map<RegionID, ComputedRegion*> _dynamic;

MultiviewModel
    std::list<Binding*> Bindings(int viewID);
    std::list<Mark*> Marks(int viewID);
    Binding* CreateBindingForMarkAndRegion(MarkID markID,
RegionID regionID, Time startTime, Time endTime, int
viewID);
    int BestViewForRegionAtTime(RegionID ID, Time t);
    std::map<int, BindingModel*> _models;

BindingModel
    std::map<RegionID, InterpolatedRegion*> _regions;
    std::map<MarkID, Mark*> _marks;
    std::map<BindingID, Binding*> _bindings;

Binding
    BindingID _ID;
    InterpolatedRegion* _region;
    Mark* _mark;
    Time _startTime;
    Time _endTime;

Mark
    MarkID _ID;
```

```
enum PROJECTIONTYPE { INVALID_PROJECTION,
NO_PROJECTION, INTERPOLATED_PROJECTION,
EXPLICIT_PROJECTION, COMPUTED_PROJECTION };

typedef struct
{
    PROJECTIONTYPE type;
    Shape* shape;
    AbstractRegion* region;
} Projection;

AbstractRegion
    RegionID _ID;
    virtual Projection ProjectionsForTime(Time t) = 0;

ComputedRegion
    void SetPathType(PATHTYPE type);
    void SetSpeedRadiusAngleOffset(double
speed, double radius, double offset);

InterpolatedRegion
    std::vector<ShapeAndTime> _shapes;
    void DeleteInterpolatedProjections();
    Shape* GetOrCreateInterpolatedShape
(ShapeAndTime below, ShapeAndTime above,
Time t);

enum SHAPEANDTIMETYPE { START_SHAPEANDTIME,
CONTINUOUS_SHAPEANDTIME, DISJOINT_SHAPEANDTIME,
END_SHAPEANDTIME };

typedef struct
{
    SHAPEANDTIMETYPE type;
    Shape* shape;
    Time time;
} ShapeAndTime;

Shape
    std::vector<Point> _vertices;
```

Figure G.1: Class diagram for the data model to support annotation regions for multiview video.



```
MultiviewModel* _model;
void SetPlayheadTime(Time t) { _playheadTime = t; }
```

```
MultiviewModel* _model;
void SetPlayheadTime(Time t) { _playheadTime = t; }

BookmarkCreateCallback _createCallback;
BookmarkUpdateCallback _updateCallback;
BookmarkDeleteCallback _deleteCallback;
```

```
FrameController* _frameController;

Vector _absoluteSource;
Vector _absoluteDestination;

RegionID _regionID;
MarkID _markID;

MultiviewModel* _model;
```

```
Projection _projection;

std::list<Lightning::Vector> _tail;

void Move(Lightning::Vector normalisedDelta);

void Draw(Lightning::Renderer::Engine &renderer,
boost::uuids::uuid closestID, boost::uuids::uuid selectedID);
```

```
Projection _projection;

bool VertexNearNormalisedPosition(int* vertexIndex,
Lightning::Vector normalisedPosition);

void Move(Lightning::Vector normalisedDelta);

void Draw(Lightning::Renderer::Engine &renderer, RegionID
closestID, RegionID dragID, RegionID selectedID, int vertexIndex);
```

```
virtual void OnSelectorUp(const Lightning::Selector::SelectorInfo &info);
virtual void OnSelectorMotion(const Lightning::Selector::SelectorInfo &info);
virtual void OnSelectorDown(const Lightning::Selector::SelectorInfo &info);
virtual void Draw(Lightning::Renderer::Engine &renderer);
```

```
virtual void SetSelectedID(boost::uuids::uuid selectedID) = 0;
virtual void SetAppearances(std::list<Projection> projections) = 0;
```

```
std::map<boost::uuids::uuid, SimpleProjectionUI*> _appearances;
```

```
EditAnnotationOperation _currentOperation;
std::map<boost::uuids::uuid, ProjectionUI*> _appearances;
BindingOverlay* _bindingOverlay;
```

```
Time _currentTime;

AnnotationLayer* _annotationLayer;

AbstractMultiviewModel* _model;

unsigned _currentViewID;

RegionID _selectedRegionID;

int CallbackCreateAppearance(Projection* projection,
boost::uuids::uuid ID, const Lightning::Vector &position);

int CallbackDeleteAppearance
(boost::uuids::uuid ID);
```

Figure G.2: Class diagram for the view and controller objects to view and manipulate annotation regions.

Appendix H

Video Annotation Design Guidelines

The following design guidelines for video annotation tools have been drawn from relevant literature, interviews with expert annotators (Appendix E), a study of novice annotators and feedback on an annotation tool prototype. The guidelines should "help the designer understand the trade-off for the design that would result in following or disregarding some" [30].

H.1 Guidelines for Annotators

- **Annotation should be an iterative process**

  Annotators should iterate on annotations [93]. Annotators can make specific fixes on each pass, concentrating on the same task throughout.

  During an initial "first pass", annotators should annotate significant events and provide landmarks for easy navigation later [42]. In this way, annotators focus their efforts on significant portions of the video.

  Iterations on annotations can be performed by different annotators. Vondrick et al. developed the crowdsourced video annotation tool - Vatic - for distributed annotation [92]. To improve the quality of annotation with distributed annotation systems, the researchers report that annotation should be split into two asynchronous phases: (1) creation, and (2) review.

- **Annotation should be a collaborative process**

  Cockburn et al. explored synchronous collaborative annotation with an interface for both co-located and remote annotators, although only co-located annotation was examined [24]. Their research suggests that annotators discuss their annotations during pauses in video playback.

  Rahn et al. point out that if techniques for resolving conflicting annotations could be developed, then richer annotations could be created from existing annotations - both synchronously and asynchronously [73].

Topkara at al. argue that annotation metadata should be shared and modified during collaborative annotations in the workplace [89]. Workers can correct the mistakes of others, or reuse metadata to improve the quality of workplace media annotations. However, expert annotators we interviewed pointed out the complexity of verifying the work of multiple annotators - particularly work from crowd-sourced annotators (Appendix E).

H.2 Guidelines for Tool Designers

- **Tools should visualise a range of annotation scales**

  MacKay proposed that tools visualise different levels of annotation "granularity" [60]. Annotations can range from comments on quick events such as sports goals, to detailed statistics on a player over the course of a game.

  Annotation overviews should (1) provide a map of where annotation can be focussed, (2) reveal errors in the annotations, and (3) support navigation within video. During our interviews, annotators recommended visualising player paths on a rink overview - this would allow them to see "breaks" in annotations that need additional work (Appendix E).

  To use a sports example, coaches annotating ice-hockey footage are not interested in the entire game - they prefer to see where players performance is critical, such as during power plays or face-offs, and avoid fights and when players are "killing time".

- **Tools should provide structured spatio-temporal entites and metadata**

  A number of researchers advise structuring both spatio-temporal entites and metadata, with sufficient flexibility to provide accurate annotations.

  Topkara at al. argue that consistency in annotation metadata makes annotation of workplace video more reliable [89]. But they also point out that the metadata must to be adjustable to best describe or elaborate on objects in the video.

During user studies, Weber et al. noticed that annotators needed metadata for (1) navigation and (2) to describe or elaborate on semantic entities [93]. Cockburn et al. advises annotation tools designers to allow searching within annotated video [24].

Annotations are often created for (1) further analysis or (2) to search for objects and events within video, so annotators create spatio-temporal entites and metadata that are structured and exportable (for example ANVIL [51], ELAN [95], EVA [60], GALATEA [71]).

Grudin suggests that tools provide "universal annotation support" - that any media can be annotated with any other media [40]. But the flexibility of spatio-temporal entites and metadata should be appropriate for the context of annotation [5]. For example, the home user will not spend a lot of time on video annotations, preferring simple - yet quick - annotation tools such as YouTube Annotations [3].

- **Tools should allow rich manipulation of media**

  MacKay suggests reordering video segments is useful for annotating batches of events or objects [60]. Cockburn et al. suggests that video segment reordering is useful for video presentation [24].

  However, Grudin et al. highlight the challenge of sharing consistent annotations among video annotators and viewers, while simultaneously allowing annotators to rearrange the video [40].

  Annotations can serve as indexes to manage libraries of videos - the Family Video Archive managed home movies [5], SVAT managed theological video [75], Videotater managed news reports [28], and ELAN managed video for speech research [95].

  Annotation tools can support multiple streams of media. The EVA annotation tool [60] and ELAN [95] supported annotations of keystroke logs, audio recordings and video. Overview visualisations are especially important to help organise annotations in multi-stream media.

- **Tools should support rich navigation within media**

  Potel et al. noticed that tools should allow video playback at any speed and in any direction [71]. Expert annotators reviewing annotations often review video backwards (Appendix E).

  Harrison et al. and Cockburn et al. recommend that navigation controls should be physically close to annotation controls because of the tight relationship between creating and correcting annotations and navigating within video [24, 42].

Tools should allow annotators to skip to portions of video previously annotated, especially when annotating libraries of video. In early formative studies we conducted, annotators suggested annotation tools should provide onion-skin previews of neighbouring frames to reduce navigation among frames.

The MediaDiver demonstrated how annotators can use previews of multi-angled video to navigate between views of a sports game while correcting shared annotations [63]. However, view changes should avoid confusing the annotator.

Cockburn cautions that interruption to video playback during annotation should be minimised, especially during synchronised collaborative annotation [24]. This is also emphasised by MacKay, who points out that annotation in time with video playback is a quick way to create a "first pass" [60].

In some contexts, frame by frame annotation is not necessary. While rotoscopers may want frame by frame navigation [9], home users would find it unappealing to navigate to exact times to annotate video. The Family Video Archive supports annotation on "scenes" - relatively large video chunks that don't require fine navigation [5]. Goldman et al. developed computer vision assisted annotation methods to avoid fine navigation within video while creating spatio-temporal entites [38].

- **Tools should automate annotation when possible**

  Rehatschek et al. point out that many automation techniques can speed up the annotation process, such as object and event detection and segmentation [75]. The Viper tool supported face detection [31], as did Videolyzer [29]. The Family Video Archive [5] and MuLVAT [88] detected scenes within video. TrackMarks [27] and Vatic [92] tracked objects within the video. Videolyzer [29] and ELAN [95] recognised spoken words within video.

  Other automatiion techniques that assist annotators include (1) video stabilisation (LabelMeVideo [96]), (2) checklists of objects yet to be found in video (Expert Interviews, Appendix E, (3) multiangle video to help confirm object identities (MediaDiver [63] and (4) distributed annotation (Vatic [92]).

  Even when automation struggles with video of overlapping objects or objects that disintegrate, active learning techniques can flag the issues for manual annotation [91].

- **No annotation tool can support all annotations**

  Grudin et al. argue that no single annotation tool can support all video annotation contexts [40]. They propose that annotation system designers consider annotation frameworks that support (1) interoperability of annotations across task specific interfaces, (2) flexible structures for annotation spatio-temporal entites and metadata, and (3) flexibility of annotation storage for distributed and single annotator contexts.

  Annotation system designers can tailor tools to particular annotation contexts, however. The Family Video Archive focused on indexing home movies [5]. The MediaDiver catered to annotators watching multiangle sports video from an arena [63].

  Alternatively, annotation system designers could create tools that are customisable. Harrison et al. developed a customisable interface to allow annotators to rearrange interface elements for particular annotation contexts [42]. Topkara et al. developed a "mashable" interface to be incorporated into existing workplace toolsets [89].