

An Immersive System for Browsing and Visualizing Surveillance Video

Philip DeCamp
MIT Media Lab
75 Amherst Street
Cambridge, MA 02139
decamp@mit.edu

George Shaw
MIT Media Lab
75 Amherst Street
Cambridge, MA 02139
gshaw@media.mit.edu

Rony Kubat
MIT Media Lab
75 Amherst Street
Cambridge, MA 02139
rkubat@media.mit.edu

Deb Roy
MIT Media Lab
75 Amherst Street
Cambridge, MA 02139
dkroy@media.mit.edu

ABSTRACT

HouseFly is an interactive data browsing and visualization system that synthesizes audio-visual recordings from multiple sensors, as well as the meta-data derived from those recordings, into a unified viewing experience. The system is being applied to study human behavior in both domestic and retail situations grounded in longitudinal video recordings. HouseFly uses an *immersive video* technique to display multiple streams of high resolution video using a realtime warping procedure that projects the video onto a 3D model of the recorded space. The system interface provides the user with simultaneous control over both playback rate and vantage point, enabling the user to navigate the data spatially and temporally. Beyond applications in video browsing, this system serves as an intuitive platform for visualizing patterns over time in a variety of multi-modal data, including person tracks and speech transcripts.

Categories and Subject Descriptors

H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—*artificial, augmented, and virtual realities, video*; I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—*camera calibration, imaging geometry*; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—*sensor fusion, tracking*

General Terms

Design, Human Factors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'10, October 25–29, 2010, Firenze, Italy.

Copyright 2010 ACM 978-1-60558-933-6/10/10 ...\$10.00.



Figure 1: A synthesized home environment constructed from 11-camera video. HouseFly uses immersive video as a platform for multi-modal data visualization.

Keywords

visualization, immersive video, multi-modal interface, behavioral analysis

1. INTRODUCTION

Cheaper digital video recorders, storage, and processors mean that people will find new reasons to record and access large collections of audio-video data. Of the possible applications, our research focuses on how recording technologies can provide dense observational data on human activity at such a qualitatively greater scale that it transforms how research is performed in behavioral and social psychology, forensics, ethnography, and other fields that rely on human observation. However, the ability to apply cheap recording technologies to scientific research is currently limited by the cost of extracting actionable insights from the raw behavioral recordings. This cost can be lowered both by de-

veloping better machine perception systems to process data automatically and by developing better data mining tools to enable researchers to find patterns of behavior more efficiently. This paper addresses the latter by looking at how 3D graphics techniques can be applied to visualizing and browsing large audio-video corpora.

This work is motivated by several ongoing research projects, one of which is *Human Speechome Project* [15], an effort to study child language development using quantitative analyses of dense, longitudinal audio-video recordings. In the first data collection effort for this project, we installed 11 high-resolution cameras with fisheye lenses and 14 microphones throughout the home of a typical child. Data was collected for three years, resulting in over 90,000 hours of video recordings that represent roughly 70% of the child’s waking experience. Several types of information are now being extracted from the raw audio-video data, including speech transcripts, person tracks, and interaction events. In a separate research project, we installed similar recording systems in several functioning retail service environments and have collected months of video from a very different commercial environment. The goal of this project is to combine longitudinal video data (no audio was captured for privacy reasons) with de-identified electronic transaction data from the retail service centers to gain new insights into the relationships between the types of transactions completed by customers, and how these customers interact with other people—customers and employees—as a function of the architectural design of the retail environment.

In common between these two projects is the nearly comprehensive video coverage obtained by using overlapping, wide-angled cameras. This paper describes a data browsing system, *HouseFly*, that takes advantage of that completeness by dewarping, cutting, and stitching back together all the video streams to synthesize a cohesive 3D reconstruction of the recorded space, as if the video streams were being projected onto a 3D model of the environment. HouseFly utilizes modern rendering hardware technologies so that multiple streams of high resolution video can be processed in realtime while remaining responsive and interactive.

The user controls the video playback in a manner similar to a digital video editing system. Unlike normal video browsing systems, the user can also navigate through space, simulating a first person perspective. In the case of the home environment, where both video and audio recordings are available, audio can be selected from the nearest microphone so that the user also hears sound that is localized to the view. The simulation does not necessarily achieve a high degree of realism as moving objects appear as images projected against a static surface rather than as full 3D models. However, we feel that the content of the video remains easily discernible, and that the fluid navigation and interaction achieve many of the immersive qualities found in first person video games.

Watching video in this way can provide a sense of space and proportion that would be difficult to achieve otherwise. More significantly, many kinds of data, beyond just video, can be combined into this framework in ways that can give a more intuitive, cohesive understanding of the events. We aim to develop multi-modal visualizations that reveal patterns of behavior and social interaction by exploring the spatial-temporal relationships between the modes of data.

After describing the core HouseFly framework, this pa-

per will describe several visualizations and application areas. For the data collected from a home environment, the applications include observing the evolution of a given word as used by a child, visualizing word use as a function of location, and identifying patterns of child-caregiver interaction. For data collected from a commercial environment, applications include visualizing traffic patterns, gaze patterns, and transaction events.

2. BACKGROUND

One of the central focuses of HouseFly is to generate visualizations of data that is both time and space dependent. We build on our previous data browsing interface, *TotalRecall*, also developed for the Human Speechome Project [11]. TotalRecall employed several strategies to summarize both audio and video streams to aid navigation and transcription tasks, but used a 2D interface with conventional audio-video playback. Other visualization systems have displayed synchronized multi-camera recordings alongside an interactive map of the recorded space in order to aid navigation [6, 7, 4]. The system in [4] additionally displays track data directly over the raw video, and also provides a simplified 3D view of the recorded space that overlays the segmented foreground pixels from the video in order to display people moving through a 3D model. A system built by Kaper and Wright renders an elevation map in 3D with overlaid path data, as well as other information, using the z-axis to indicate the time of events [8]. Our system incorporates many of these ideas for displaying multiple video streams, track data, event timelines, and other metadata, while attempting to unify these components further into a single display context.

The earliest reference to *immersive video*, in the sense of generating a fully navigable, 3D environment from multiple video streams, was produced in 1996 by Moezzi et al. [13]. This work uses recordings of sporting events where most of the focus is on generating 3D voxel representations of people captured from multiple angles. In another work, published later by the same group, more effort was placed on stitching together video from cameras pointed at different sections of a shared area to create a more expansive virtual environment [10].

The results of these efforts were limited by the graphics technologies available at the time. The concept reemerged years later as a proposed technique for viewing surveillance video in similar work performed separately by Sawhney et al. [16] and Sebe et al. [17]. In the first paper, the authors demonstrated video being projected live onto 3D model of an area in a town. The system also implemented dynamic camera alignment that enabled video from a moving source, such as from a helicopter, to be projected accurately. This system was later incorporated into a commercial surveillance application.

Given the ubiquity of both digital video and of 3D game engines, there are relatively few other examples of immersive video. Both of the systems described used video from cameras with standard, non-wideangle perspective. Without a large field of view, it is difficult to capture a significant portion of a scene without a prohibitively large number of cameras, and in the existing work of which we are aware, only a small portion of the modeled environment being displayed may be updated at a given time. Immersive video can be made more compelling by applying it to video that captures

the scene from all angles such that the user can explore the environment freely, with minimal consideration for the direction of the cameras. We apply our system to video from ceiling mounted cameras equipped with 180° circular fisheye lenses, which make it possible to capture nearly everything within a concave room and generate a fully panoramic environment with a single camera. Because fisheye video cannot be projected accurately onto a 3D environment with a standard homography, it was necessary to adapt previous rendering methods for HouseFly by making use of the far greater power and flexibility of current graphics processors.

Methods for projecting wide-angle images are well studied. Carrol et al. provided a recent account of the standard projections, which all have advantages and disadvantages [1]. They also describe their own solution, which utilizes limited information about the spatial content of an image to minimize the perceived distortion, at the expense of requiring user input. Although HouseFly is far more focused on speed than on image quality, it might be considered an example of a wide-angle image projection that utilizes a full 3D model.

3. SYSTEM DESCRIPTION

This section first describes the core components of HouseFly, starting with the 3D representation, video playback, and a user interface for spatial and temporal navigation. In the context of this basic framework, other visualizations for various, individual data types will be demonstrated, including audio, transcripts, person tracks, identifications and classifications, retail transactions, and interest maps. Last, this section will describe methods for performing selection, aggregation and retrieval tasks across multiple data types.

3.1 Video Projection

The primary display element of HouseFly is a 3D representation of the recorded space, which is defined by a polygonal model. The models used for this work were built manually with a CAD system, and this work does not address the issue of automatic model acquisition from video. Figure 2(a) shows one of the models used, representing a home instrumented with 11 cameras. While this model is proportionally accurate to the actual house, the model is relatively undetailed and contains only the floors, walls, and stationary furniture.

The details of the environment are supplied by using raw video as textures. Figure 2(b) shows exemplar images from eight of the cameras. The combination of using overhead cameras with fisheye lenses helps to reduce occlusions and minimize the amount of untextured surfaces.

To map the video from the cameras onto the 3D model, each camera is calibrated such that any given *world point* can be mapped onto an *image point* for that camera. Camera calibration is a routine task and there exist many camera models and calibration methods. The camera model used for this work is not notable on its own, but is described here in full because it satisfies two important conditions: it works for very wide fisheye lenses, and is efficient enough to be used for realtime rendering of high resolution scenes. This model does not account for any lens abnormalities and is not suited for applications that demand very high accuracy.

Given the use of fisheye lenses, it is easiest to ignore the lens entirely and instead model the camera sensor as a sphere. The zenith axis of the sphere, Z , exits the front of the camera through the lens. The azimuth axis, X , exits the

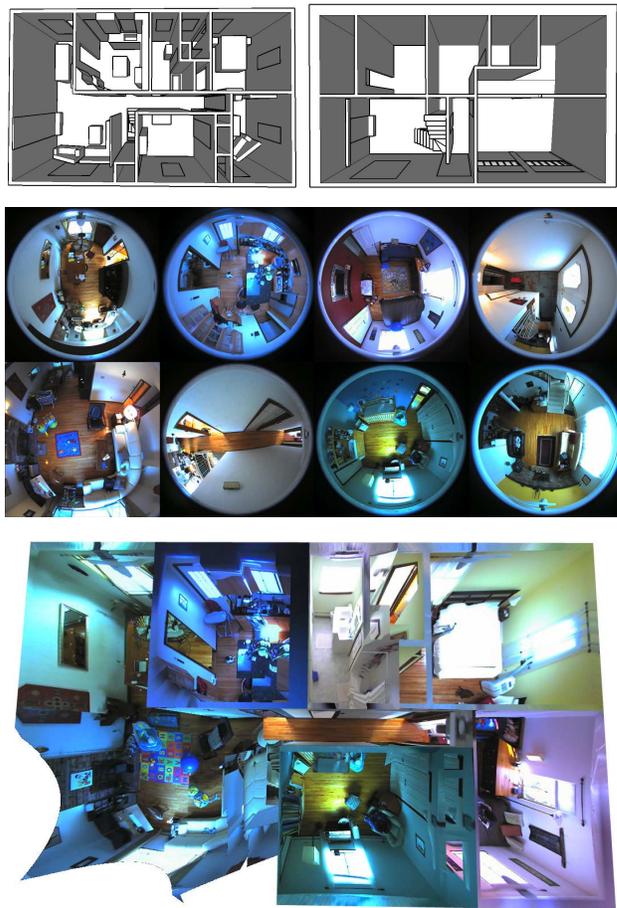


Figure 2: Creating the environment. From top to bottom: (a) polygon model of the two story house. (b) representative images from eight of the cameras installed in the house. (c) textured model. Bottom floor not shown.

right side of the camera. $Z \times X$ is designated Y and exits the bottom of the camera, and the center of the sphere is C . To map a point in world space, P , to an image coordinate, U , P is first mapped onto the camera axes:

$$\tilde{P} = [XYZ](P - C) \quad (1)$$

\tilde{P} is then projected onto the sensor sphere:

$$\theta = \cos^{-1} \frac{\tilde{P}_z}{|\tilde{P}|} \quad (2)$$

$$\phi = \tan^{-1} \frac{\tilde{P}_y}{\tilde{P}_x} \quad (3)$$

where θ is the *inclination* and ϕ is the *azimuth*. Last, (θ, ϕ) is mapped into image coordinates, U :

$$U = \begin{bmatrix} S_x \theta \cos \phi + T_x \\ S_y \theta \sin \phi + T_y \end{bmatrix} \quad (4)$$

where S_x and S_y are scaling parameters, and T_x and T_y are translation parameters. Thus, Equation 4 contains four *scalar* parameters, while Equations 1-3 require six scalar parameters: three to define the center of the sensor, C , and

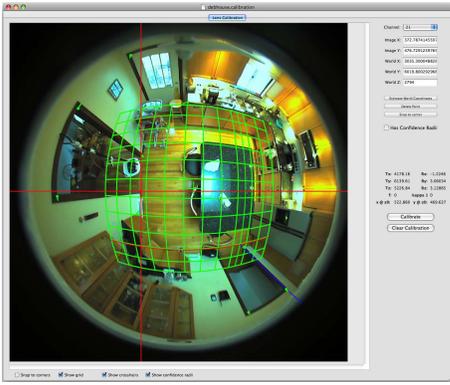


Figure 3: Calibration interface. A user inputs the model-coordinates for a set of points on the image to calibrate one of the cameras.

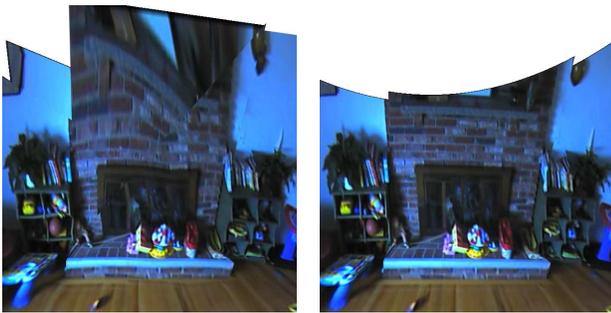


Figure 4: Video projection samples. From left to right: (a) Piece-wise linear projection. (b) Per-pixel projection performed with a fragment shader.

three to define the orientation, $[XYZ]$ (although $[XYZ]$ forms a 3×3 matrix, it is defined as a set of unit-length axes and can be equivalently represented as a set of Euler angles: *yaw*, *pitch* and *roll*). Together, these equations define a mapping function between world points and image points, $f(P : \Theta) \rightarrow U$, where Θ represents the ten camera parameters.

Determining the ten parameters for each camera is performed with a calibration interface shown in Figure 3. Calibration is performed by annotating pairs of correspondence points (P_i, U_i) , where each pair consists of a world-coordinate point on the surface of the 3D model, P_i , and the corresponding image-coordinate point on the image, U_i . These correspondence points are used to define an error for a given set of camera parameters, Θ :

$$e = \sum_i |f(P_i : \Theta) - U_i|$$

This error is minimized, as possible, with a non-linear solver to determine Θ . Our implementation uses the Levenberg-Marquardt solver provided by the MINPACK library [14].

In addition to determining the calibrations, it must be determined which surfaces on the 3D model to texture for each camera. This process is performed by partitioning the model such that each partition is textured by a single camera. In the case of the home environment, there is generally one partition for each room. For control reasons, we define

these partitions manually, although it is straightforward to automatically split and sort each polygon based on its visibility and proximity to each camera.

To render the environment, a frame of video from each camera is projected onto all of the polygons in the corresponding partition. The simplest strategy is to pre-compute the image coordinates for each polygon vertex, and then use linear interpolation to texture each fragment (pixel), just as texturing is normally performed. However, because of the non-linearity of the camera model, this results in severe texture tearing unless the polygons are highly tessellated, as shown in Figure 4(a). We found that performing the texture mapping for each fragment using a fragment shader yielded better results, as in Figure 4(b). Because of the widespread support for fast, programmable shaders in commodity graphics hardware, performing per-pixel mapping did not significantly affect rendering speed.

One limitation of this approach is the problem of occlusions, where there exist sections of the 3D model that are not captured in the video. While these areas might be rendered as a solid color to indicate the missing texture, such patches would stand out against the surrounding textured areas. The current system ignores occlusions and projects video fragments onto multiple polygons where occlusions exist, resulting, for example, in a patch of floor with the same texture as an edge of a table that is positioned between the floor and the camera. While this produces undesired artifacts, the problem is largely unavoidable without complete video coverage of every surface.

Given a fast texture mapping process, performing video playback can be performed by decoding video separately and writing each video stream into the same texture buffer used for rendering. Although outside the scope of this paper, video decoding is the most significant performance bottleneck in this system. All video used in this paper has a resolution of roughly one megapixel and is compressed with motion-JPEG, which requires significant processing power to decode. HouseFly uses a purpose-built rendering engine that relies on the OpenGL[18] library for hardware accelerated graphics. The performance of HouseFly was tested on a computer with eight, 2.8 GHz processor cores, 4 GiB of memory, and a Radeon HD 2600 graphics card. The display resolution was set at 2560 by 1600 pixels, and we used the environment shown in Figure 8, which is comprised of 1512 polygons and 18 partitions corresponding to 18 cameras. The system renders at a rate of 52 frames per second when all video is paused, and 48 frames per second when 6 of the 18 streams of video are playing. However, as additional video streams are played, the available processing resources are eventually exhausted and the frame rate drops sharply, down to 7 frames per second when playing all 18 video streams. Although still useable, this frame rate results in significant “choppiness” when navigating and reduces the responsiveness of the controls. System performance could be greatly improved by selectively playing only the video streams currently in view of, or nearby, the rendering camera, although fundamental scalability limits remain for environments with many cameras.

While many improvements might be made to this projection strategy, it succeeds in performing accurate, multi-image projection in realtime, without the need for preprocessing. This makes the system applicable to very large data corpora, where processing and storing a copy of all the video

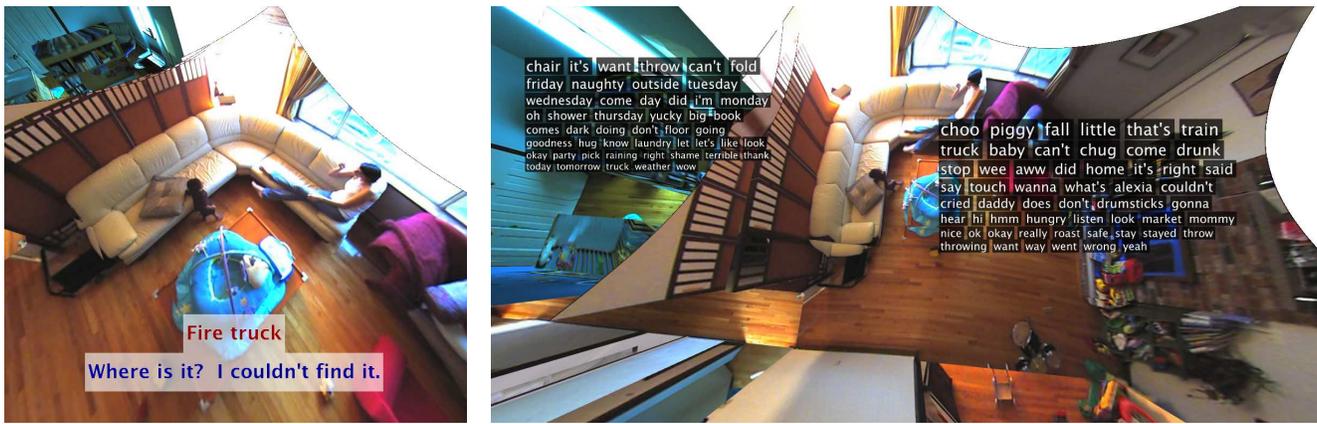


Figure 5: Transcript browsing. From left to right: (a) transcripts shown as subtitles. (b) transcripts combined into word clouds that summarize speech activity and topics in each room of the house.

can be prohibitively expensive in terms of both storage and logistics. Additionally, the system can easily be adapted to play live streaming video, or video from moving cameras.

3.2 User Interface

The challenge of designing a user interface for HouseFly is to provide fluid control over both spatial position (camera position and orientation) as well as time. Two input schemes have been implemented, with different properties.

The first of the two schemes utilizes two commercially available peripherals. A jog-shuttle controller is used for time navigation, providing controls similar to a digital video editing application, with fast transitions between video playback, skimming, and frame-by-frame jogging. For spatial navigation, a 3D motion controller is used that provides six degrees-of-freedom. By pushing and rotating the puck along any axis, the user simultaneously controls position and velocity. Like the jog-shuttle controller, the motion controller requires only one hand, such that the user can simultaneously manipulate both motion and jog-shuttle controllers. However, in addition to requiring atypical hardware, this approach has several disadvantages. We have found that new users find the motion controller difficult to use at first, which required the addition of a “novice” mode that limits the degrees-of-freedom of the device, limits sensitivity, and removes camera roll. But even for experienced users, the motion controller is suited for adjusting the camera, but ill-suited for picking tasks—selecting points on the screen—which is far easier to perform with a mouse.

As a result, HouseFly provides a more conventional keyboard-and-mouse interface that is identical to the setup used in most *first-person shooter* video games. In this setup, the user can “walk” orthogonal to the ground by pressing the “WASD” keys on the keyboard, and control the pitch and yaw of the gaze direction with the mouse. This approach requires two hands to control the perspective, making it more difficult to control time and speed of movement.

One other feature important to the user interface is the ability to store *views*, where a view consists of the position and orientation of the rendering camera. The user can map the current view to a button on the jog-shuttle controller or keyboard, and can then press that button at a later time to quickly return to that view. Similar to the views, a given

point in time may also be stored and retrieved, providing a bookmarking mechanism for events of interest within the data.

3.3 Audio

For the data collected from a home environment, audio recordings were made using 16 microphones installed in the ceilings. Boundary-layer microphones were used, providing clear recordings of the speech occurring anywhere in the house.

While navigating the virtual environment, HouseFly monitors the user’s location and selects the audio stream from the closest microphone for playback. Audio playback is synchronized to the video, and for faster-than-realtime playback, SOLAFS pitch correction is used to enhance comprehension [5]. Audio is automatically silenced if playback exceeds a manually specified speed. In spite of its simplicity, the use of localized audio greatly enhances the sense of immersion and provides a fuller sensation of the people and speech in the recordings.

3.4 Transcripts

As part of the HSP effort to analyze language development, a large portion of the speech data has been transcribed. At the time of this writing, transcripts have been produced for approximately eight months of audio recordings, or about half of the data collected from the home when the child was between the ages of 9 and 24 months.

HouseFly incorporates speech transcripts in several ways. In the most basic application, shown in Figure 5(a), transcripts are displayed as subtitles to aid audio comprehension. Figure 5(b) shows a *word cloud* visualization [9] of frequencies computed directly from the transcripts. HouseFly assembles and updates these word clouds dynamically, providing the user with a summary of speech activity in that room. When skimming the data at very high speeds, the word cloud representation provides a summary of where speech is occurring, and indicates the topics about which people are speaking.

3.5 Person Tracking

In the majority of multi-camera track visualizations, the tracks are drawn as lines that are either overlaid on a floor

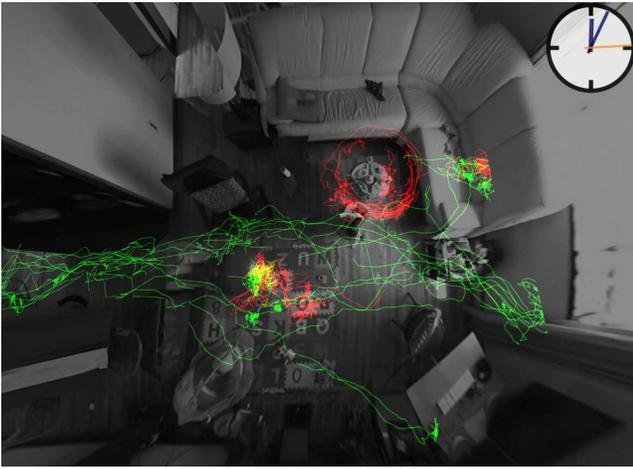


Figure 6: Adding child tracks (red) to caregiver tracks (green) to expose areas of interaction (yellow).

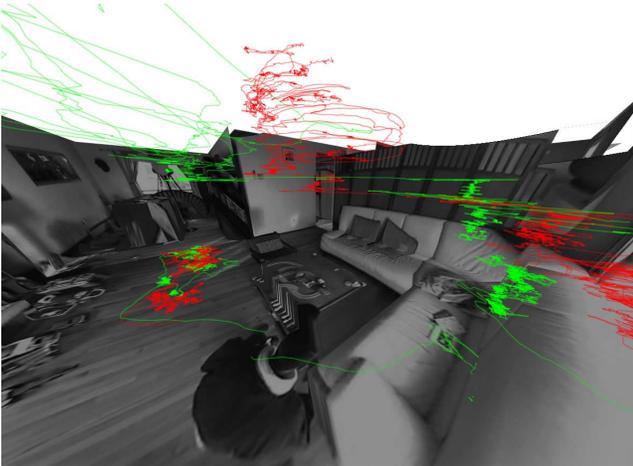


Figure 7: Tracks can be drawn so that they rise vertically as they progress in time to provide a chronological summary of events.

plan that is separate from the video, or overlaid directly on individual streams of video in a way that makes it difficult to understand the patterns of motion at the larger scale as they occur across cameras. HouseFly provides a natural way to render track data in 3D space, across multiple cameras, in the context of the actual video.

Figures 6 and 7 show examples of track visualizations using approximately half an hour of video data from the home environment, where the tracks of the parent and shown in green and the tracks of the child in red. HouseFly does not perform realtime tracking; the data shown here was generated with a separate, semi-automatic tracking interface [2]. When the user enables the track data visualization, the video is rendered in black-and-white to improve the visibility of the tracks.

Of particular interest to observing child development are the instances when the child and caregiver are in close proximity, which indicates that they are more likely to be interacting and the child is attending to the speech and actions

of the caregiver. These areas of overlap are referred to as *social hot spots*. By rendering the tracks using an additive color blend, these spots emerge as yellow areas, as seen in Figure 6. Visualizing these social hot spots is a way to enhance the user’s perception of everyday interactions. Section 3.7 discusses selecting and exploring these interactions in greater detail, paving the way for their use as defining features in the study of child language acquisition.

Figure 7 shows the same tracks, but where the vertical position is determined by each line segment’s place in time such that the tracks rise from the floor as time moves forward; a technique demonstrated in [8]. This visualization is intended to give a better sense of the timing of the events; the user can quickly see that the child and caregiver were first together in the middle of the room, then moved to the couch, and then the caregiver began traveling to the kitchen as the child circled his toy. Although the full structure of the tracks may not be obvious in this 2D image, the 3D shape is more apparent as the user navigates the space and views the scene from the different angles. Additive color blending is disabled for this view because the variable height of the tracks results in track segments that overlap on screen but are not proximate in space.

Figure 9 shows approximately four hours of track data extracted from video collected at a retail service environment. This track data was generated with a fully automatic tracker. Ignoring for a moment the coloring of the tracks, this kind of visualization might be used in a retail space to look at overall traffic patterns within the space, to indicate areas of the store that might be underutilized, and to look at changes in the usage of the space at different times.

The coloring in this image indicates a classification over customers (red) and employees (green). This classification was generated by an automatic process, which starts with a point discretization step. Given a set of tracks, all the location points are collected, without regard to the relationships between points, and processed by a modified LBG *vector quantization* algorithm [12]. The output of the LBG algorithm is a placement of the nodes such that each node has an approximate number of associated points; thus, areas with denser traffic have more nodes and sparser areas have fewer. The total number of nodes can be adjusted as a parameter, and can be thought of as the resolution at which the discretized track data is realized. The nodes are given arbitrary indices. Writing expressions over these indices then provides the basis for matching tracks from a corpus to behaviors of interest.

A training set was then created by hand-labeling a subset of the quantized track data. This data was then used to train a probabilistic model for each the customers and employees. In addition to the track data, several color histograms were generated for each target during tracking at different times. Using the same training set, the color histograms were labeled and used to train an SVM. The likelihood scores generated by both these classifiers were then used as input for a third and final classifier, which generated the track classifications shown. Discarding the low-confidence instances, this classifier achieved over 90% accuracy for leave-one-out cross-validation.

In addition to using track data as a basis for action classifications, the retail spaces shown provide additional information in the form of de-identified transaction records of the customers. Figure 11 shows HouseFly as it renders the on-

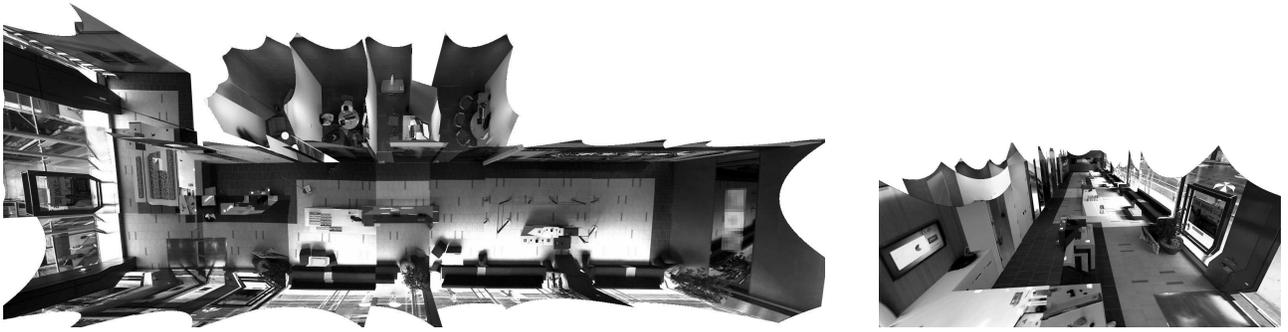


Figure 8: Retail service environment constructed from 18 cameras installed at a functioning business.

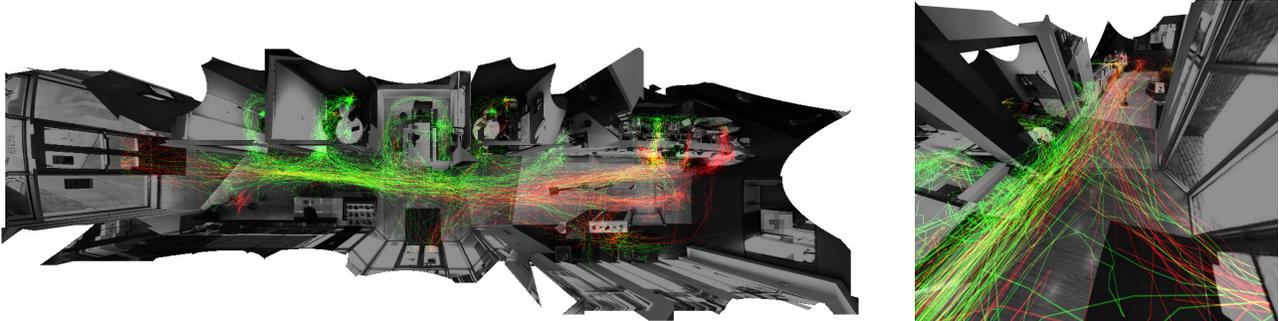


Figure 9: A different retail service environment from the one shown above, this environment is constructed from 11 cameras. This image also shows several hours of track data that have been classified automatically as customers (red) or employees (green).

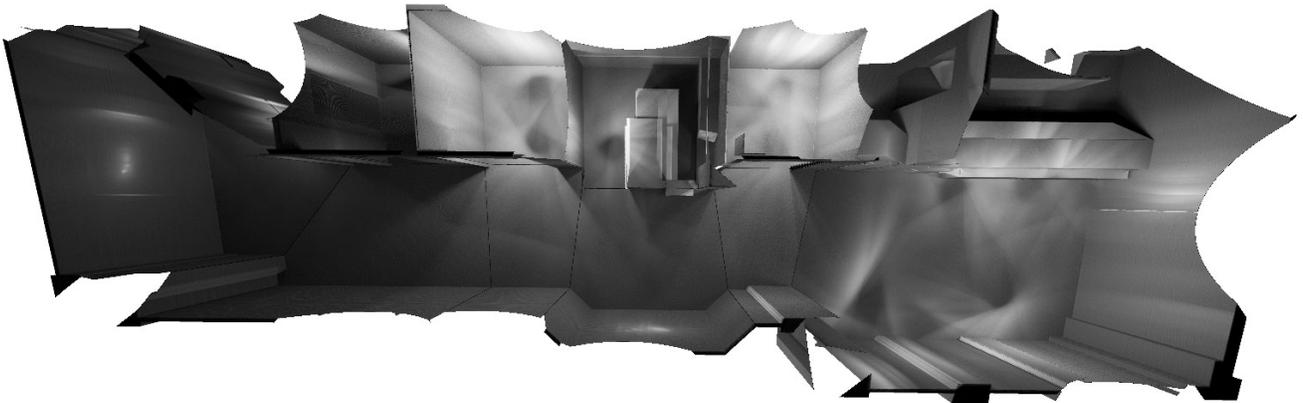


Figure 10: Interest map visualizing the frequency with which people look at different parts of the store.

going transactions as progress bars above each employee. By itself, merely indicating the occurrence of a given transaction may be of limited use, but by providing access to such behavioral features extracted from multiple sources, HouseFly provides a framework for combining data types and searching for possible patterns. Transaction events might be combined with any number of features derived from the track data, such as time spent waiting in line, interactions with employees, or measurements of the crowdedness of the store, in order to answer questions about how different aspects of the environment affect customer behavior.

3.6 Interest Mapping

Gaze direction indicates a person's attentional focus and makes a particularly useful feature for behavioral analyses. For analyzing child behavior, the gaze of a child can indicate when a child is attending to a caregiver versus a nearby toy. For commercial spaces, aggregate gaze data can be used to estimate the areas of interest in a space that receive the greatest amount of visual attention.

In order to display aggregate gaze information, an *interest map* visualization described in Farenzena et al. [3] was adapted for use in HouseFly. Similar to their approach, we

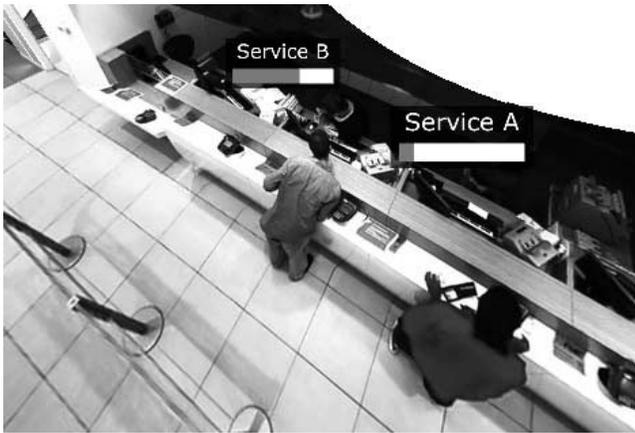


Figure 11: Progress bars show the ongoing transactions between customers and employees.

estimate gaze coarsely by computing the velocity of a person from the track data and using the velocity as the primary indication of body orientation. For a given point in time, the distribution over gaze direction is estimated as a Gaussian centered on the velocity vector with a falloff that is proportional to the person's speed; a distribution that assumes a person moving quickly is more likely to be looking straight ahead, and that a person standing still is more likely to be looking around.

This process has not been evaluated, but is being used in this instance to explore possible visualizations. Figure 10 shows a screenshot of HouseFly displaying an *interest map*, where the brightness of each point of the model's surface is determined by the gaze estimates and indicates the likelihood that that point is in a person's field of view.

3.7 Search and Retrieval

This paper has so far enumerated a variety of data types and visualizations that HouseFly brings together into a single display area. While the different visualizations have been shown as separate elements, they represent shared behavioral events that occur at a given time and place, making it natural to show them in any number of combinations. Just as the data types can be displayed together, they can also be used together for multi-modal retrieval tasks. This section describes two of the ways in which HouseFly can select and display subsets of the data.

The first selection mechanism uses the track data to retrieve data based on person locations. Earlier in Figure 6, *social hotspots* were shown and described as areas in the environment that a child and caregiver co-occupied for some period of time. A user that identifies such an area might want to look more closely at just the data associated with that interaction. The user can do this by clicking on the model to place a sphere, and then drag the mouse away to expand the sphere so that it encapsulates the area of interest. As the user drags the sphere, only the person tracks that intersect the sphere are rendered, as shown in Figure 12(a). For a full day of track data collected from one of the more densely populated retail environments, the time required for this selection is small compared to the time required to render the frame, making it possible to respond fluidly to the user and update the display dynamically as the user per-

forms the selection gesture. When the user releases the mouse, the system computes the set of time intervals that contains completely the intersected tracks, essentially using the track data to map a selected spatial region into a set of time intervals that define the actual result set.

Figure 12(b) shows HouseFly as it displays a result set. A timeline at the bottom of the screen indicates the range of the results, where the red regions indicate areas with selected data and the small, down-pointing arrow indicates the user's current time position. In this case, most of the data selected occurred within the first third of the timeline, although several stray tracks intersected briefly later in time. Given the result set, the user might browse the contained audio-video in order to watch the events fully, or might just look at the word clouds to see a summary of the interaction speech, also shown in Figure 12(b).

The second selection mechanism is similar to many video retrieval interfaces that have been designed for television media, where the first step of a query is often a text search over the closed-caption data embedded in the video corpus. The data in this paper includes transcripts for the home environment, and transaction data for the retail environments, both of which can be indexed in a similar manner. In the case of the home environment, a user might want to observe a child utilizing a single word over time to analyze speech

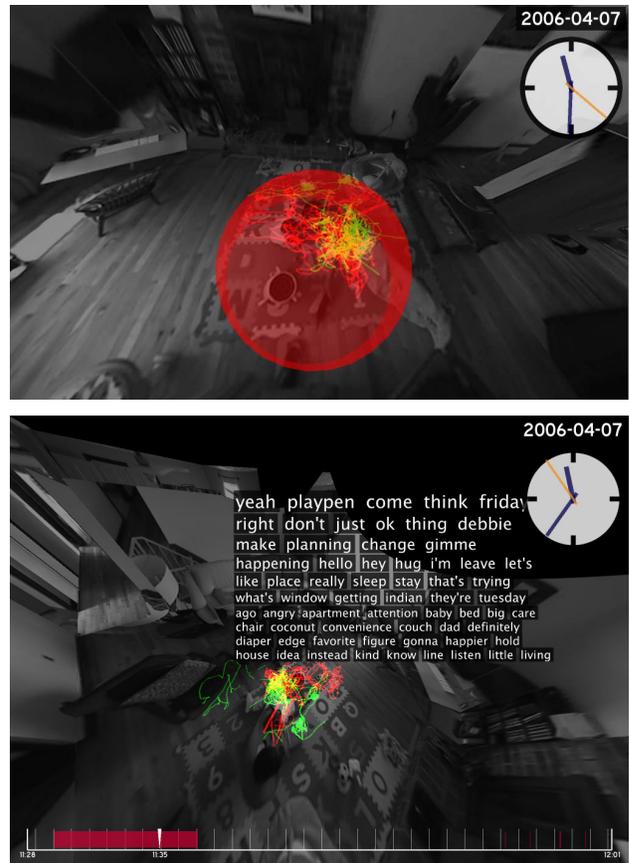


Figure 12: From top to bottom: (a) The user can draw a sphere anywhere on the screen to select a region of space. (b) HouseFly shows a subset of the data determined by the user's selection.

development and look for patterns in the child’s usage and context. The user can perform this by typing in a word of interest—for example, “ball”—and the system will scan the transcripts for all instances of that word and collate a result set containing those instances.

The result set generated by a keyword search is similar to the result set generated by the location search, except that each time interval is associated with a location in the house where the utterance was recorded. As the user browses the selected segments, HouseFly automatically transitions the user’s position using the view system described in Section 3.2. In either selection process, the user can manually traverse from one selected segment to the next, or play them all back sequentially as a montage.

4. CONCLUSIONS AND FUTURE WORK

The primary contribution of this work is to demonstrate the possibilities for viewing and combining information when the underlying data is large enough and rich enough to generate an immersive, virtual world. This paper has shown numerous visualizations and interface techniques that may be used within an immersive video environment and that operate over a diverse assortment of data; from continuous video recordings to discrete transaction events; individual utterances to aggregations that span from hours to years. Rather than develop an interface that creates partitions for every data type—an area for the floor plan, one for each video stream, one for transcript summaries, and so forth—we have attempted to create a multi-modal browsing system that can combine these elements in a single context.

This is motivated in part by the many kinds of space-dependent data that fit naturally into a 3D environment, but also by the patterns that can emerge from bringing different sources of information together. Something as simple as displaying track data on top of the raw video can be cumbersome with multi-camera video, let alone selecting and isolating interactions that occur across multiple rooms. In the case of the retail service environments, viewing the tracks in the context of the building reveal not just where people have been, but which areas and resources in the building are being utilized. If an interesting or anomalous path through the space is found, then the user can go back and watch the underlying raw data to see what happened. Similarly, in the home environment, the user can quickly skim through hundreds of instances of speech events, pausing at any point to look more closely at the context and surrounding events, and perform additional queries based on spatial position.

A secondary contribution of this paper is the use of HouseFly to demonstrate the different layers of meta-data that can be extracted from dense, audio-video recordings, and to provide a few examples of how this kind of data can drive behavioral research. The video shown in this paper was used to produce person tracks, and then person classifications on top of that. The audio recordings were used to produce transcripts, which are also being classified in ongoing work and being used to extract prosodic features like pitch and amplitude. As these processes become more automated, it becomes more affordable to combine a person’s identity, movement, speech, tone, actions and interactions to analyze the fine-grained speech development of an individual child, or the lunchtime traffic patterns in a busy store, or any number of aspects of human life that might unfold over the course of a single minute, or several years.

HouseFly has not been developed to improve the performance of some known task, but as an experimental system to explore new concepts in data browsing. Future work will need to shift more towards matching and adapting the capabilities of HouseFly to emerging real world applications, whether they involve analyzing one of the corpora mentioned in this paper, or studying data collected from another context for different research goals altogether. It is hoped that as surveillance recordings become denser and cheaper to obtain, this research will find applications in behavioral analytics where researchers can benefit from an intuitive, multi-modal system for navigating dense behavioral data.

5. ACKNOWLEDGEMENTS

We thank Ken Jackowitz, research affiliate at the MIT Media Lab, for his contributions and support. This research was supported by the U.S. Office of Naval Research, award no. N000140910187.

6. REFERENCES

- [1] R. Carroll, M. Agrawal, and A. Agarwala. Optimizing content-preserving projections for wide-angle images. In *SIGGRAPH ’09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM.
- [2] P. DeCamp and D. Roy. A human-machine collaborative approach to tracking human movement. In *Multi-Camera Video. ACM International Conference on Image and Video Retrieval*, 2009.
- [3] M. Farenzena, L. Bazzani, V. Murino, and M. Cristani. Towards a subject-centered analysis for automated video surveillance. In *Proceedings of the 15th International Conference on Image Analysis and Processing*, page 489. Springer, 2009.
- [4] A. Girgensohn, D. Kimber, J. Vaughan, T. Yang, F. Shipman, T. Turner, E. Rieffel, L. Wilcox, F. Chen, and T. Dunnigan. Dots: support for effective video surveillance. In *MULTIMEDIA ’07: Proceedings of the 15th international conference on Multimedia*, pages 423–432, New York, NY, USA, 2007. ACM.
- [5] D. Hejna and B. R. Musicus. The solafs time-scale modification algorithm. Technical report, BBN, 1991.
- [6] L. Huston, R. Sukthankar, J. Campbell, and P. Pillai. Forensic video reconstruction. In *VSSN ’04: Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 20–28, New York, NY, USA, 2004. ACM.
- [7] Y. Ivanov, C. Wren, A. Sorokin, and I. Kaur. Visualizing the history of living spaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1153–1160, 2007.
- [8] T. Kapler and W. Wright. Geotime information visualization. In *INFOVIS ’04: Proceedings of the IEEE Symposium on Information Visualization*, pages 25–32, Washington, DC, USA, 2004. IEEE Computer Society.
- [9] O. Kaser and D. Lemire. Tag-cloud drawing: Algorithms for cloud visualization. *CoRR*, abs/cs/0703109, 2007.
- [10] A. Katkere, S. Moezzi, D. Y. Kuramura, P. Kelly, and R. Jain. Towards video-based immersive environments. *Multimedia Syst.*, 5(2):69–85, 1997.

- [11] R. Kubat, P. DeCamp, and B. Roy. Totalrecall: visualization and semi-automatic annotation of very large audio-visual corpora. In *ICMI '07: Proceedings of the 9th international conference on Multimodal interfaces*, pages 208–215, New York, NY, USA, 2007. ACM.
- [12] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantization design. *IEEE Transactions on Communications*, 28(1):84–94, 1980.
- [13] S. Moezzi, A. Katkere, D. Y. Kuramura, and R. Jain. Reality modeling and visualization from multiple video sequences. *IEEE Comput. Graph. Appl.*, 16(6):58–63, 1996.
- [14] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. User guide for MINPACK-1. Report ANL-80-74, 1980.
- [15] D. Roy, R. Patel, P. DeCamp, R. Kubat, M. Fleischman, B. Roy, N. Mavridis, S. Tellex, A. Salata, J. Guinness, M. Levit, and P. Gorniak. The human speechome project. In P. V. et al., editor, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, pages 192–196. Springer, 2006.
- [16] H. S. Sawhney, A. Arpa, R. Kumar, S. Samarasekera, M. Aggarwal, S. Hsu, D. Nister, and K. Hanna. Video flashlights: real time rendering of multiple videos for immersive model visualization. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 157–168, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.
- [17] I. O. Sebe, J. Hu, S. You, and U. Neumann. 3d video surveillance with augmented virtual environments. In *IWVS '03: First ACM SIGMM international workshop on Video surveillance*, pages 107–112, New York, NY, USA, 2003. ACM.
- [18] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.