

HeadLock: Wide-Range Head Pose Estimation for Low Resolution Video

Philip DeCamp

B.S., Massachusetts Institute of Technology (2004)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences
at the
Massachusetts Institute of Technology

February 2008

© Massachusetts Institute of Technology 2007. All rights reserved.

Author _____
Program in Media Arts and Sciences
October 5, 2007

Certified by _____
Deb Roy
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by _____
Deb Roy
Chair, Program in Media Arts and Sciences

HeadLock: Wide-Range Head Pose Estimation for Low Resolution Video

by

Philip DeCamp

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on October 5, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

This thesis focuses on data mining technologies to extract head pose information from low resolution video recordings. Head pose, as an approximation of gaze direction, is a key indicator of human behavior and interaction. Extracting head pose information from video recordings is a labor intensive endeavor that severely limits the feasibility of using large video corpora to perform tasks that require analysis of human behavior. *HeadLock* is a novel head pose annotation and tracking tool. Pose annotation is formulated as a semi-automatic process in which a human annotator is aided by computationally generated head pose estimates, significantly reducing the human effort required to accurately annotate video recordings.

HeadLock has been designed to perform head pose tracking on video from overhead, wide-angle cameras. The head pose estimation system used by HeadLock can perform pose estimation to arbitrary precision on images that reveal only the top or back of a head. This system takes a 3D model-based approach in which heads are modeled as 3D surfaces covered with localized features. The set of features used can be reliably extracted from both hair and skin regions at any resolution, providing better performance for images that may contain small facial regions and no discernible facial features.

HeadLock is evaluated on video recorded for the *Human Speechome Project* (HSP), a research initiative to study human language development by analyzing longitudinal audio-video recordings of a developing child. Results indicate that HeadLock may enable annotation of head pose at ten times the speed of a manual approach. In addition to head tracking, this thesis describes the data collection and data management systems that have been developed for HSP, providing a comprehensive example of how very large corpora of video recordings may be used to research human development, health and behavior.

Thesis Supervisor: Deb Roy

Title: Associate Professor of Media Arts and Sciences, Program in Media Arts and Sciences

**HeadLock: Wide-Range Head Pose Estimation
for Low Resolution Video**

Philip DeCamp

The following people served as readers for this thesis:

Thesis Reader _____

William T. Freeman
Professor
Dept. of Electrical Engineering and Computer Science, MIT

Thesis Reader _____

V. Michael Bove
Principal Research Scientist
Program in Media Arts and Sciences

Contents

1	Executive Summary	9
2	Background	15
2.1	The Human Speechome Project	15
2.2	Analyzing Thousands of Hours of Video Data	17
2.3	Head Pose Annotation	19
3	Data Collection	23
3.1	Introduction	23
3.2	The Sensor Network	23
3.3	Privacy Interface	27
3.4	Data Processing and Recording	28
3.5	Post Processing	35
3.6	Data Storage	35
4	Data Management	39
5	Head Pose Annotation	43
5.1	Annotation Interface	43
5.2	Head Tracking Overview	45
5.3	Position Tracker	48
5.4	Feature Extractor	51
5.5	Pose Estimator	53
6	Evaluation	57
6.1	Example Results	61
6.2	Results by Subject	63
6.3	Initialization Effects	65
6.4	Segmentation Errors	67
6.5	Estimation Confidence	68
6.6	Comparisons to Other System Configurations	69
6.7	Head Annotation Efficiency	70
7	Future Work	75
8	Conclusion	81

1 Executive Summary

The advancement of data storage technologies, combined with the increasing availability of affordable digital recording devices, has generated a need for analysis technologies to exploit the growing collections of audio-visual recordings. Already in organizations that have large scale surveillance systems, including airports, law enforcement agencies, and retail stores, significant resources are employed to record and archive huge collections of video recordings, yet extracting useful information from such data remains costly and tedious. The ability to economically extract high-level information from recordings – information that includes the locations, identities, and behaviors of people – could benefit existing operations as well as open up new application domains. For example, the imminent boom within the elderly care industry might find applications in which cameras are employed in assisted living environments in order to monitor and detect potential health problems. Early diagnosis of behavioral disorders, such as Autism Spectrum Disorders (ASD), might be made more efficient by introducing tools that can analyze child behavior without the intense involvement of a health specialist. Currently, the applicability of such systems is less limited by the cost of sensors as it is by the lack of effective tools to utilize such data.

This thesis focuses on data mining technologies to extract human behavioral data from large collections of video recordings. In particular, I propose a method for efficiently extracting head pose information. Head pose, which includes the position and orientation of the head, acts as the best approximation for gaze direction in cases where eye tracking is not feasible, such as in low resolution video or video taken from an overhead perspective. Because gaze direction is a key indicator of attentional focus, it provides valuable insight into the experience, actions, and interactions of humans.

This research is particularly motivated by the video analysis challenges posed by the Hu-

man Speechome Project (HSP)[31]. HSP represents a novel research methodology in which human language development is studied by analyzing longitudinal audio-visual recordings of a developing child. In the case of language development, analyzing attentional focus may provide numerous insights into the patterns of interactions that are involved in the learning process. Figure 1.1 shows several frames taken from a video recording of an interaction between a child and caregiver. The caregiver first calls out to the child to draw the child’s attention, then points to an object on the other side of the room and asks the child to identify it. Using the orientation cues given by the caregiver, the child shifts his attention to the object, identifies it, then returns his attention to the caregiver, possibly to receive confirmation. The ability to more efficiently extract gaze direction from such video may enable more detailed analysis of child-caregiver interactions and provide a better understanding of the role such interactions play in language development.

While head pose may be manually annotated, this is a very time consuming endeavor and is not practical for large collections of video recordings. However, existing computer vision technology is not capable of automatically estimating head pose with the level of accuracy that is required by projects like HSP. For this thesis, the manual and automatic approaches are combined to create a system, HeadLock, that adopts a strategy of human-machine collaboration to efficiently annotate head pose in video. HeadLock provides an interface for manual annotation that enables a human operator to quickly indicate the identity, position, and orientation of a head in a frame of video. Using the manual annotation as a seed, HeadLock uses a novel head pose estimation system to propagate the head pose to as much of the surrounding video as possible. When HeadLock fails to create an accurate annotation, the human operator may create additional annotations to correct and guide the automatic estimation process. The head pose estimation system reduces the number of annotations required by the human operator, while the human operator ensures that the data is annotated accurately.

None of the existing head pose estimation systems adequately address the difficulties present in the HSP video. Figure 1.2 shows several typical examples of persons appearing in the HSP corpus. Due to the camera placement and wide-angled lenses used for recording, heads



(a) Caregiver: “Hey!” The caregiver initiates interaction by drawing the child’s attention.



(b) Caregiver: “What’s that over there?” After the child turns his attention, the caregiver points to an object in the room.



(c) Child: “Green ball.” The child orients his attention according to the cues given by the caregiver and identifies the object.



(d) Caregiver: “Oh.” The child returns attention to caregiver and receives feedback.

Figure 1.1: A typical interaction between caregiver and child is displayed by visualizing the gaze of the participants. The caregiver has the child identify an object in the room by first orientating the child towards himself, and then to the object.



Figure 1.2: Typical examples of child-caregiver interactions found in the HSP video. The resolution and quality of the video provides significant challenges to computer vision.

appearing in the video have limited resolution, often less than 30 pixels on a side. The recorded video contains a wide range of lighting conditions and, under low light conditions, includes significant motion blur for moving subjects. The floors throughout the house are predominately wood and have a color that closely matches the skin tone of the primary members of the house, making segmentation difficult for much of the video. People usually appear in one camera at a time, eliminating the possibility of utilizing depth cues from multiple camera recording. Together, these factors pose a formidable challenge to accurate head pose estimation.

The head pose estimation system used by HeadLock is a generative model in which textured, 3D mesh models of human heads are used as structured priors. The search over the space of pose parameters is performed by rendering a 3D head model at different scales and orientations and computing the distance between the rendered head and the head appearing in the video. The parameter search is preceded by a tracking step to improve performance in the presence of inter-frame position changes, and a segmentation step that provides greater invariance to lighting conditions and target appearances.

HeadLock was evaluated on a representative set of video segments selected from the HSP

corpus. For the evaluation, a human annotator provided an single “seed” annotation, which HeadLock propagated forward for each subsequent video frame until it produced an inaccurate pose estimate. On average, the system was able to track a human head for 2.47 seconds. If a human annotator is required to create one annotation for each video frame that HeadLock fails to annotate, this figure indicates that HeadLock may enable head annotation tasks to be performed at 10 times the speed of a similar, manual system that does not use pose estimation. Furthermore, the evaluation reveals several ways in which HeadLock might be greatly improved without significantly altering the overall approach taken to annotation or pose estimation.

In addition to HeadLock, this thesis also discusses the data collection and data management systems that were created for HSP. The data collection system addresses the technical issues of longitudinal audio-video recording and the practical issues of recording a family in a home environment, aspects that affect the nature of the HSP data. The data browsing system provides data management and browsing tools on which HeadLock is built, and addresses many of the less obvious, but crucial challenges involved with annotating dense, longitudinal recordings. By covering data collection, management, and analysis, this thesis provides a comprehensive look at the use of very large video corpora to research human development, health and behavior.

Chapter 2 will provide an overview of related work in both human development research and computer vision. Chapter 3 describes the HSP data collection system. Chapter 4 provides a brief description of the data management platform on which HeadLock is built. Chapter 5 describes HeadLock in detail, covering both the manual annotation interface and the head pose estimation system. Chapter 6 describes the HeadLock evaluation. Chapter 7 discusses future work. Chapter 8 ends the thesis by summarizing the results and discussing the conclusions.

2 Background

2.1 The Human Speechome Project

This thesis takes place in the context of the Human Speechome Project [31], a research initiative led by Professor Deb Roy and the Cognitive Machines group at the MIT Media Lab. The high-level goal of HSP is to observe and computationally model the course of language development for a single child at an unprecedented scale. In order to accomplish this goal, we have instrumented a child’s home so that a very large portion of what the child hears and sees is recorded from birth until two to three years of age. The child’s audio-video experiential record will then be used as input to develop computational models of language learning.

For this project, we have installed video cameras and microphones in each room in the home of a child and his parents. Audio-video recordings are made for approximately eight to ten hours a day, throughout the house. To date, recordings have been made for the first 25 months of the child’s life. In this time, we have collected approximately 40,000 hours of video and 120,000 hours of audio representing the majority of the child’s waking experience. In terms of completeness and detail, this set of recordings of a single child already surpasses previous experiments by more than two orders of magnitude.

A primary motivation for collecting a near-complete record of a child’s early linguistic experience is to extend current methods of studying language acquisition. HSP harnesses the strengths of several existing methods and attempts to establish a new methodology that leverages the current technological advancements in data collection, mining, and analysis.

Language acquisition has often been studied by bringing children into a laboratory envi-

ronment to collect observational recordings. Although this practice has been invaluable for researchers, it does not establish the natural home environment of the child, and thus the observed speech and interactions of the child take place in an altered context. As a result, many researchers have argued that collecting observations in a child’s home, *in vivo*, is more suitable than in a laboratory. As stated by J. Bruner [9]:

I had decided that you could only study language acquisition at home, in vivo, not in the lab, in vitro. The issues of context sensitivity and the format of the mother-child interaction had already led me to desert the handsomely equipped but contrived video laboratory... in favor of the clutter of life at home. We went to the children rather than them come to us. Generally, child observation can be categorized as either in vivo, in the home or natural environment of the child, or in vitro, in a laboratory or artificial environment.

Despite the trend towards *in vivo* observation of language learning, to date, the quality and quantity of audio-video data collected from such studies have been poor. Scientists conducting these studies may only record a child in her natural home environment for one to two hours a week. These observations, spaced several days or weeks apart, can easily miss dramatic shifts in linguistic ability. Additionally, these studies are prone to strong observer effects resulting from the disruptive presence of observers in the home. As Tomasello and Stahl point out, due to the dearth of high-quality, dense longitudinal data, the “fine-grained effects of experience on language acquisition are poorly understood” [35].

The importance of observing the details of language development as it unfolds over time in the home environment is well established in the field of language acquisition [6, 29]. Historically, researchers used longitudinal diaries to observe a child’s linguistic development in her everyday context. Longitudinal diary studies are long term studies that require the caregiver of the child to record a limited set of significant events in the child’s life over the course of several years. These records typically exist as written documentation and do not require extensive codification. Because diary studies span a long period of time, they can capture trends in a child’s language development in a way that can be more effective than short-term lab studies. However, diary observations were typically text-based, biased toward the caretaker’s opinions, and sparse. As a result, these studies, like short-term lab

studies, were unable to capture the fine-grained effects of experience.

HSP overcomes this problem by capturing as much of the child’s waking moments as possible, amassing a comprehensive set of audio-visual footage representing the early stages of a single child’s language development. Ultra-dense observational recording is enabled by the recent surge in availability of digital sensing and recording technologies. By leveraging these technological advances, HSP can capture rapid changes in linguistic abilities, such as the use of new vocabulary and phrases, as well as trace overall language usage patterns. The density of the HSP corpus represents a significant improvement over existing corpora. For example, the CHILDES archive of speech recordings covers less than 1.5% of a child’s complete linguistic experience and far less visual context [25]. This density lowers the random effects due to sampling errors and increases the possibility for theoretical advancements in the field of language acquisition. By collecting dense audio-visual data over an extended period of a child’s early life, HSP proposes a methodology that couples the context-rich approach of current *in vivo* studies with high quality audio-visual observations that laboratory settings allow.

2.2 Analyzing Thousands of Hours of Video Data

While HSP has succeeded in collecting a unique data set, its success as a scientific endeavor depends on our ability to analyze the raw data. For example, HSP researchers may wish to track the history of a particular word as it’s used by the child. When the child says “cup,” does he use it as a request for something to drink, or as a statement that he sees a cup? Does the child use the word differently around his mother than around his father? Answering these questions will certainly require extracting speech from the raw audio, but the video recordings also play a critical role in establishing the context in which the language occurs. Video can provide the locations and movements of the recorded subjects and the interactions between them, but extracting this information from the raw video requires human effort.

Of particular interest is the analysis of interactions that express intersubjectivity. Fig-



(a) An example of primary intersubjectivity, in which the child interacts with a caregiver face-to-face. (b) An example of secondary intersubjectivity, or joint-attention, in which the child and caregiver share focus on a tertiary object.

Figure 2.1: Intersubjectivity, including shared attention, context, and understanding, that may inform language development.

Figure 2.1(a) shows an example of *primary intersubjectivity*, which involves the child interacting with a caregiver face-to-face. Figure 2.1(b) shows an example of *secondary intersubjectivity*, also referred to as *joint attention*, in which the child and caregiver share focus on an object. The importance of such interactions to language development is well established [34, 10] and may be described as the process through the which the child and caregiver establish a shared context that facilitates communication.

An analysis of these interactions minimally requires knowledge of the participants' attentional focus. While eye gaze may be the most useful indicator of attention, head orientation offers a suitable approximation in cases where detailed images of the face are not available for eye tracking. The images in Figure 2.1 do not reveal the exact orientation of the participants' eyes, but the focus of the subjects is still evident in the orientation of the head. Thus, extracting the position and orientation of heads in the video may provide extremely useful information for analyzing patterns of joint-attention and other important interactions.

Much of the research that has been performed for HSP has been to address the substantial challenges of extracting information from a corpus that contains hundreds of thousands of hours of recordings. These efforts resulted in the development of *TotalRecall* [22], a data management tool that provides general browsing, playback, and annotation capabilities

and is able to manage several years of continuous, multitrack data. TotalRecall is also the platform that is used by HSP researchers to develop tools for annotation and analysis. The primary focus of this thesis, HeadLock, is implemented as an extension of the TotalRecall system that leverages the services and functionality provided to address practical issues of locating and viewing data of interest, and organizing large-scale annotation tasks that require multiple annotators.

2.3 Head Pose Annotation

While HeadLock is still a work in progress and does not yet provide an annotation interface, it has been designed to act as an interactive system. The approach taken by HeadLock draws from previous work on interactive tracking systems that enable user-driven annotation. A system implemented by Agarwala et al. performs semi-automatic tracking of object contours that allows the user to interactively refine the contour estimates [1]. However, this system only performs interpolative tracking, requiring the user to annotate at least two keyframes before tracking is performed on the intermediate frames. In [13], Dakss describes the *HypeActive* tool for video markup that allows a user to rapidly indicate a semantically significant segmentation over several frames of video and perform bidirectional tracking of those segments throughout a video sequence. This interaction is similar to HeadLock, in which the user provides one or more key annotations and the system propagates the annotations throughout a segment of video. Also similar to HeadLock, HyperActive maintains a target database that is used to perform tracking of individual objects across disjoint video segments. HeadLock applies the same interactive tracking techniques to the domain of head pose annotation.

The cameras used in HSP are positioned overhead and use wide-angled lenses. As a result, heads appear in nearly every orientation, have limited resolution between 20 to 70 pixels on a side, often lack discernible facial features, and often show only the top or back of the head. To perform head pose tracking on this corpus, the tracking algorithm must operate on low resolution images, and must track heads over the entire orientation domain (yaw,

pitch and roll).

Head pose estimation has long been studied in the field of computer vision, both explicitly in systems that perform just head tracking, and implicitly in systems that perform face detection or full body pose estimation. Approaches to pose estimation include local feature approaches that infer head orientation from the arrangement of multiple, locally detected features, and global approaches that use the entire head image. HeadLock uses a head representation that is derived from local feature approaches that are 3D model-based, but selects features and uses search strategies that are better associated with global template matching approaches.

3D model based approaches model the head as a 3D surface, covered with localized features. These models can be constructed manually, or learned from annotated data by placing observed features onto the corresponding location of the head model. Pose estimation is then performed by locating similar features on an image, and finding the pose of the 3D model that produces the best match between model and image features. Depending on the precision of feature localization, this approach can produce highly accurate pose estimates. The 3D structure of the head may be a simple geometric model, such as a plane [5], cylinder [11, 2] or ellipsoid [38, 16]; or a mesh that captures detailed facial contours [19, 24]. Geometric models are often easier to work with and enable pose parameters to be computed directly from the feature placement, avoiding an expensive search over the parameter space. This can produce faster and more accurate results [11]. However, neither planes nor cylinders are able to model full range of head orientations and cannot be used with overhead camera angles, such as the HSP video. While ellipsoidal models are a possible solution, HeadLock uses a 3D mesh with a low polygon count to more fully leverage the shape information of the head.

The choice of features is another important distinction between 3D model-based approaches. Several approaches use finely structured features that may represent eyes, eyebrows, and lips [37, 16, 36]. However, these features must be extracted from high resolution images that clearly display the face. In general, images of hair offer features that are far less reliable and less distinct than images of faces. The features used by HeadLock are derived independently

from each pixel and are derived from luminosity and a hair-skin-background segmentation. While I make no rigorous argument that these features are ideal, they can be computed at nearly any resolution, can reliably be extracted from both skin and hair regions, and produced better results than other attempts that included combinations of luminance, raw RGB values, and features derived from Sobel and Difference-of-Gaussians filters.

An earlier work by Wu and Toyama was the only 3D model-based approach I reviewed that attempted to track heads in low resolution video at any angle [38]. This system uses features derived from Gaussian and *Gabor wavelet* [23] convolutions, which can be extracted at very low resolutions (32 by 32 pixels). In the best evaluation scenario, this system produced fairly even results regardless of the yaw of the head, although none of the images or results from the evaluation indicated a significant range in pitch or roll. Furthermore, this system only operates on cropped images of heads and does not determine position or scale parameters. Still, this study suggests that coarse, frequency-based features may provide useful pose cues at very low resolutions; a result that may inform future work.

For lower resolution video, template matching is a more common approach that uses the entire head to perform pose estimation. For these systems, a set of templates must be learned that represent heads at different orientations and, in some cases, of different shapes. Matching a template to a video frame is performed by searching over a chosen parameter space that may include position, scale, the templates, and template deformations. Different systems have chosen different template representations: Support Vector Machines(SVM) [32] are used in [4, 30], Gabor wavelets are used in [27], and neural networks are used in [33, 17], to name a few.

In all of these systems, templates that represent two-dimensional images are used to model heads that are inherently three-dimensional. As a result, a very large set of templates is required to represent the full domain of head orientations for a single head. A large template set requires large amounts of training data and makes matching more difficult by increasing the number of templates must be compared to the image. To make template matching tractable for head pose estimation, the domain of head orientation must be coarsely discretized and, as in 3D model-based approaches, the range is usually restricted.

Subsequently, most template matching systems have only been evaluated on images that clearly display the face of the subject and may not generalize to images of the top or back of the head. The system in [33] was evaluated on video that best approximates the video of HSP; video taken from low-resolution, wide-angle, overhead cameras. However, this system was evaluated on only nine poses over just the yaw parameter. Furthermore, the system relies on multiple camera recordings, which greatly simplifies segmentation and matching.

Instead of using 2D templates, HeadLock uses 3D surface models to represent individual heads. For matching, HeadLock performs a parameter search that is similar to a template matching approach, but uses the 3D model to dynamically generate 2D templates at any orientation and scale. While HeadLock currently requires the manual creation of 3D models for different head types, it avoids many of the problems caused by large template sets and may be better suited to annotation of a limited number of targets. This approach also enables the parameter search to be performed at progressively finer detail to arbitrary precision.

3 Data Collection

3.1 Introduction

The first challenge of HSP was to create a system to record audio and video from a home environment. Many systems for longitudinal audio or video recording exist for surveillance purposes, but none of these systems are designed for behavioral observation in a home environment. The HSP recording system was designed to meet five key requirements: (1) to collect high quality video and audio from dozens of sensors; (2) to maintain synchronization between recordings at greater than word-level precision; (3) to be minimally invasive; (4) to provide sufficient privacy to the participants to ensure their comfort; and (5) to run continuously, with minimal intervention, for three years.

The HSP recording system was created by several members of the Cognitive Machines group at the MIT Media Lab. I constructed the computing grid and created the software for processing and recording sensor data. Deb Roy designed and installed the sensor network. Stefanie Tellex and Soroush Vosoughi created the privacy management interface. Brandon Roy created the system that emails daily status reports and performs additional processing on the recorded data every night. Jethran Guinness constructed the data storage system.

3.2 The Sensor Network

Figure 3.1(a) shows the house from which data is collected. 11 cameras and 14 microphones have been placed in most of the rooms of the house, including the main hallway, entrance, bedrooms, and basement rooms. Figure 3.1(b) shows a camera and microphone installed



(a) The house where data is collected for HSP. 11 cameras and 14 microphones have been installed throughout this house. (b) A camera and microphone embedded in the ceiling of one of the rooms. The microphone is the small, button-like object on the right. A shutter covers the camera when recording is disabled.

Figure 3.1: The HSP recording site and sensors.

in one of the rooms. Cameras are mounted in the center of the ceiling in each room. Only the 2" lens of each camera is visible. Next to each camera is a white, servo-driven shutter that covers the camera when recording is disabled. The microphones are also embedded in the ceiling and have the size and appearance of a silver button.

The video sensors, Lumenera Le165c video cameras, are digital surveillance cameras that transmit data over standard Ethernet. 10 of the 11 cameras are outfitted with Fujinon FE185C057HA-1 lenses, which have a 185° angle-of-view that allow the camera to capture the entire room from the ceiling. The living room has a higher ceiling than the other rooms, and a Fujinon FE185C086HA-1 is used that has a slightly narrower angle-of-view. The cameras are configured to record at 15 frames per second at a resolution of 960 by 960 pixels. A single frame of video is shown in Figure 3.2. Figure 3.3 shows frames taken from 9 of the 11 cameras. As seen in these images, the resolution of the video is sufficient to identify human participants and many of the surrounding objects.

Automatic gain control is enabled on cameras to account for the large disparity in lighting conditions that occur throughout the day. Under low-light conditions, digital gain is used to reduce long exposure times that would result in low frame rates. Digital gain introduces sample noise under certain conditions, which can be seen in the lower-right frame in Figure 3.3. Although the cameras are able to perform automatic white balancing, this feature



Figure 3.2: Single frame of video taken from the kitchen camera while one of the adult residents was feeding a child. The native resolution of this image is 960 by 960 pixels.

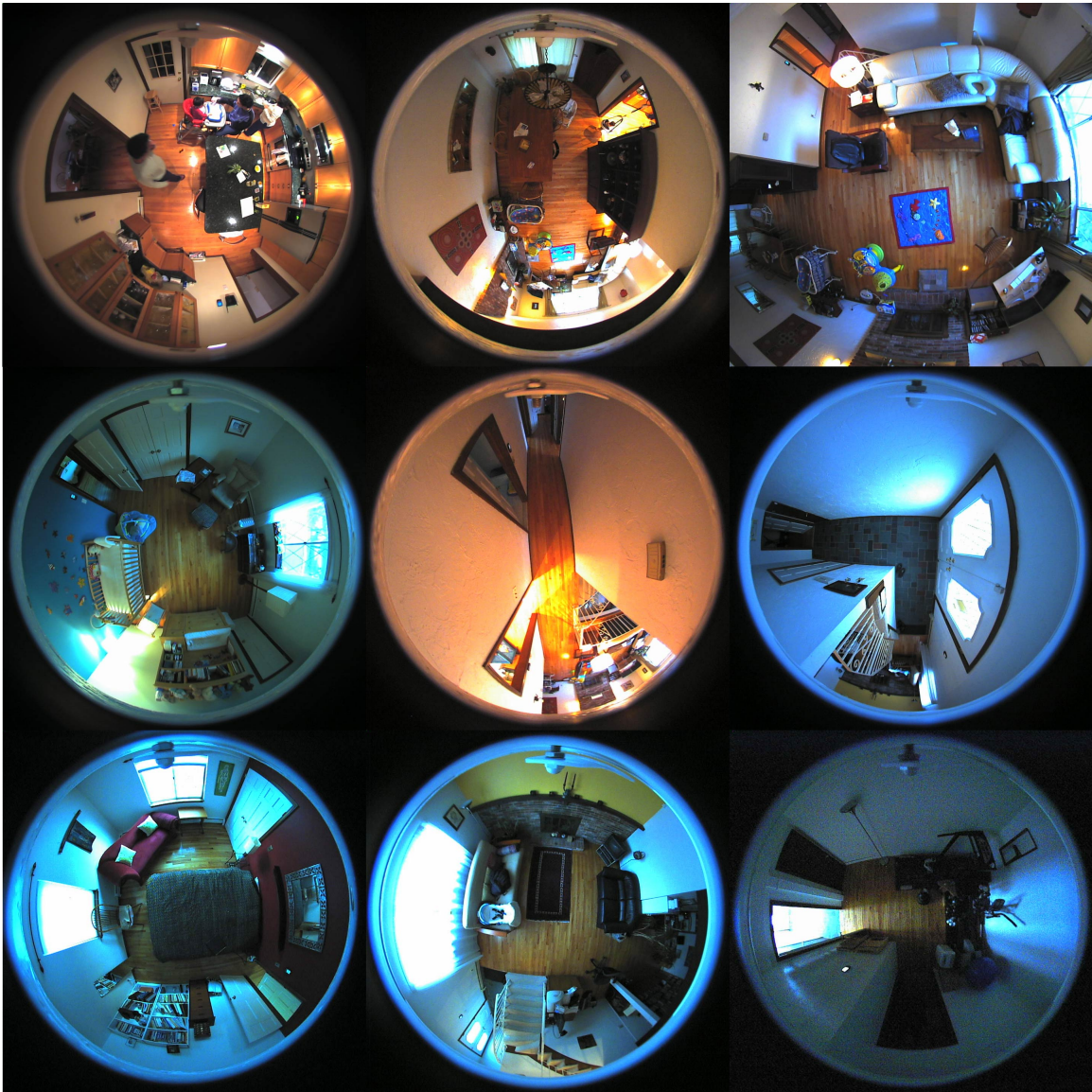


Figure 3.3: Frames taken from 9 of the 11 video cameras. From top-left to bottom-right, the kitchen, dining room, living room, baby bedroom, hallway, entrance, guest bedroom, basement den, exercise room. The living room camera, top-right, has a narrower field-of-view than the other cameras.

has been disabled because it was found to produce significant color fluctuations at relatively high frequencies, greater than 2Hz.

The audio sensors are AKG C562CM *boundary layer* microphones. Boundary layer microphones use the surface in which they are embedded as a pickup. A microphone placed in the center of a room's ceiling is capable of picking up whispered speech in any corner of the room. The analog signals from each microphone are amplified and converted to a 48KHz digital signals using a pair of Presonus *Firepods*.

3.3 Privacy Interface

When recording in a home environment, it is important to consider the comfort and privacy of the residents. To provide control to the participants, a control panel was installed in each room of the house next to the light switch. Each control panel is a small PDA device with a touch screen interface, which is shown in Figure 3.4. The control panels communicate with the recording system through a wireless network provided by three wireless hubs stationed throughout the house. The two largest buttons on the device are on/off toggles for audio and video. Pressing the video toggle causes video recording to be enabled or disabled for one room in the house. When video recording is disabled, a servo-driven shutter physically covers the camera lens to provide a stronger assurance of privacy, as in Figure 3.1(b). While the video sensors share little overlap and may be toggled individually, the sensitivity of the audio sensors makes it possible to hear conversations that occur several rooms away. To ensure the privacy of the audio, the audio toggle button enables or disables audio recording for the entire house.

Despite the control panels mounted in each room, the system inevitably captures some events that are not to be saved. The button on the control panel in Figure 3.4 that displays an exclamation mark, referred to as the *oops button*, enables the participants to delete a portion of recently recorded data. When a user presses this button, the control panel displays an options panel, into which the user may specify the channels of data and duration of time to delete.

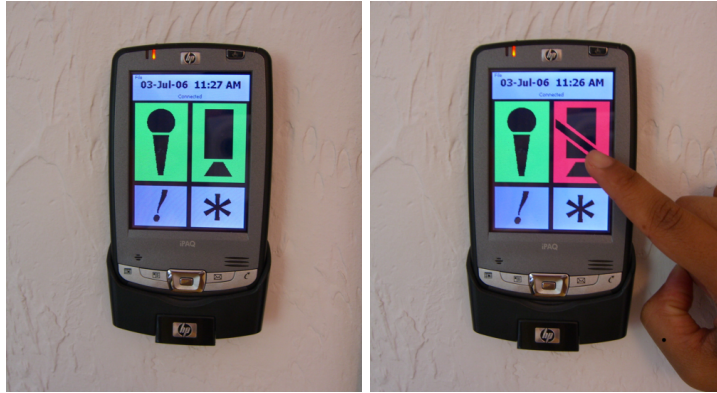


Figure 3.4: PDAs mounted in each room provide a control interface for the recording system. Each PDA has four buttons: (1) the top-left toggles audio recording throughout the house; (2) the top-right toggles video recording for one room; (3) the bottom-left button deletes a portion of recently recorded data to be deleted; and (4) the bottom-right can be pressed to mark events of interest.

Many events occur that are of particular interest to the participants or researchers, such as the child’s first steps or many of the child’s first words. The control panel in Figure 3.4 contains a button that displays an asterisk, the *ooh button*, that may be pressed to mark an event. The ooh events are collected at the end of each day and placed in a relational database that contains metadata for the recordings, as described in Section 3.5. These events may then be easily located and viewed at a later date using the system described in Chapter 4.

3.4 Data Processing and Recording

The audio-video sensors embedded in the house provide a constant stream of data that must be processed and stored. All sensors are connected to a server room in the basement of the house that contains a data processing and recording system, *BlackEye*. The robustness of BlackEye is paramount to avoiding any data loss. As a result, BlackEye performs only tasks that are essential to the recording process. The primary functions include data retrieval, synchronization, compression, formatting, and storage.

Each sensor produces a stream of data, referred to as a *channel*. BlackEye requires five computers to handle all of the data channels. Four of the computers act as video servers,

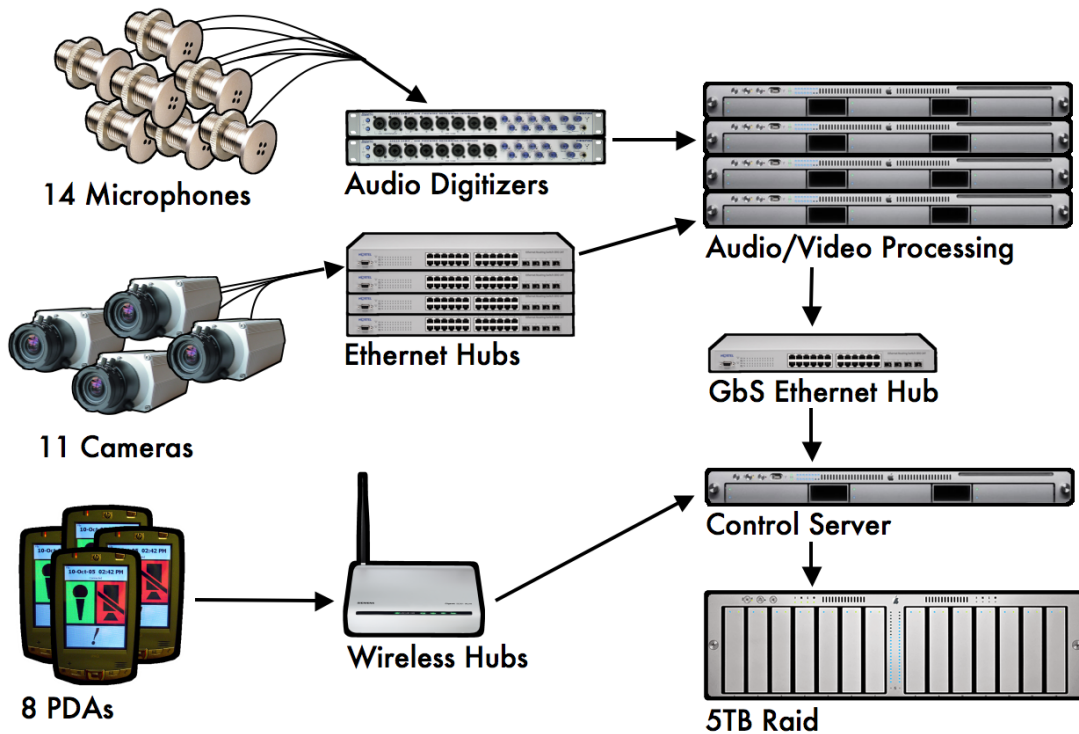


Figure 3.5: Hardware diagram of the data collection system. Arrows indicate the primary flow of data.

which process the video data. One of the video servers also doubles as the audio server, which processes all of the audio data. The fifth computer acts as a control server, which coordinates recording tasks and writes all processed data to a five TB disk array. The hardware configuration for the entire recording system is shown in Figure 3.5.

Maintaining accurate synchronization across data channels poses a significant challenge. The digital cameras transmit data with variable latency and data processing tasks are shared by multiple machines. The accuracy of the system is maintained by synchronizing the internal clocks of all cameras and servers using *Network Time Protocol* (NTP) [28]. BlackEye employs a three-tiered synchronization system, in which cameras synchronize to one of the four video servers, the video servers synchronize to the control server, and the control server synchronized to an external NTP server. The time offset between the video servers and the control server is reliably under 10 milliseconds and usually less than one millisecond. The time offset between the cameras and video servers is not easily determined,

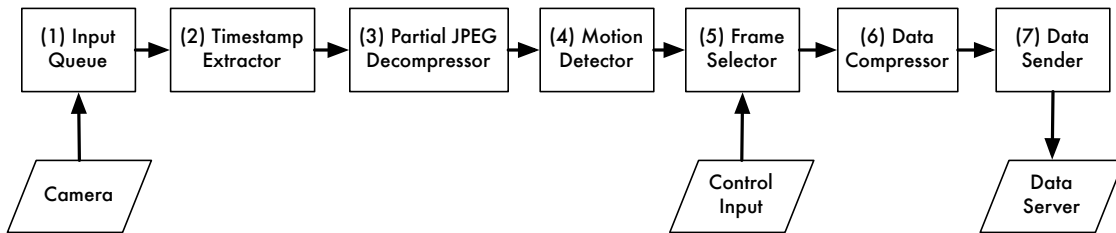


Figure 3.6: System diagram for the video processing software.

but is believed to be of a similar magnitude.

The cameras embed a timestamp into each frame of video. Timestamps are generated at the time of image exposure, thus avoiding the problem of transmission latency. Audio data is delivered in packets by a software library that adjusts audio packet timestamps to account for device latencies. Despite several uncertainties regarding the exact sensor timings, manual observations of the data indicate that the synchronization between audio and video channels is usually accurate to within the limits of human perception.

The recorded data from each channel is stored separately from other channels in a series of files that partition the data at minute boundaries. Each sensor, then, generates one data file every minute. These files are stored in a directory hierarchy that partitions time into years, months, days, hours, and then minutes. Each data file is stored into the corresponding, minute-level directory. To guard against misplaced files, each filename contains a date and time, and timestamps are embedded in each file that indicate the start and stop time of the file to microsecond precision.

In terms of complexity and computational resources, most of the data processing is performed on the video data. Figure 3.6 shows the system diagram for the video processing system that handles each video channel. The purpose of the video processing system is to select the data to record, compress the data, and then to package the selected video frames into files. Data is stored in two formats: one for the full resolution video, and one for a reduced resolution version that is useful when multiple video channels must be transmitted or decoded in realtime. The full video is recorded at 960 by 960 pixels, while the reduced video is recorded at 120 by 120 pixels.

Processing the video data can be described as a seven stage process, which are labeled in Figure 3.6:

(1) Each camera provides a continuous stream of video frames that are independently encoded in JPEG format [18]. Each frame is approximately 100 kilobytes in size. A dedicated software thread continuously pulls video frames from the camera and places them into a queue to be processed by a worker thread.

(2) The timestamp is extracted from each frame.

(3) Each frame must be decoded before the color values can be accessed. This requires the largest portion of computational resources and requires that the video servers decode 16 megabytes of compressed data, representing 165 megapixels, each second. To reduce the computational burden, each frame is only partially decoded to retrieve a version of the image with reduced resolution.

JPEG encoding begins by converting an uncompressed digital image into a *YUV* color space that contains one channel for luminance, and two channels for chrominance. Each channel is then separated into 8x8 blocks of samples, each of which is converted to cosine transform coefficients, which are then quantized, *run length encoded*, *entropy encoded*, and, finally, *Huffman encoded* [18]. When decoding the JPEG images, the video processing system only performs the first four steps of the JPEG decoding process; Huffman decoding, entropy decoding, run length decoding, and dequantization; leaving out the inverse cosine transform, which is generally the most computationally intensive. This is sufficient to access the cosine transform coefficients. The first coefficient of each sample block contains the *DC coefficient* of the block, or the average value of the uncompressed samples. A smaller version of the image is created by extracting just the DC coefficients of each sample block. The resulting image is a one-eighth scaled version of full image and has a resolution of 120 by 120 pixels.

(4) At any given time, most of the rooms of the house are usually empty of people. The video data collected from these rooms is of little interest to the HSP researchers, and removing this video greatly reduces the amount, and cost, of required data storage. To detect and

remove empty frames of video, motion detection is run on each video channel, providing each frame with a flag that indicates if motion is present. If, at a given point in the video stream, there are no frames with a motion flag for ten seconds in either direction, the frame rate of the video is reduced to one frame per second. This greatly reduces storage requirements by removing much of the empty video, but still maintains a minimal frame rate in case the motion detector fails for any reason.

Motion detection is performed by applying a Σ - Δ *filter* [26] to the luminance values of the reduced resolution video frames generated in step (3). The Σ - Δ filter was chosen because it runs quickly and adjusts to changes in lighting conditions and camera noise. The filter operates by maintaining estimates of the average value and standard deviation of each pixel. These estimates are adjusted incrementally by each input frame, such that each average value estimate is adjusted towards the input value, and each standard deviation estimate is adjusted towards the absolute difference between the average value and input value. Estimates are adjusted by a fixed amount that depends on the time between frames, intentionally limiting the speed at which the estimates adapt to changes in the video. Each input value is compared to the average value and, if the absolute difference between them is greater than the standard deviation plus a constant, the corresponding pixel is classified as containing motion. If the number of pixels with motion passes a set threshold, the frame is classified as containing motion. This algorithm is described in Algorithm 1. Parameter optimization was performed by Nikolaos Mavridis.

(5) When a user disables or enables recording using one of the control panels, the control panel sends a message to the control server, which then sends the command to the video server responsible for that processing that video channel. The timestamp of each frame of video is checked and, if the frame occurs when recording is disabled, then the frame is discarded. Because of implementation details, this step is easiest to perform after motion detection. However, a different implementation may choose to remove frames before they are decoded to reduce the computational load when video recording is disabled.

(6) The video frames must be encoded and collated before writing them to disk. This is performed for both the full resolution frames and the low resolution frames that are

```

input : grayscale image,  $P$ , with timestamp
output: true if motion is detected, otherwise false

define :  $p_i$  = input image values,  $p_i \in P$ , normalized to the range from 0 to 1
define :  $\Delta_t$  = the time delta between the current and previous frame, in seconds
define :  $\alpha_i$  = average value estimate for pixel  $i$ 
define :  $\beta_i$  = standard deviation estimate for pixel  $i$ 

if first frame then
  forall  $i$  do
     $\alpha_i \leftarrow p_i$ 
     $\beta_i \leftarrow 0$ 
  end
  return true
else
   $c \leftarrow 0$ 

  forall  $i$  do
     $\Delta_p \leftarrow p_i - \alpha_i$ 
    if  $\Delta_p > 0$  then
       $\alpha_i \leftarrow \alpha_i + 0.059\Delta_t$ 
    else
       $\alpha_i \leftarrow \alpha_i - 0.059\Delta_t$ 
    end

    if  $|\Delta_p| > \beta_i$  then
       $\beta_i \leftarrow \beta_i + 0.00017\Delta_t$ 
    else
       $\beta_i \leftarrow \beta_i - 0.00084\Delta_t$ 
    end

    if  $|\Delta_p| > \beta_i + 0.17$  then
       $c \leftarrow c + 1$ 
    end
  end

  end
  return ( $c \geq 2$ )
end

```

Algorithm 1: The Σ - Δ filter used to detect motion.

generated during step (3). The full resolution video is stored in format called *squint*, and the low resolution in the *wink* format.

The *squint* format begins with a header that contains the time and origin of the video, followed by an index for the video frames. The rest of the format simply stores the sequential series of JPEG images just as they were retrieved from the camera. This is similar to existing MJPEG formats. A *squint* file containing one minute of video at full frame rate is typically 85 megabytes in size.

The *wink* format begins with the same header, followed by a similar frame index. The first frame of the file is compressed using the DEFLATE algorithm [15]. The first frame acts as the keyframe for the rest of the file. For each subsequent frame, the difference between that frame and the keyframe is compressed and stored, which greatly improves the compression ratio that is achieved over compressing the unaltered frames. When decoding a *wink* file, the keyframe must be decoded first, after which any of the other frames may be decoded. A *wink* file containing one minute of video at full frame rate is typically five megabytes in size.

(7) After the video data has been encoded into files, the files are sent to the control server over a 1GbS Ethernet connection to be written to disk.

Processing the audio is a simpler procedure. The raw audio samples are read from the digitizer, written directly into minute-segmented files, and sent to the control server to be written to disk. The audio is stored as a series of 16-bit samples at 48 KHz. Because the audio requires a fraction of the space that is required by video, no attempt was made to compress the audio and the audio samples are stored in a raw format. Typically, a file containing one minute of audio is 5.5 MB.

During the first year of recording, improvements to the recording system were made on a regular basis, causing the BlackEye system to be restarted approximately once a month or less. Currently, BlackEye achieves an average uptime on the order of several months. The most frequent cause of failure are power outages. The system had to be restarted once because of a failed hard drive.

3.5 Post Processing

The BlackEye system is limited to the functions that are essential to realtime recording operations. Several other processing tasks are performed that are not essential to recording, but that provide useful services that enable data management tasks and guard against system failure. These processing tasks are performed by a set of software applications that run once a day at approximately midnight. Five primary tasks are performed each night:

(1) Images are generated for all of the audio and video data that visualize the contents of the data. For audio, *spectrogram* images are generated. For video, *video volume* images are generated. These images are used by a data browsing system and are described in Chapter 4.

(2) All data files generated from the previous day are found and catalogued in a relational database. This process is useful for data management tasks and allows applications to check the existence of data without having to scan the large file structure that contains the data.

(3) BlackEye continuously logs system events and errors into several log files. Each night, the event logs are parsed to collect all ooh events, which are placed into the metadata database. The error logs are parsed to collect information on recording errors that may have occurred.

(4) An email is sent to the HSP researchers that contains a summary of recording activity from the previous day. Figure 3.7 displays a typical example. The summary also contains information on recording errors, the remaining storage space, and the number of ooh events. Checking this report enables the HSP researchers to check that the recording system is functioning properly.

3.6 Data Storage

Under typical usage patterns, audio and video is recorded for eight to ten hours a day, centered around the daylight hours. The child participant is most often on the top floor

RecordingFinder summary for Thu Jul 12 00:00:00 EDT 2007 to Fri Jul 13 00:00:00 EDT 2007

Audio channels

<i>Id</i>	<i>Channel</i>	<i>Segments</i>	<i>Duration</i>	<i>First segment start</i>	<i>Last segment end</i>	<i>Data (GB)</i>
0	Master Bed	2	11:08:00	10:20:00	21:33:59	3.57
1	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	3.57
3	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	3.57
6	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	3.57
2	Baby Bed	2	11:08:00	10:20:00	21:33:59	3.57
4	Guest Bed	2	11:08:00	10:20:00	21:33:59	3.57
5	Bathroom	2	11:08:00	10:20:00	21:33:59	3.57
7	Kitchen	2	11:08:00	10:20:00	21:33:59	3.57
10	Play Room Entrance	2	11:08:00	10:20:00	21:33:59	3.57
13	Play Room Entrance	2	11:08:00	10:20:00	21:33:59	3.57
14	Play Room Entrance	2	11:08:00	10:20:00	21:33:59	3.57
15	Play Room Entrance	2	11:08:00	10:20:00	21:33:59	3.57
17	Play Room Entrance	2	11:08:00	10:20:00	21:33:59	3.57
11	Exercise	2	11:08:00	10:20:00	21:33:59	3.57
12	Exercise	2	11:08:00	10:20:00	21:33:59	3.57
16	Unknown	0	00:00:00***			
						53.6

Video channels

<i>Id</i>	<i>Channel</i>	<i>Segments</i>	<i>Duration</i>	<i>First segment start</i>	<i>Last segment end</i>	<i>Data (GB)</i>
20	Master Bed	0	00:00:00***			
21	Kitchen	3	10:44:00	10:22:00	21:33:59	40.34
22	Baby Bed	2	09:27:00	10:22:00	21:33:59	8.45
23	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	33.61
26	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	31.45
30	Living Dining Hallway	2	11:08:00	10:20:00	21:33:59	28.67
24	Guest Bed	1	02:37:00	18:58:00	21:34:59	4.22
25	Bathroom	0	00:00:00***			
27	Play Room Entrance	0	00:00:00***			
28	Play Room Entrance	0	00:00:00***			
29	Exercise	0	00:00:00***			
						146.74

Successfully parsed 4 out of 4 squint error logs
 Total dropped video frames found: 0

Total data: **200.34 GB**
 Volume: /Network/Servers/argento.local/Volumes/fulci/avdata
 Filesystem: argento.local:/Volumes/fulci/avdata
 Filesystem free space: **626.81 GB**

"Ooh" events: No ooh events today

Figure 3.7: A report is emailed to HSP researchers each night that summarizes the data collected from the previous day. The report also contains informations about recording errors, the amount of remaining storage space, and the number of ooh events.

of the house, specifically in the living room, kitchen, hallway, and baby bedroom. Video recording for these rooms is usually enabled throughout the day. Video recording is usually enabled for other rooms only when the child is present. Many days contain a short period during which the participants disable all or most recording in order to enjoy a period of privacy.

Approximately 200 GB of data is collected each day, causing the 5 TB disk array to be filled every three to four weeks. When the disk array is nearly filled, the data is transferred to a tape library. These tapes are carried to the Media Lab for permanent storage on a 200 TB, RAID 1 disk array, shown in Figure 3.8.



Figure 3.8: Several hundred terrabytes of HSP data is stored on dedicated hard drives, which occupy the four racks in the center.

4 Data Management

HeadLock is an integrated component of an existing application called *TotalRecall*, a data management platform developed for HSP by members of the Cognitive Machines group. TotalRecall includes functionality for audio and video visualization, navigation, playback, and annotation. As a platform, TotalRecall addresses the practical challenges of managing a very large corpus and enables HeadLock to be applied to large scale annotation tasks. TotalRecall was initially developed by Rony Kubat with additional development by Brandon Roy, Jethran Guinness, Stefanie Tellex, Brian Kardon, and myself. This chapter provides a brief overview of only the relevant features as TotalRecall contains many components that are outside the scope of this thesis. For a more detailed description, see [22].

TotalRecall displays image-based visualizations for both audio and video. Audio is displayed using spectrograms, which display the frequency composition of the audio signal. Video is displayed using a *video volume* technique, shown in Figure 4.1, that provides a visual summary of the motion that occurs within the video. Video volumes are generated using an approach described in [14], but adapted to produce rectilinear images that better accommodate continuous video recordings. Sequences of video frames are placed from left

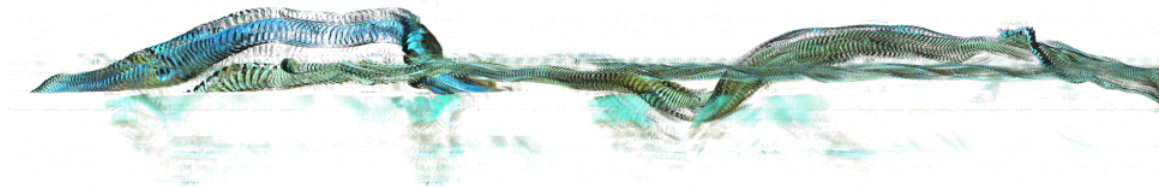


Figure 4.1: Video volume images provide a continuous summary of motion present in the video. Filtered video frames are overlapped from left to right such that people in the video form continuous streaks that move across the image. This image shows two people entering a room. One person remains in one place, while the other moves around.

to right, with each frame overlapping most of the previous frame. Pixels that do not change between frames are then made transparent such that people moving through the space are revealed as streaks of color. Figure 4.1 depicts two people entering a room and moving for approximately a minute.

Spectrograms and video volumes are generated at multiple time scales for all HSP video recordings, as described in Section 3.5. These images are then used by the TotalRecall navigator, shown in figures 4.2 to 4.4. The navigator presents a timeline view of multi-channel data in which each horizontal band displays a single data stream. Each channel, unless hidden, displays a continuous visualization image of the data. The user may view data at any time scale from several seconds to several months. Figure 4.5 shows the video player, which can display one stream of video at full resolution and up to ten streams of video at reduced resolution. The video player contains a limited navigation interface and is primarily controlled by the navigator.

In addition to data visualization and browsing, TotalRecall enables annotations to be created and viewed, as shown in Figure 4.2. Existing annotations include information on sound classifications, speaker identities, and speech transcriptions. Annotations are stored in a relational database, which allows metadata to be flexibly shared by other software systems in addition to TotalRecall.

One application of TotalRecall has been to indicate the room location of the child throughout the data and to indicate if the child is awake or asleep. By our informal tests, the use of video volumes allows a TotalRecall user to track and annotate the child through a day of data (approximately 72 hours of video) in two hours. In future work, a similar visualization might be developed to display tracking, identification and head pose estimates.

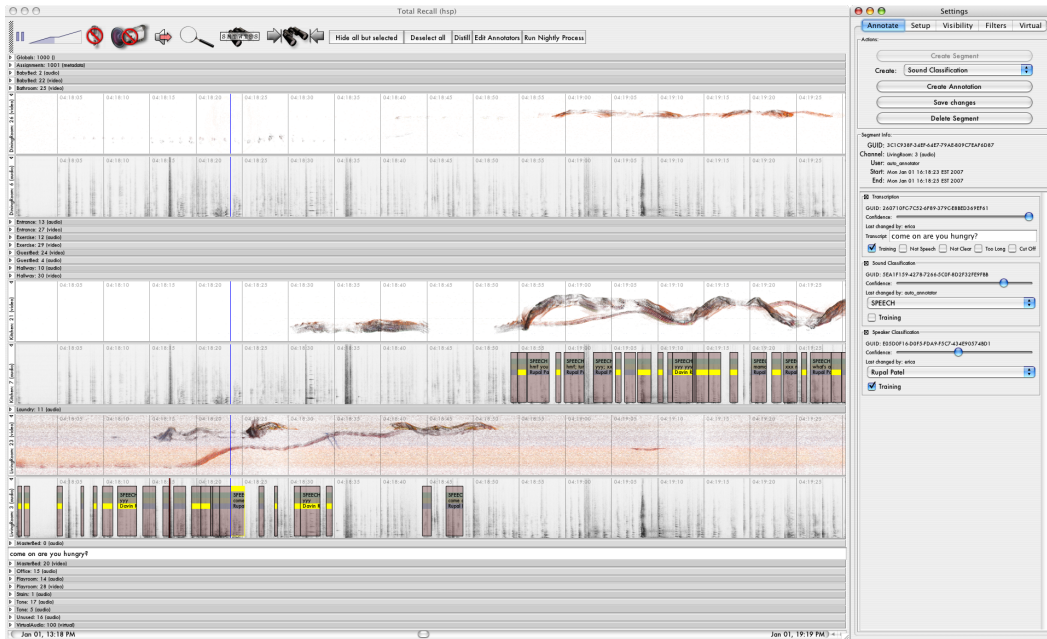


Figure 4.2: TotalRecall displaying approximately one minute of data (left). The rectangles appearing in two of the channels indicate annotations that, for this image, contain information on sound classifications, speaker identities, and speech transcriptions. The option panel (right) displays more detailed information on selected annotations.

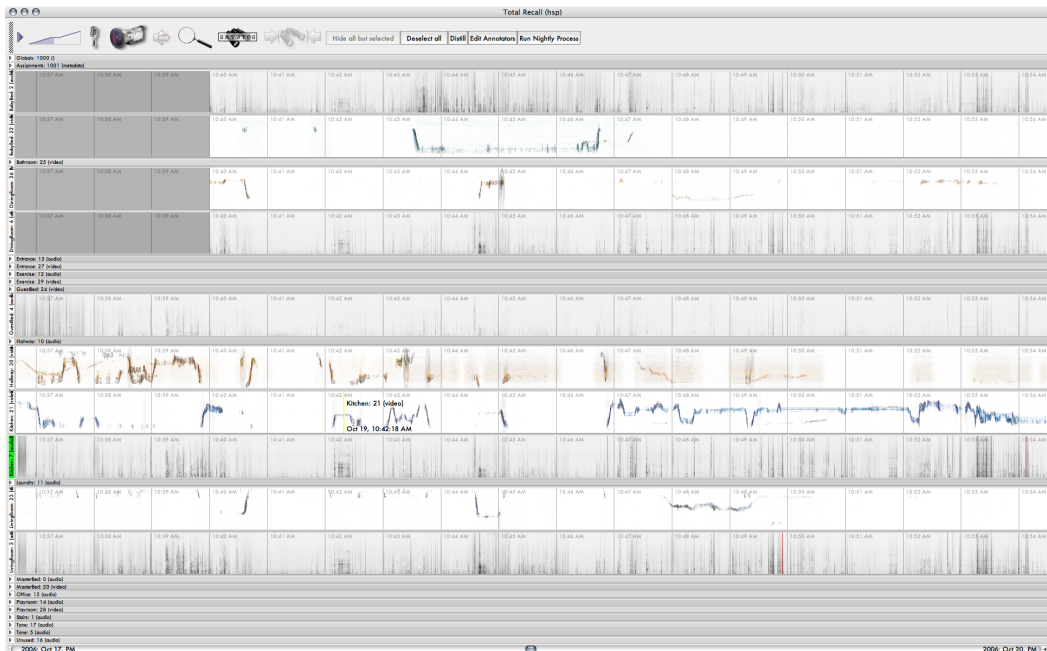


Figure 4.3: The navigator window of TotalRecall. Each horizontal strip represents a channel of data through time. This window shows approximately 20 minutes of data from five audio channels and five video channels. Portions of channels that are grayed out indicate that no data was recorded for that time.

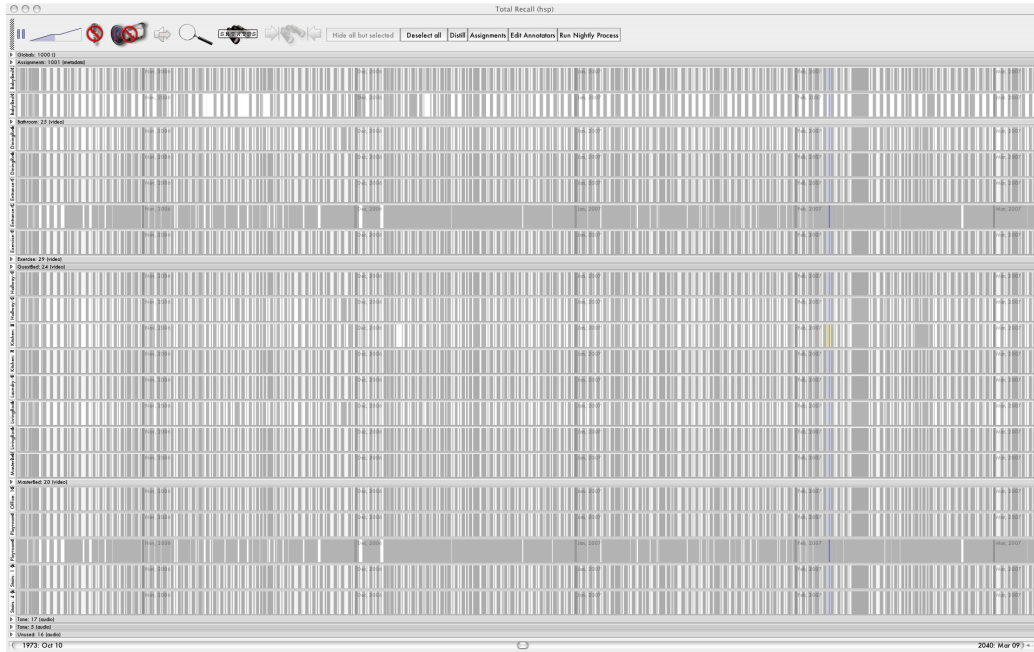


Figure 4.4: TotalRecall displaying six months of data. The vertical strips of white and gray indicate patterns of night and day. The noticeably large, vertical, gray strip near the right indicates a period in February when the participants were on vacation. The two horizontal strips that are mostly gray show that video recording is usually disabled for the bathroom and master bedroom.



Figure 4.5: The TotalRecall video player displays ten channels of reduced resolution video and one channel of full resolution video. Video annotations may be entered directly onto the video (see 5).

5 Head Pose Annotation

HeadLock approaches head pose annotation as a semi-automatic tracking process. While manual annotation of head pose is labor intensive, automatic approaches are prone to error and cannot be used alone to accurately extract pose information. However, the amount of human effort required for annotation tasks can be drastically reduced if head pose can be accurately tracked for even short segments of video.

In HeadLock, an annotation task begins with a *seed* annotation that is entered manually. A head tracker then propagates the annotation to surrounding frames. While HeadLock is a work in progress and the interface has not been fully implemented, it is intended that the human annotator detects when the head tracker makes an error, which can then be manually corrected to continue the annotation process. This chapter describes both the annotation interface and head tracker that comprise HeadLock.

For this research, head pose is defined by six parameters: x , y , σ , roll, pitch, and yaw. x and y represent the center of the head as it occurs within the video frame. σ represents the scale, or size of the head, and is directly related to the z position of the head. Roll, pitch and yaw form a set of Euler angles representing the orientation of the head. Each pose annotation contains these six parameters in addition to the identity of the head.

5.1 Annotation Interface

The annotation interface, shown in Figure 5.1, is built into the TotalRecall video player. By selecting a “Head Annotation” option at the bottom of the video player, the user is able to make annotations by directly marking any frame of video. To make an annotation, the

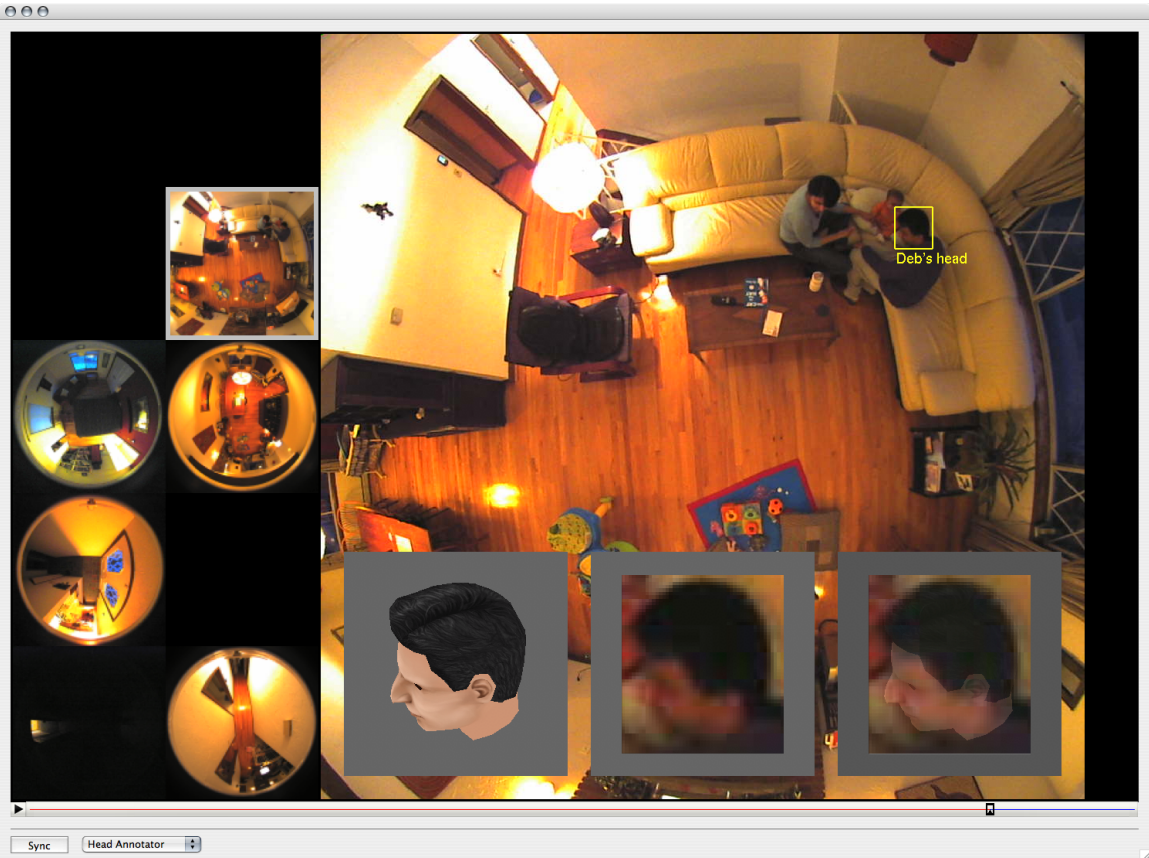


Figure 5.1: The head annotation interface. After the user has drawn a bounding box around a head, three insets appear that display a CG head, a zoomed view of the image head, and the CG head superimposed on the image head. The user annotates the head pose by adjusting the CG head until it matches the image head.

user first draws a rough bounding box around the head of a person. The person then selects the identity of the person from a popup menu. On top of the video frame, three images appear. The middle image provides a magnified view of the head to provide the user with a reference. The left image displays a textured, 3D model of a head. The system has access to several computer graphics (CG) heads. For each member of the house, a head model was generated manually by scaling a single, prototype head to roughly match the shape of the house member's head. The heads for each of the principal subjects are shown in Figure 5.2. Based on the identity of the annotated person, as indicated by the user, the system selects the CG head that best matches that real head. The annotator then adjusts the position, scale, and orientation of the CG head until it matches the video image. To help the user validate the alignment, the right image displays the magnified view of the video with the CG



Figure 5.2: The four principal participants and associated 3D surface model. The 3D models were all generated by scaling the same original model until it roughly matched the participant’s head, resulting in only slight differences between the models.

head superimposed. After the annotator completes the alignment process, the annotation is stored in a relational database.

Although the user must provide six parameters in order to properly match the CG head to the real head, a standard mouse has only two DOF and thus restricts the user to manipulating only two parameters at a time. To expedite annotation, the system utilizes a 6-DOF input device that can manipulate all six parameters at once. Although a number of suitable devices exist, the head annotation system currently uses an inexpensive 3DConnexion, *SpaceNavigator* device. This device consists of a black rubber puck that is flexibly mounted on a weighted metal base. The puck can be moved approximately one centimeter in any direction relative to its base, and can be rotate about any axis by approximately 10° . By pushing and twisting the puck, the user can position, scale and rotate the CG head in an intuitive manner.

5.2 Head Tracking Overview

The HSP video poses several difficult challenges: (1) head images have low resolution; (2) backgrounds may be cluttered; (3) lighting conditions are uncontrolled; (4) many images contain regions with low contrast between foreground and background. However, the HSP video also provides several mitigating conveniences that are leverage by the HeadLock head



Figure 5.3: The 6-DOF input device used to enter the six head pose parameters.

tracker: (1) there are a limited number of people within the video, all of whom have high skin-hair contrast; (2) the camera is stationary and the video is unedited; (3) bodies are generally visible, providing useful context for head tracking and detection; (4) head estimation is performed on video rather than single images, which provides a strong prior for each frame and allows temporal filtering.

Summarily, head pose is estimated with a 3D model that is densely populated with features derived from luminosity and a skin-hair-background segmentation. For a given image, the system searches the parameter space by rendering the 3D model in different orientations and determines the similarity of the features on the rendered model to the features extracted from the video image. This section will describe this system in detail.

Figure 5.4 shows the system diagram for the head pose estimation system in terms of information flow. The system can be broken into three main components: a position tracker, a feature extractor, and an orientation estimator. The main components are further divided into a series of subcomponents, which are labeled on the image from *A1* to *C3*. Head tracking begins with a single initialization step, in which a seed annotation and corresponding video frame are used to train the position tracker (*A2*) and the skin-hair segmenter (*B1*). After initialization, head pose is estimated for each subsequent frame according to a linear sequence of operations:

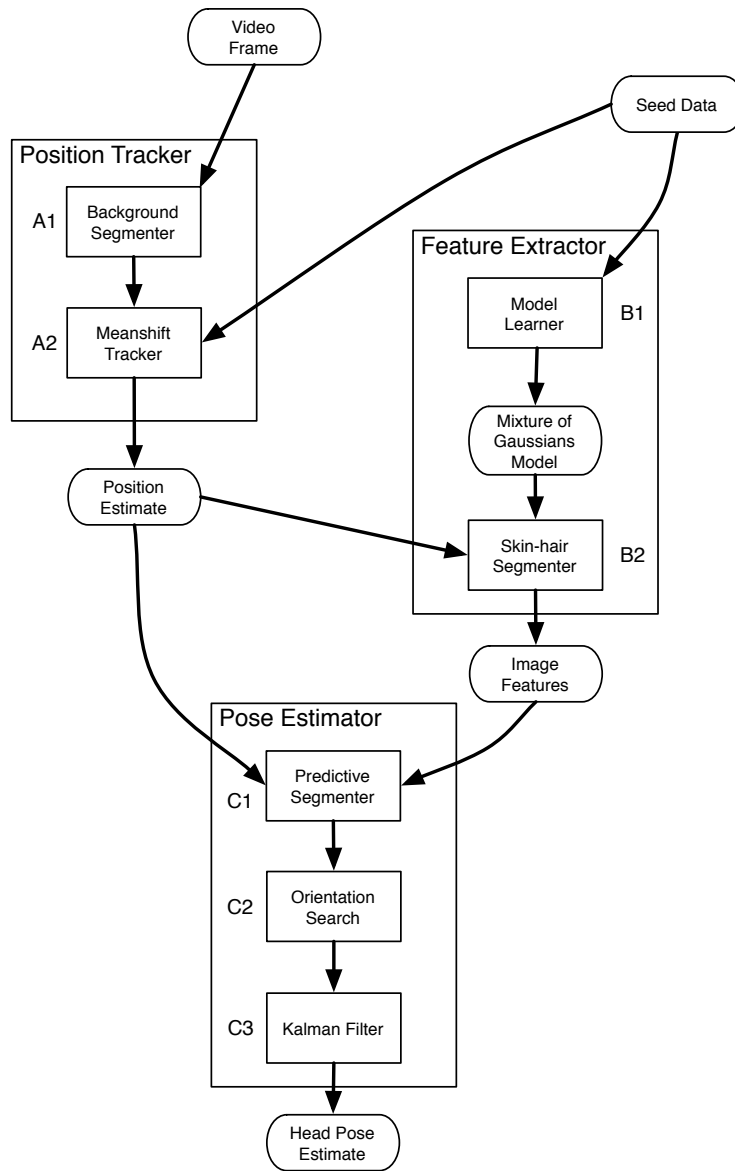


Figure 5.4: Diagram of the head pose tracking system. Rounded boxes represent data while square boxes indicate procedures. Arrows indicate the flow of information.

- A1.* A foreground-background segmentation is performed on the image.
- A2.* The position is estimated using a meanshift tracker that has been modified to utilize foreground information.
- B2.* Each pixel is converted to a feature vector containing skin-hair-background segmentation features.
- C1.* A predictive segmentation step is performed that utilizes the previous pose estimate to adjust the image features.
- C2.* A search is performed over the 6-dimensional pose space to match a 3D head model to the image features.
- C3.* The pose estimate is filtered with a Kalman filter.

For the first part of the head estimation task, we will assume that the system is first given an accurately annotated head as it occurs in a single frame of video, and that the system only provides annotations for subsequent frames. This will allow us to bypass the initial problems of head localization and alleviate the problem of target modeling. As stated in the previous section, the system defines head pose using six parameters, x , y , σ , yaw, pitch and roll. The task for the automated head pose estimator is to determine these parameters for a persisting head in a segment of video.

5.3 Position Tracker

The position tracker provides an initial estimate of head location. Although the pose estimator also searches over location, the position tracker is able to perform this task more robustly and efficiently. The position tracker builds on the mean-shift tracker described in [12]. Mean-shift tracking is performed by adjusting the location of a bounding box to maintain a consistent distribution of color features within the box. The HeadLock position tracker modifies the standard mean-shift tracking algorithm to incorporate temporal features derived from a foreground-background segmentation. The position tracker is divided into two subcomponents: a foreground-background segmenter, *A1*, and the mean-shift tracker, *A2*.

A1. The foreground-background segmenter uses a very simple keyframe approach that leverages the continuity of the HSP video as well as the motion flags that are embedded in the video during recording (see Section 3.4). Upon initialization, the segmenter attempts to locate a keyframe that does not contain any human subjects; essentially, a frame that just represents the background. Starting at the seed frame, the segmenter searches backwards through the video stream and examines the motion flags embedded in the video until a frame is found that does not contain motion and that is not within ten seconds of a frame that does contain motion. After initialization, for each frame given to the segmenter, the color vector of each pixel, ρ_i , is compared to the corresponding color vector from the keyframe. If the Euclidean distance between the color vectors is greater than a set threshold, the pixel is given a *motion weight*, m_i , of 3. Otherwise, the pixel is given a motion weight of 1. These weights were chosen empirically by observing typical segmentation results on the HSP corpus. While any number of approaches to segmentation could have been taken, this approach was selected largely because it offered a very simple implementation.

A2. Upon initialization, the tracker is trained using the seed video frame and seed annotation. Using the seed frame, the tracker models the target as a histogram of quantized color values, A . The chosen quantization function, q , allocates four bits to each RGB color sample:

$$q(\rho_i) = \left[\begin{array}{ccc} \lfloor 16\rho_i^r \rfloor & \lfloor 16\rho_i^g \rfloor & \lfloor 16\rho_i^b \rfloor \end{array} \right]$$

where ρ_i^r , ρ_i^g , and ρ_i^b represent the red, green, and blue color components, respectively, and occur exclusively between 0 and 1.

The seed annotation, which contains the location and scale of the target, is used to infer a bounding box around the head image. Each element in this bounding box is quantized, weighted according to its Gaussian distance to the center of the box (Gaussian distance is derived by applying a Gaussian function to the Euclidean distance), and added to the histogram. The resulting histogram, A , contains the approximate color distribution of the target.

For each frame on which tracking is performed, the tracker attempts to adjust the bounding box so that it best contains the target. First, a histogram of background color values, B , is computed on the pixels in a box that is concentric to the bounding box and that has twice the area. The importance of a given color value to locating the target is proportional to that value's frequency near the target, and inversely proportional to that value's frequency in the larger region containing the target and surrounding area. To this effect, the target histogram, A , is divided elementwise by the background histogram, B , to produce a table of *value weights* for each color value, $V \leftarrow \frac{A}{B}$. This process of computing value weights is the standard approach used by mean-shift trackers.

After the value weights and motion weights are computed, the weight of each pixel in the bounding box may be computed as $w_i = V_{q(\rho_i)}m_i$, where w_i is the pixel weight, V_i is the value weight, q is the quantization function, and m_i is the motion weight. Figure 5.5 displays the an example of the pixel weights generated by the tracker. Although this figure displays an entire body as the tracking target, HeadLock only uses the tracker on the head region.

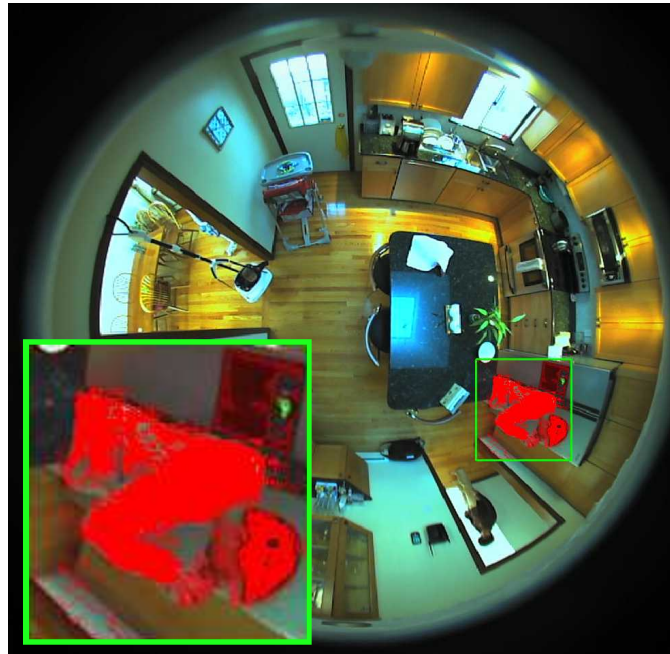


Figure 5.5: Pixel weights, indicated in red, are computed for each pixel in the bounding box based on color value frequencies and segmentation information.

The tracker then computes the centroid of the pixel weights and centers the bounding box

on that location. In typical applications, the pixel weights are also used to compute the dimensions of the bounding box. In HeadLock, the scale of the target is estimated at a later stage.

5.4 Feature Extractor

Each pixel of the head image is considered a feature vector. When the video frame is first read, each pixel contains three color values as features. However, color values are very sensitive to lighting and were found to produce poor results when used for head pose estimation in the HeadLock framework. Many types of features have been used for head estimation tasks, as discussed in Section 2.3. The features used by this system include luminance and hair-skin-background segmentation values. The segmentation values are computed with a Mixture-of-Gaussians(MoG) model [3]. While these features might not be optimal, they can be computed individually for each pixel, providing excellent invariance to the resolution of the image. The feature extractor is divided into two subcomponents: (B1) a model learner that creates a MoG model at initialization; (B2) a segmenter that uses the learned MoG to convert the color vector of each pixel into a vector in the chosen feature space.

B1. The model learner generates a MoG using the seed annotation and seed video frame. First, the position estimate provided by the position tracker is used to extract the rectangular portion of the video frame that contains the head to an image, Ψ_v , such that the dimensions of the image are 30% larger than a tight bounding box around the estimated head. A pixel from row i and column j is expressed as $\rho_{ij} \in \Psi_v$, and the color vector is expressed as $\rho_{ij} = [\rho_{ij}^r \ \rho_{ij}^g \ \rho_{ij}^b]^T$. Based on the person identity label provided by the seed annotation, a 3D model is selected that matches the target head. The 3D model is rendered into a second image, Ψ_{cg} , such that the dimensions of Ψ_{cg} match those of Ψ_v . The 3D model is rendered using the pose parameters provided by the seed annotation, which results in a close alignment between the heads represented in Ψ_{cg} and Ψ_v . Each polygon in the 3D model holds a *hair* or *skin* label, providing a convenient method for extracting a set

of labels from Ψ_{cg} . Pixels drawn by *hair* polygons are given *hair* labels, and the same for *skin* pixels. Pixels that were not produced by any polygon are given the *background* label. Because the two images are aligned, the set of labels may be applied to both. Thus, a set of pixels, Ψ_v , and a corresponding set of labels, L , are obtained. An example of this labeling process is shown in Figure 5.6.



Figure 5.6: The images and pixel sets used for training and applying the feature extractor. From left to right: Ψ_v , Ψ_{cg} , the set of *hair* pixels, the set of *skin* pixels, the set of *background* pixels, and $\hat{\Psi}_v$, which displays the set of features on which matching is performed.

Using the labeled image, the system trains three MoG models, one for each class label, and then combines the three models into a unified model. For each class, $j \in [hair, skin, background]$, an EM algorithm [3] is employed to fit a set of 3-dimensional Gaussians on the set of pixels with that class label, $\rho_n \in \Psi_v, \ell_n \in L = j$. The MoG model for *hair* and *skin* each contain two Gaussians, and the MoG model for *background* contains five. The EM algorithm provides both the parameters, θ_k^j , and class probabilities, p_k^j , for each Gaussian. Independently, each MoG produces a likelihood value that pixel belongs to a given class:

$$f^j(\rho_i; \theta^j) = \sum_{k=1}^{n_j} p_k^j N(\rho_i; \theta_k^j)$$

where $j \in [hair, skin, back]$ represents the class label, ρ_i represents an input pixel, and k iterates over each Gaussian contained in the MoG model.

B2. After initialization, each frame is given to the segmenter to transform the color vector of each pixel, $\rho_i \in \Psi_v$, to a 4-dimensional feature vector, $\tilde{\rho}_i \in \tilde{\Psi}_v$. The first three values are generated using the three MoG models, and the fourth value, $\tilde{\rho}_i^y$, represents the luminance of the color vector. A perceptually equivalent luminance value is computed using a linear operation that is standard for *sRGB* color values [39]:

$$\tilde{\rho}_i^y = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \end{bmatrix} \begin{bmatrix} \rho_i^r \\ \rho_i^g \\ \rho_i^b \end{bmatrix}$$

Combined with the segmentation values, the complete feature vector generated from each pixel may be expressed:

$$\tilde{\rho}_i = \begin{bmatrix} \frac{f^{hair}(\rho_i; \theta^{hair})}{w} & \frac{f^{skin}(\rho_i; \theta^{skin})}{w} & \frac{f^{back}(\rho_i; \theta^{back})}{w} & \tilde{\rho}_i^y \end{bmatrix}$$

$$w = \sum_j f^j(\rho_i; \theta^j)$$

The w term is used so that the first three values form a valid distribution over class probabilities. Figure 5.6 shows an example of a transformed image.

5.5 Pose Estimator

After the position of the head has been estimated by the tracker, and the head image, Ψ_v , has been extracted from the video frame and converted into the chosen feature space, $\tilde{\Psi}_v$, the system can perform a search over the pose parameters. The system uses the estimated parameters from the previous frame, ϕ_{n-1} , as a starting point to estimate the parameters of the current frame, ϕ_n . In the case of the first frame to be estimated, the seed annotation provided by the user is used. The pose estimator is divided into three subcomponents: (1) a predictive segmenter that removes features around the edges of the image that might be interface with the matching process; (2) an orientation search that attempts to find the pose parameters that optimize the match between the 3D head model and the head image; (3) a Kalman filter to reduce the noise in the pose estimates.

C1. While the skin-hair segmenter(*B2*) produces a background probability for each pixel, the segmenter relies only on color information and may not reliably distinguish between background and foreground when the two share similar colors. A second segmentation

step is performed to further identify background pixels. In this second segmentation, the CG model is rendered at the same resolution as the head image, $\tilde{\Psi}_v$, using the orientation and scale provided by the previous pose estimate, ϕ_{n-1} . The background layer of this rendered image is isolated to provide an alpha mask, M . A Gaussian blur is applied to M to soften the edges. Each element is then multiplied by 3 and clamped to the range $[0 : 1]$, effectively inflating the edges of the mask and created an expanded silhouette of the previous head. Each pixel feature, $\tilde{\rho}_i$, now has a corresponding segmentation value, $m_i \in M$, and a background probability, $\tilde{\rho}_i^{back}$. Treating both values as alpha masks, each pixel feature is premultiplied by both m_i and $\tilde{\rho}_i^{back}$, after which the $\tilde{\rho}_i^{back}$ values are discarded:

$$\hat{\rho}_i \in \hat{\Psi}_v = \left[\begin{array}{ccc} m_i \tilde{\rho}_i^{back} \tilde{\rho}_i^{hair} & m_i \tilde{\rho}_i^{back} \tilde{\rho}_i^{skin} & m_i \tilde{\rho}_i^{back} \tilde{\rho}_i^y \end{array} \right]$$

C2. The basic approach to estimating orientation is to render the CG head with different orientation parameters and select the best match. The primary implementation bottleneck is the speed at which the CG head can be rendered. Rendering is performed using the OpenGL graphics library, providing hardware acceleration to the rendering process. Furthermore, the texture of the head is altered so all skin polygons are rendered into the first color channel, hair polygons are rendered into the second channel, and the intensity of the original texture is rendered into the third. By using this alternate texture, the head is rendered directly into the feature space of $\hat{\Psi}_v$.

In order to compare the head image, $\hat{\Psi}_v$, to the rendered head, $\hat{\Psi}_{cg}$, cross correlation is used as the cost function:

$$C(\hat{\Psi}_v, \hat{\Psi}_{cg}) = 1 - \frac{\sum_{y=1}^h \sum_{x=1}^w \hat{\Psi}_v(y, x)^T \hat{\Psi}_{cg}(y, x)}{\sqrt{\sum_{y=1}^h \sum_{x=1}^w \hat{\Psi}_v(y, x)^T \hat{\Psi}_v(y, x)} \sqrt{\sum_{y=1}^h \sum_{x=1}^w \hat{\Psi}_{cg}(y, x)^T \hat{\Psi}_{cg}(y, x)}}$$

The cross correlation is negated for convenience to produce a cost between the values of 0 and 1. Using this cost function, a hill climbing search is performed over all six parameters, $\phi_n = [x, y, \sigma, roll, pitch, yaw]$ to minimize the cost. The search starts using the parameters

from the previous frame, $\phi_{n-1} \rightarrow \phi_n$, and uses a *simulated annealing* approach to determine the amount by which each parameter is changed [21]. At the beginning of the search, the entropy value, e , is set to 1.0. Parameters are changed one at a time in orthogonal steps. The location and rotation parameters are changed additively, while the scale parameter is changed multiplicatively. When no parameter can be changed by the current step size without increasing the cost, the entropy value is reduced by 10% and the process is repeated. When the entropy value reaches 0.05, the search is stopped and the final parameter values are taken as the estimate.

C3. To reduce noise in the pose parameter estimates, parameter estimates are smoothed with a Kalman filter [20]. Filtering is performed on both pose parameters and their derivatives, $[x, y, \sigma, roll, pitch, yaw, \dot{x}, \dot{y}, \dot{\sigma}, \dot{roll}, \dot{pitch}, \dot{yaw}]$.

After the filtering step, the resulting pose is used as the final pose estimate for that video frame. Although the Kalman filter might also be used to predict the head pose of the next frame, this has not yet been implemented in the current system. Instead, each pose estimate is used directly as the initial guess for the pose of the next video frame. For each frame of video that follows, the tracking process is repeated; the target position is updated, features are extracted from the video, and then the target pose is updated and filtered. Although the tracking process is performed contiguously in a forward direction, it is hoped that future modifications will allow tracking to be performed in a more complex manner that enables greater interactivity with a human operator, as described in Chapter 7.

6 Evaluation

In this section, the performance of HeadLock is evaluated and discussed. The primary purpose of the evaluation is to determine how HeadLock may be used to increase the efficiency of head pose annotation tasks. Head annotation is envisioned as a semi-automated process in which a human operator provides an initial annotation that contains the identity, position, size, and orientation of the head. HeadLock then propagates this annotation into the subsequent video frames. When HeadLock fails to estimate the head pose in a frame of video, the operator adds a corrective annotation and the process continues. In this scenario, the average time-to-failure, or the average length of video that HeadLock can annotate without error, may be viewed as the frequency at which the human operator must provide annotations. In addition to mean-time-to-failure, this evaluation also examines how the quality of video affects performance and how HeadLock might further help a human operator find and correct errors.

In order to determine the mean-time-to-failure, each pose estimate is given a binary label of *correct* or *incorrect*. Correctness is defined by two error thresholds, one for position error and one for gaze direction error. The position error is computed as the euclidean distance between the human-annotated and system-estimated positions, measured from the center of the head. Position error is reported in units of pixels. The gaze direction error is computed as the angular distance, in degrees, between the human-annotated and system-estimated gaze directions. A pose estimate is labeled incorrect if the position error exceeds six pixels or the gaze direction error exceeds 30° .

Figure 6.1 illustrates a gaze direction error of approximately 30° . The error thresholds were chosen prior to the evaluation and represent a level of precision I believed is possible given the resolution of the video. Using this measure, the probability of randomly selecting a

correct gaze direction is 0.067.



Figure 6.1: The center image shows a rendered head that accurately portrays the pose of the head on the left. The right image shows a rendered head with a gaze direction that is off by 30° .

Gaze direction depends on only two of the three orientation parameters: yaw and pitch. The roll of the head represents rotation about the axis passing from the nose to the back of the head, making it independent of gaze direction. Although the roll parameter is omitted, this measure was chosen because it provides a more intuitive error that can be expressed in degrees. Informally, the system rarely produced a correct estimate of gaze direction without accurately estimating all orientation parameters.

The evaluation was performed on a test set of 71 video segments. Each video segment begins on a frame containing at least one human. A computer program was used to randomly select video segments from the HSP corpus. Video segments were selected with uniform probability from a subset of the HSP video recordings that spanned eight days, three different rooms, and contained a child participant¹. The resulting set includes the child, his parents, and a third daytime caregiver that appear regularly from the HSP data of that period. The set contains a representative sampling of poses, lighting conditions, and occlusions. One clip was removed from testing because it contained no heads in the initial frame.

The test set was selected randomly in order to demonstrate the performance of HeadLock on the entire range of HSP data and to determine as fully as possible the set of conditions under which HeadLock may enhance head annotation tasks. As a result, the test set poses a number of difficult problems. One of the primary challenges is the relatively low resolution of the heads. The average head in test set is 42×42 pixels as measured by a tight bounding

¹An eight day period of video was manually annotated to indicate the presence of the child at all times, producing a set of video segments that all contained the child. Recordings originating from all but three of the rooms were then discarded. The test set was chosen randomly from the remaining video recordings.

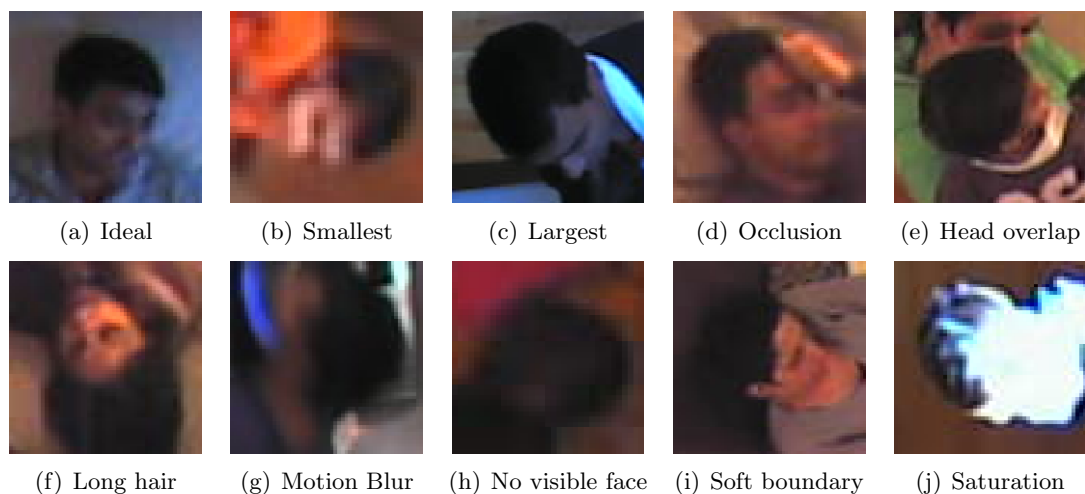


Figure 6.2: Example heads from the test set.

box. The largest head was 71×70 pixels and the smallest head was 24×22 , as shown in Figure 6.2. Figure 6.2 also illustrates several other challenges: heads are sometimes partially occluded and may overlap with other heads, participants may have long hair that does not hold a definite shape, many of the heads do not display any portion of the face or skin, many of the backgrounds are cluttered or contain color values that are identical to the foreground colors, and some of the video contains motion blur and color saturation.

The evaluation was conducted by two human *operators*, myself and an undergraduate assistant. Each video segment in the test provided one *trial*. For each trial, a human operator provided a seed annotation for the chosen target head using the annotation interface described in the *Head Annotation* section. HeadLock was then run on the video segment and generated pose estimates for each subsequent frame. In order to locate the first incorrect pose estimate, the operator observed the estimates for each video frame. When an estimate appeared incorrect or nearly incorrect, the operator verified the estimate by manually annotating and same video frame. When the operator completed such an annotation, the interface displayed the difference between the operator’s pose and the estimated pose and indicated whether the estimated pose was correct. In most trials, after the operator discovered an incorrect pose estimate, the operator determined whether it was the *earliest* error by annotating previous frames until a correct estimate was also found. The trial was terminated when the first error was located.



Figure 6.3: Test Set with Time To Failure

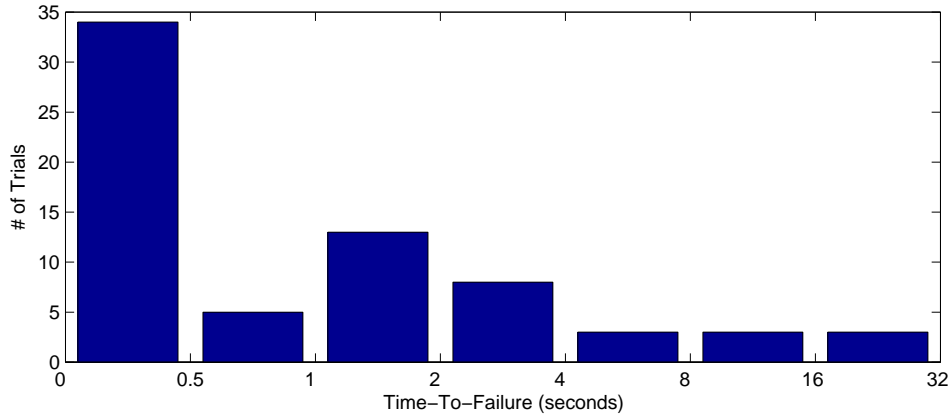


Figure 6.4: Time-to-failure distribution shown in logarithmic space.

For each trial, the time-to-failure was computed as the difference in timestamps between the seed frame and the first misestimated frame. The frame rate of the video varied between 14 and 17 frames per second, where low light conditions required a longer exposure time and produced in lower frame rates than brighter light conditions.

The results of each trial are shown in Figure 6.3, and the distribution of time-to-failures is shown in Figure 6.4. The average time-to-failure was 2.47 seconds. Three of the trials ended while the target head was occluded. Approximately half of the trials failed immediately, indicating that many of the video segments contained initial conditions that HeadLock was unable to accommodate.

The mean time-to-failure omits one of the 70 trials. Trial 1 contained a target participant that was asleep for approximately nine minutes. As a result, the time-to-failure for this trial was more than 20 times longer than the next longest trial and skewed the average from 2.47 to 10.1 seconds. While trial 1 is omitted from the average, it showed that the system was able to achieve fair stability when contending with only camera noise and slight motion.

6.1 Example Results

Figure 6.5 displays several estimates from trial 15. The top row shows the target head as it appears in the raw video. The position estimates were used to extract these heads, so

the accuracy of the position estimates may be judged by how well the head is centered. The bottom row shows the orientation and scale estimates by rendering a 3D model with those parameters. The first column shows the seed frame, which was annotated by a human operator rather than estimated. The last column shows the first error, which occurred 2.79 seconds into the video segment.



Figure 6.5: Selected pose estimates from Trial 15

In this sequence, the target head is successfully tracked across a room with a cluttered background. Although the pose of the target head changes significantly, pose estimates fall well within the 30° limit. As shown in the final column, several accurate estimates were provided even when only the back of the head is visible.

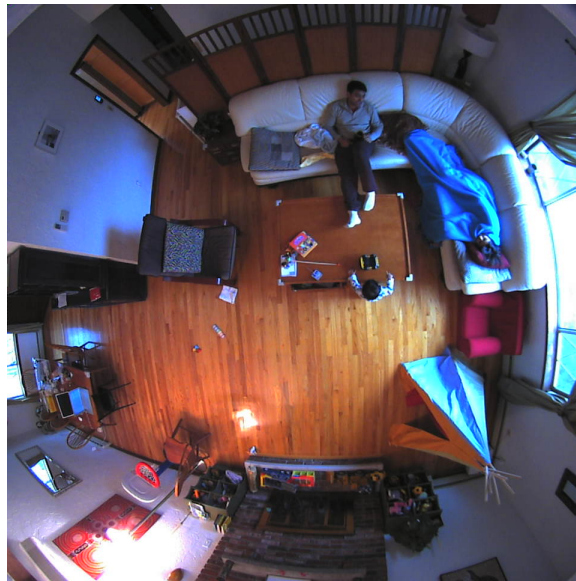


Figure 6.6: The target subject, at the top, shifts his attention between the two attendant subjects.

Figure 6.6 shows the first frame of video for a Trial 3. Three subjects appear in this video segment: the top subject is the target to be tracked, and the bottom two are attendant

subjects. As the video progresses, the target shifts his attention between the two nearby persons. Figure 6.7 shows several estimates from this trial. The third and fifth images show the target looking at the child in front of him, and the other images show the target looking at the adult on his left. Although the head orientation only changes slightly as the target looks between the two people, the estimated poses appear adequate to resolve the focus of attention. This trial ended after 19.76 seconds when the target partially occluded his face by drinking from a cup.



Figure 6.7: Selected pose estimates from Trial 3

6.2 Results by Subject

Trial results were significantly affected by the identity of the target being tracked. Table 6.1 displays the mean time-to-failure for each subject.

Subject A had the best performance, which might be caused by having the most accurately matching model. *Subject A* possesses short hair, resulting in a well-defined hairline that changed little across the test set.

Although *Subject B* also had short hair, this subject is a young child with significantly different proportions than the associated 3D model. In addition, *Subject B* has a much smaller head than the other subjects and head pose must be estimated on much smaller regions.

Subject C had the worst performance. This is likely due to *Subject C* possessing long hair that does not match with the short hair on the 3D model.

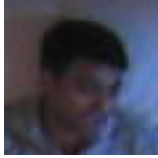
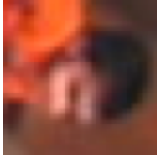
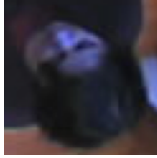
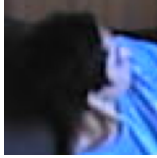




	Subject A	Subject B	Subject C	Subject D	Total
	Caregiver 	Child 	Caregiver 	Caregiver 	
Model					
# of Trials	12	18	17	22	69
Mean TTF	6.11	1.78	1.05	2.13	2.47

Table 6.1: Mean time-to-failure by subject.

Figure 6.8 shows one of several trials in which inaccuracies in the model caused an error. To give a better sense of how HeadLock operates, the bottom row displays how HeadLock transforms each frame of video before performing a pose search. In the bottom row images, pixels classified as *hair* are colored red, pixels classified as *skin* are colored green, and pixels classified as *background* are made transparent and appear white on the page. Although it is not apparent, the luminance of the original frame is mapped to the blue channel.

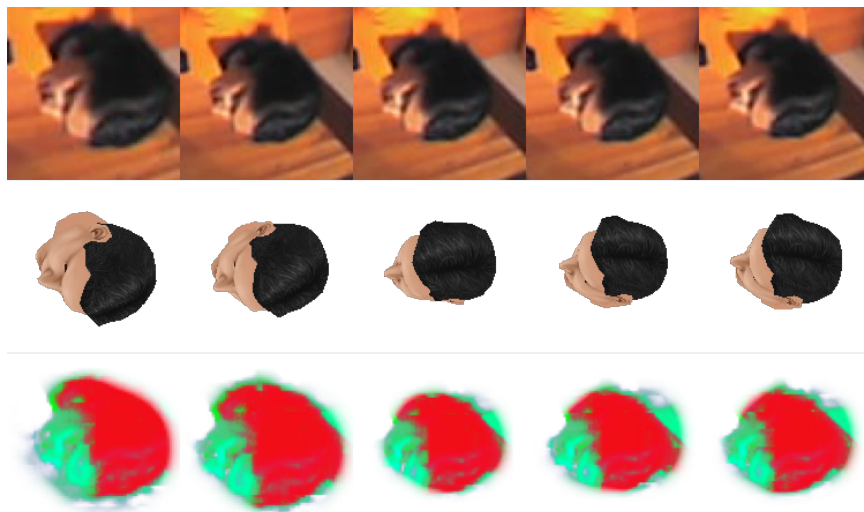


Figure 6.8: An inaccurate head model caused trial 47 to fail quickly.

The background in this trial had similar colors as the face, but the bottom row of images shows that HeadLock had no problems differentiating the hair, skin and background. However, because the subject’s hair covers the ear and neck, HeadLock attempts to hide the neck of the 3D model and increase the prominence of the hair. As a result, the system fails after just a few frames.

Part of this problem may be fixed by simply providing more accurate 3D models for each subject. However, for subjects with long hair, the shape of the head is more morphable and may change from frame to frame. The lack of accurate 3D models and the lack of a mechanism to accommodate head morphability had significant impact on performance.

6.3 Initialization Effects

Trials tend to perform better when the seed frame clearly shows both the skin and hair of the target subject. HeadLock uses a pixel classification step that is trained only on the seed frame, meaning that the seed frame alone determines the performance of classification. In Figure 6.9(a), the first image shows the head and the second shows the pose that was provided by the human annotator. The next three images each show the three sets of pixels used for training the segmentation classes, one each for hair, skin, and background. Although the alignment is not perfect, most of the pixels in each training set are labeled correctly. The final image shows the resulting image transform.

Figure 6.9(b) shows an example of poor classification performance. The seed frame contains no skin pixels, but the rendered head model contains a number of skin pixels in the back of the neck and tips of the ears. As a result, the set of pixels used for training the skin model are completely inaccurate and the resulting segmentation mistakes a large, indistinct portion of the background as skin. This trial ended after just two frames of video.

The test set contains numerous seed frames that contain many hair pixels but few skin pixels. A human operator can easily glance at a head and determine the proportion of hair to skin, and thus could select frames with a balanced distribution to use as seed frames.

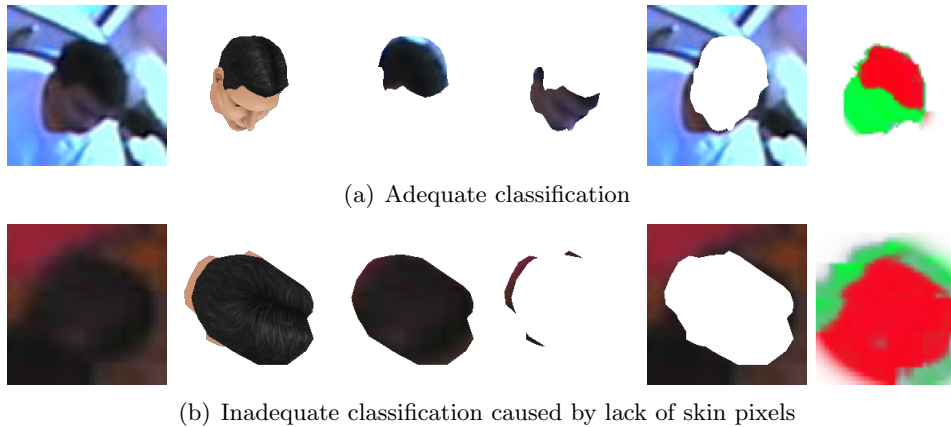


Figure 6.9: A computer rendered head is used to extract the hair, skin, and background pixels from a frame of video. Inaccurate or insufficient pixel labels may degrade classification performance.

To test whether this heuristic actually improves performance, the test set was culled of all video segments containing a poor skin-to-hair distribution. Specifically, a video segment was removed if the hair pixels constituted less than 25% or more than 75% of the total pixels belonging to the head. This reduced test set contains 35 video segments, representing 51% of the full test set.

The results of the reduced set are shown in Table 6.2. The results show substantial increases in mean time-to-failure. In the full test set, subjects *B* and *D* had the largest number of video segments with poor skin/hair distributions, and also saw the largest increase in performance when these segments were removed. The performance of *Subject B*, the toddler, more than doubled. *Subject C*, the subject with long hair, remained the only subject with a mean time-to-failure of less than three seconds.

Subject	# of Trials	MTTF (Full Set)	MTTF (Reduced Set)	Δ MTTF
Subject A	8 of 12	6.12	5.58	-8.7%
Subject B	6 of 18	1.78	4.44	149.0%
Subject C	14 of 17	1.05	1.07	2.1%
Subject D	7 of 22	2.13	3.53	65.4%
Total	35 of 69	2.47	3.17	28.6%

Table 6.2: Mean Time-To-Failure(MTTF) for trials in both the full test set and the reduced test set with balanced skin-to-hair ratios

6.4 Segmentation Errors

Another common mode of failure was caused by the blending of foreground and background colors in the video. The pixel classifier used by HeadLock only relies on color values and does not extract edge or texture information. Additionally, the pixel classifier is trained on a small region around the target head in the initial frame. Without an adaptive learning strategy, the classifier often produces poor results when the target moves to a region of the room with a different background. Figure 6.10 shows a trial that failed when the subject moved into a dark background region. Although the initial pose estimates are not completely accurate, they remain within the 30° limit for the first 20 frames of video. On the 21st frame, the subject moves in front of a shadow. The shadow is mistaken for hair, causing the system to produce an invalid estimate.

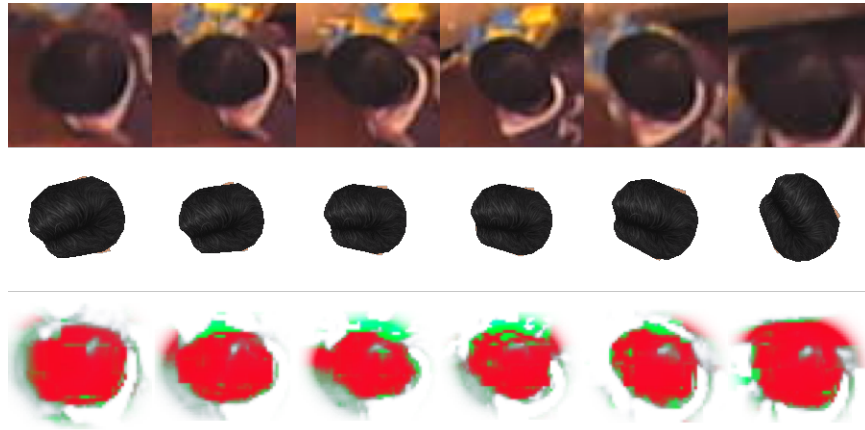


Figure 6.10: Trial 27 terminated when the target moved in front of a background with similar color values as his hair.

Figure 6.11 shows several additional examples. Confusing shadows and dark hair may be the most common problem, as in Figure 6.11(a). In figures 6.11(b) and 6.11(c), the target moved in front of a background that was mistaken for skin. In cases where the target head overlaps the head of another subject, such as in Figure 6.11(d), HeadLock was unable to determine which pixels belonged to the target head. In Figure 6.11(e), the target head is exposed to direct sunlight and much of the head is obscured by areas of saturated brightness, making it impossible to use color to distinguish the body from the head. Video segments with this level of saturation caused HeadLock to fail immediately.

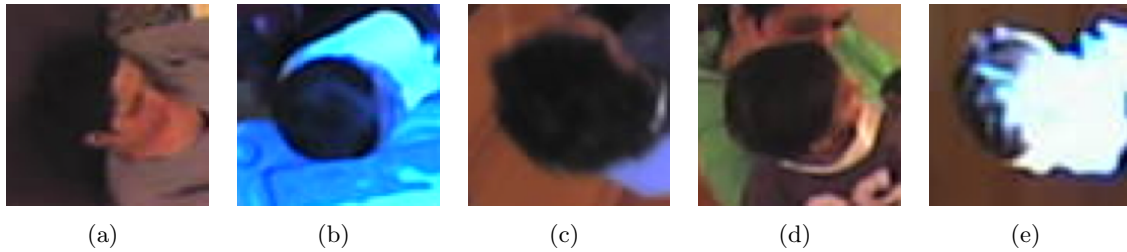


Figure 6.11: Examples of difficult segmentation problems found in the test set.

6.5 Estimation Confidence

In the context of a semi-automated pose annotation system, the ability for the system to recognize when it has made a mistake offers several potential enhancements to the annotation process. If the system has low confidence in its estimates, it may be able to notify the user and reduce the number of overlooked errors. It might also be used to automatically select a suitable frame for the user to manually annotate, and thus reduce the amount of time the user must spend browsing through frames.

For HeadLock, head pose is estimated by minimizing a cost function. If the cost of a pose is correlated with the error, it may serve as a convenient confidence score that could be used for error detection. Two tests were conducted, one to determine if the pose cost was correlated with the time-to-failure, and one to determine if the pose cost was correlated with error.

For the first test, the *initial cost* of each trial was computed by rendering a model head with the pose parameters provided by the seed annotation, and then applying the cost function to the rendered head and the initial frame. Figure 6.12 displays a scatter plot that relates the initial cost to the time-to-failure for all trials. No statistically significant regression was found for this data and it appears unlikely that the initial cost can be used to predict error.

The second test was designed to determine if the pose cost tends to increase when an error is made. For each trial that lasted at least one second, the *change in cost* was computed as the cost of the final, incorrect estimate minus the cost of the correct estimate that was provided 0.5 seconds before. Figure 6.13 shows the distribution of change in costs. In only

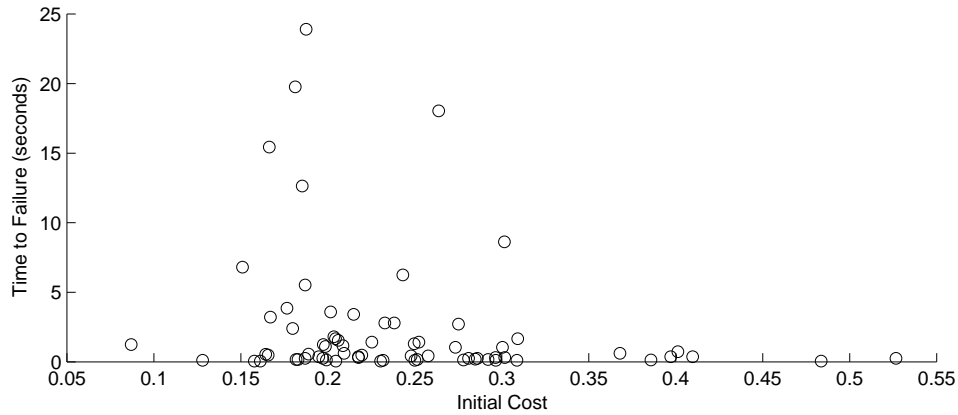


Figure 6.12: Time-to-failure plotted as a function of the initial pose cost.

47% of the trials was failure accompanied by an increase in pose cost. Both tests indicate that the current system does not readily offer a method for automatically inferring error and this goal must remain for future work.

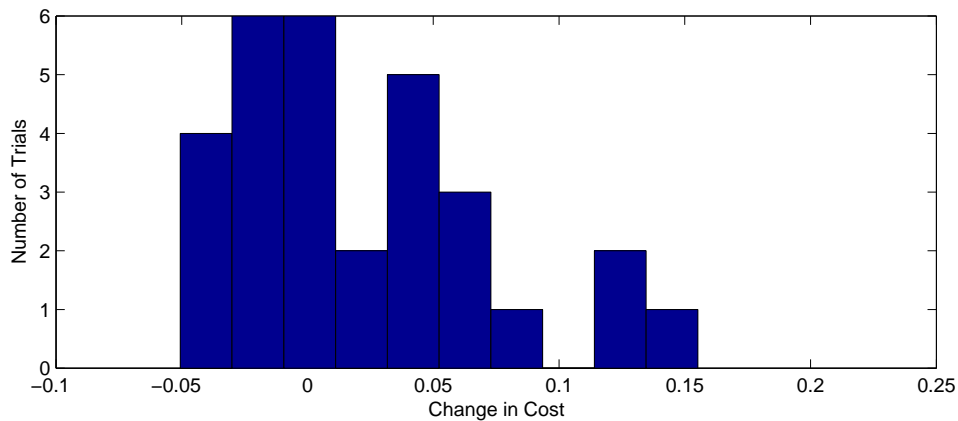


Figure 6.13: Distribution of change in pose cost during last .5 seconds of trials.

6.6 Comparisons to Other System Configurations

HeadLock contains numerous components that could not be exhaustively analyzed for this evaluation. However, a few alternate configurations were examined to evaluate a few of the key design decisions.

In the first stage of the pose estimation process, a tracker is applied to acquire the approxi-

mate position of the head. The system may function without the tracker, but it must then accommodate large shifts in position using only the hill-climbing parameter search. The parameter search requires a 3D render and cost calculation for each set of parameters, which is more computationally expensive and generally more applicable to fine-tuning position.

To test the actual utility of the tracker, a modified system was run on the first 55 samples of the full test set. Without the tracker, HeadLock had a mean time-to-failure of 2.31 seconds compared to a mean of 2.47 for the full system. The distribution over time-to-failure is shown in Figure 6.15, which shows a significant decrease in the number of trials that last more than a second.

In the second stage of the pose estimation process, each video pixel is transformed from RGB to a 4-dimension feature vector. This process relies heavily on the color values of the frame. To determine how well the system could perform with just luminance values, the test set videos were converted to grayscale and an evaluation was run on the first 55 samples. Without full color information, the system had a mean time-to-failure of 1.66. While it is not surprising that the performance dropped, the severity of the drop demonstrates HeadLock's critical dependence on color.

As described in Section 2.3, several other image transforms and pose cost metrics were tested throughout development. Image transforms include difference-of-gaussians filters, Sobel filters, the unaltered RGB values, and grayscale. Other cost metrics include inner product, euclidean distance, mutual information, and variations on these functions. None of these tests was successful enough to warrant a full evaluation, but these attempts were neither exhaustive nor conclusive.

6.7 Head Annotation Efficiency

In the course of this evaluation, two operators entered 674 annotations using the head annotation interface. For each annotation, the operator uses a mouse to draw a rough bounding box around a head, selects the identity of the head from a popup menu, positions

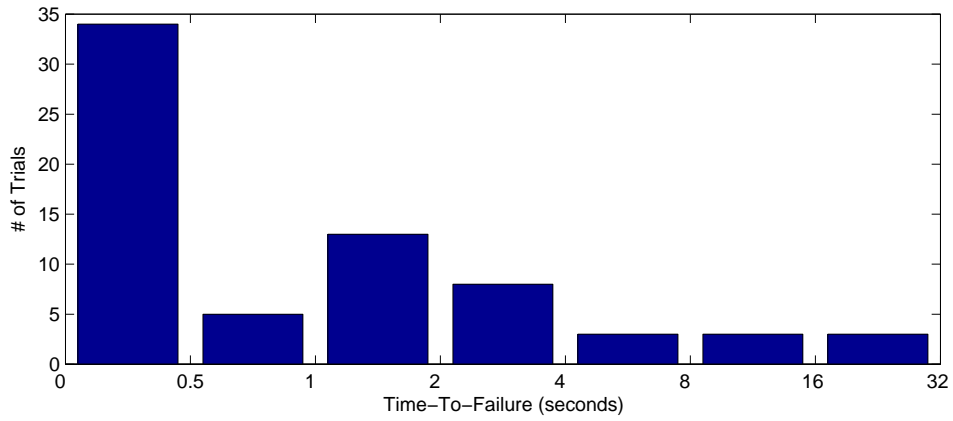


Figure 6.14: TTF distribution for base configuration, shown in logarithmic space.

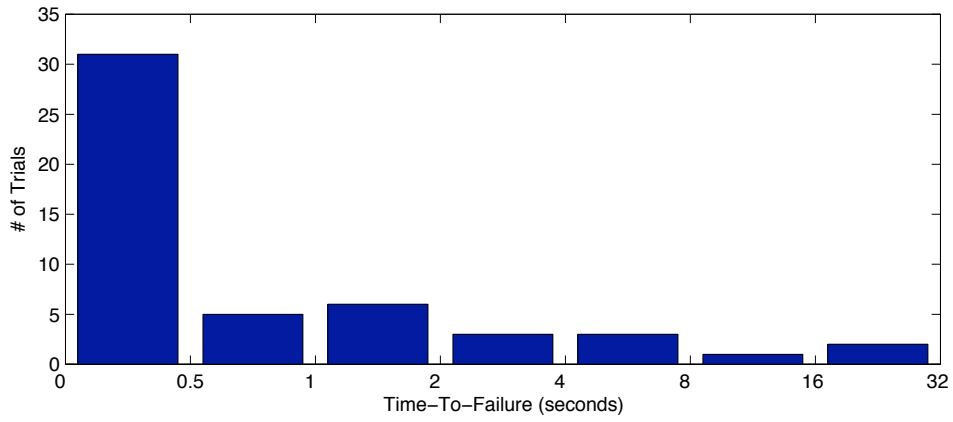


Figure 6.15: TTF distribution for system without tracker, shown in logarithmic space.

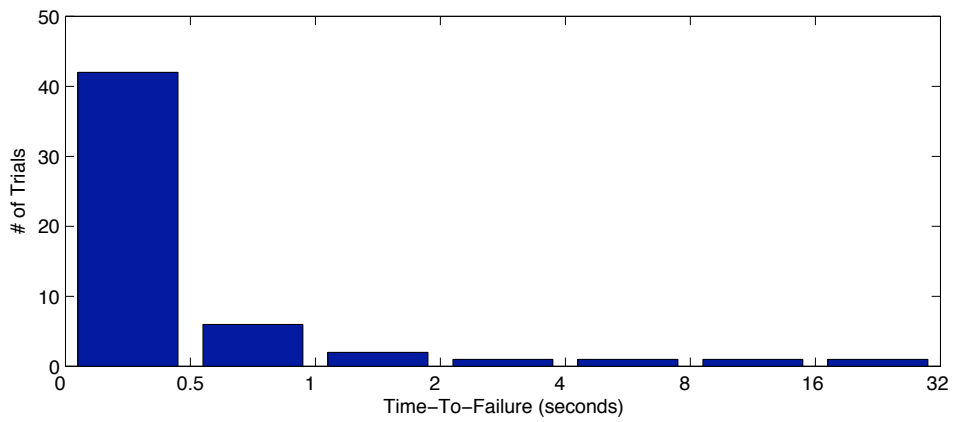


Figure 6.16: TTF distribution for system using grayscale, shown in logarithmic space.

and orients the head with a 6-DOF input device, and then presses a button on the input device to signal that the annotation is complete. The time to annotate one head is defined as the time it takes to complete this sequence of steps. Table 6.3 shows the recorded average time to annotate one head. The first 239 annotations are removed from this figure to remove training effects.

The evaluation was not specifically designed to measure annotation time, but the recorded times may still provide a rough estimate. In order to evaluate HeadLock, it was desired to provide consistent annotations and operators were instructed to annotate carefully, particularly for the seed frame. This may have resulted in longer annotation times than would be observed in an actual annotation task. Inter-operator agreement was not measured.

	Samples	Mean(s)
Operator 1	297	12.7
Operator 2	138	13.8
Total	536	13.0

Table 6.3: Average time to annotate one head

The speed at which a single head may be annotated for a segment of video depends on the desired frequency of annotations. One of the goals of HSP is to analyze intermodal coordination of speech and body orientation, including head orientation. For this purpose, the ideal frequency of head pose estimates would match the frequency of spoken words. The speed of conversational speech between adults, roughly 200 words per minute, suggests that head annotations should be made every quarter second. At this frequency, it would take approximately one hour of human effort to manually annotate one minute of video.

In a full annotation system that allowed the operator to annotate only the frames that HeadLock misestimates, the evaluation results indicate that the user would need to provide one annotation for every 2.47 seconds of video rather than every quarter second. This implies that the current HeadLock system may speed up annotation by at most 9.9 fold, which would allow the operator to annotation one minute of video in six minutes. In practice, speed is reduced by the inevitable time the operator makes between annotations and the time HeadLock takes to provide estimates. Achieving a true 10x increase in speed

will require improving the robustness and speed of HeadLock, which is discussed in the next chapter.

7 Future Work

HeadLock represents an effort to overcome the annotation and data mining challenges posed by HSP. However, HeadLock is not yet ready to be employed as a practical tool for a large scale annotation project. Not only has HeadLock not been developed into a full annotation system, but the core estimation process contains weaknesses through nearly all of the estimation components. This section will explore how HeadLock may be improved and define a roadmap for further development.

Alternative Feature Spaces

The basic approach to head pose estimation taken by HeadLock is to track a head with a dedicated tracker, transforming the image through a segmentation process, searching for pose by rendering 3D head models and applying a cost metric. Without changing the overall approach, each stage of the process may be reformulated and improved.

Perhaps the most limiting component is the segmentation stage, which transforms each pixel to a 4-dimension feature vector that contains luminance and class probabilities for skin, hair and background. The mixture-of-gaussians model used to classify pixels was, if not over-simple, quite inaccurate. In video where the background shared exact color values with the foreground, classifying each pixel independently is simply not possible. The extensive body of research on segmentation is certain to offer more robust solutions, but from a broader perspective, it is not clear that segmentation even provides the optimal feature space for pose searching.

I anticipate exploring other possible image transforms and feature spaces. Obvious candidates for features include edges and textures. Although the resolution of the heads in the HSP video is limited, it may also be possible to extract more structured features. For

example, the eyes of a subject might be located by looking for two dark spots in the head region. Such information can be added to the feature vector for each pixel and would not affect the implementation of the pose search.

Tracking

The evaluation showed a clear advantage to using a dedicated tracker to determine the rough position of the head before performing a pose search. Improving the accuracy of the tracker could significantly increase overall performance. There are two deficiencies in the tracker that I plan to improve.

The tracker is trained on the initial frame of video and models the target by examining the distribution of colors. As subsequent frames are examined, the tracker does not update its target model. This is problematic when tracking heads because the color distribution can change dramatically as the head rotates or moves to a region with a different background. This is a well known limitation of mean-shift trackers and the common solution is to use a variation called *CAMShift*, which updates the color model for each frame [7]. CAMShift is simple to implement and would likely have been a better design choice.

The second problem is that the tracker attempts to track just the head, which provides few pixels. Tracking the entire body would greatly increase the region being tracked and provide a more stable color distribution. A hierarchical tracker might be implemented that first estimates the position of the entire target, and then estimates the position of the target's head in relation to the body. This may provide more reliable results when the body is easy to track but the head is difficult to distinguish from the background, such as when a black-haired target steps in front of a dark shadow.

Head Model Generation

An obvious deficiency in the HeadLock evaluation was the 3D model used to represent the head of each subject. Although these models are critical to pose search and to the performance of HeadLock, creating an accurate model for each subject was inconvenient. For the complete system, it will be necessary to create new head models as they are needed.

I plan to develop an interface that allows an annotator to create new heads during the

annotation process. The interface would require a parametric head model that minimally contains parameters for the shape and dimensions of the head, hair color, skin tone, hair contours and hair length. The annotator would be able to generate new heads by adjusting these parameters until the 3D model adequately matches the real head. Head models may be associated with targets based on the target’s identity and the time of the video. For example, child subjects may be given a different head model every few months to account for the child’s physical development, or an adult subject may be given a new head model after receiving a haircut. When annotating head pose, HeadLock would be able to select a suitable model based on the identity of the target and the time of the video.

Although this model creation system would account for changes in the subjects over long periods of time, it does not address the problem of the heads with long hair that do not have a rigid form. HeadLock performs a search over the 6-dimensional pose parameters to find a match. The dimensionality of the search could be increased to include parameters for the shape of the head, although this could severely impact performance and may not be entirely effective. A more simple solution may be to ignore the regions of the head which are morphable. Even for subjects with long hair, the face and top of the head does not typically change. The stable parts of the head may be indicated on the 3D model, and a cost metric may be devised that ignores the unstable portions head.

A more distant goal may be to generate head models automatically. Model fitting algorithms have been implemented by a number of researchers, including [38, 8]. However, model fitting is a more complicated procedure and, depending on the amount of data that the operator must provide before a suitable model is produced, it may not improve annotation efficiency beyond that of a specialized manual interface.

Handling Multiple Annotations

The system described in this thesis uses a single annotation at a time and attempts to propagate the annotation forward. In a real annotation task, the user will provide multiple annotations to correct and guide the system. Rather than utilize each annotation independently in a feed-forward manner, HeadLock may be modified to use multiple annotations to propagate annotations both forward and backward in time.

Ideally, when the user provides an annotation, the system would first use the annotation to adjust previous estimates. A simple approach might be to perform linear interpolation over limited time window. A more advanced approach might use a continuous-state Viterbi algorithm or a related filter formulated for continuous-state hidden Markov chains. Furthermore, if HeadLock can produce a reliable confidence for each estimate, the confidence scores may be utilized by the filter to improve accuracy.

In addition to filtering pose estimates, HeadLock could be modified to generate pose estimates bidirectionally. HeadLock has no dependencies on the direction of time. There is no reason to believe it would function differently going backwards in time and that the system would be unable to produce twice as many correct annotations by estimating in both directions. For example, when an annotator determines that a target has been incorrectly estimated, the annotator may choose not to annotate the misestimated frame, but a frame that occurs shortly after. This would allow the annotator to choose a more suitable frame from which to begin the estimation process and may reduce the number of manual annotations required by having the system to estimate in both directions. Bi-directional propagation could not be used to double the efficiency, which would require that the annotator perfectly predict the midpoint between frames that cause failure, but may produce a modest improvement.

Optimization

The time HeadLock requires to produce an estimate is proportional to the resolution of the target and roughly varies between four and seven seconds for the targets in the HSP corpus. Ideally, the system would process video much faster than realtime, but to be useable within an interactive annotating environment, it will at least need to run at or near realtime. This will require a 50 fold increase in speed, approximately. Although large, there are many possibilities for achieving this speedup. The most significant optimizations will need to address the pose search phase, in which a 3D head model is rendered in different poses and compared to the video. This operation consumes the overwhelming majority of the processing resources.

HeadLock has not been optimized, and is written in the Java programming language. The

language is particularly significant in this case because Java does not provide direct access to the hardware accelerated graphics library on which HeadLock relies. Implementing this search in a lower-level language, such as *C*, is likely to increase rendering time significantly. Pre-rendering the head model at different poses or image caching may help, but the size of the parameter space is large and it is not known if rendered images could feasibly reduce the rendering time by a significant amount.

Additionally, the pose search has several parameters that affect the number of poses that are examined: the *initial entropy* that determines the magnitude with which pose parameters are altered at each step, the *entropy decay* that determines how quickly the entropy is decreased, and the *minimum entropy* that specifies how much the entropy must decay before the search is halted. The current annealing process reduces the entropy 22 times in 10% increments and, for each of the 22 entropy values, all six pose parameters are optimized. These entropy parameters have not been thoroughly evaluated and it is likely that different values may be found that provide nearly equivalent performance while requiring fewer poses.

Finally, the current implementation only utilizes a single processor for the search. After cost of a given pose is computed, the next pose is determined according to whether the cost increased or decreased. At worst, the next pose may be predicted 50% of the time. Thus, adding an additional processor and graphics card that are used to predictively compute pose costs should increase search speed by at least 50%.

The Full Interface

So far, this section has focused on how the head pose estimation portion of HeadLock may be improved. However, the current HeadLock system does not constitute a full annotation environment. The interface used for evaluation was designed to test the initial performance of HeadLock on a single annotated frame, but omits many features required for large-scale annotation operations. The design of the full interface has not been determined, but several aspects have been considered.

As part of the *TotalRecall* system, many of the components that HeadLock requires have already been implemented. TotalRecall provides data browsing and visualization that allows

a user to navigate the data and select portions of video to annotate. TotalRecall also provides features to grant various privileges to individual operators and to manage multi-operator annotation tasks. A *supervisor* can create assignments for different annotators that specify portions of data to be annotated. Access to the HSP corpus may be restricted to the data that is required for an annotator to complete an assignment. Multiple annotators may be given overlapping assignments so that annotator accuracy and agreement may be evaluated.

One of the relatively unexplored aspects of HeadLock is how the system will provide feedback to the user. This will likely depend on how much the accuracy and speed of HeadLock can be improved. If the user must examine each head annotation that HeadLock produces automatically, watching the video as it is annotated may be the appropriate solution. The estimated pose for each frame might be represented by the 3D head model, as shown in the evaluation. It may also prove helpful to illuminate the portion of the video at which each subject is gazing. In Chapter 1, Figure 1.1 displays a visualization of head pose that literally illuminates the “cone of attention” for each subject. In the alternate case in which the operator is not required to examine every frame, it would be helpful to find a visualization technique that can simultaneously show annotations from many frames in such a way that still reveals errors.

8 Conclusion

The goal of HSP is to analyze language acquisition using longitudinal audio-video recordings in a home environment. The scale of HSP is qualitatively different from that of previous efforts and represents a new research methodology that poses a number of new challenges. The research of this thesis addresses supports this effort and addresses many of the initial research challenges, including data collection, management, and annotation. While this thesis does not address the problems of analyzing language acquisition, it provides a foundation for such research, and many also be applied to other research efforts that utilize large corpora of video.

Advancements in recording and storage technologies will likely increase the frequency of large scale recording projects, whether applied to scientific research, assisted living, or understanding shopping behaviors. The data collection system developed for HSP demonstrates how a system may be designed for long term deployment in a home environment while providing the necessary privacy controls for voluntary subjects. Although I believe the overall configuration and features of the system are unique, the development of the system did not require overcoming any serious technical challenges for which solutions did not already exist. The data collection system uses components that are readily available and relatively inexpensive, or for which inexpensive alternatives exist. In terms of just the data collection process, the most significant limiting factor appears to be the cost of data storage. However, longitudinal audio-video recording may not require the number of sensors and quality of recordings that were desired for HSP, and similar recording efforts may be affordable for smaller research projects even now.

The true cost of using large audio-video corpora for research is not collection, but the human effort required to extract useful information. Speech transcription represents one of

the most common challenges for utilizing audio recordings. To manually transcribe one hour of speech with word-level alignment typically requires 15 to 20 hours of human effort. The standards and practices of annotating video at a similar resolution have not been established at nearly the same level as those for the audio, but most video annotation tasks are likely to be considerably more laborious. The challenge that HSP has accepted is to reduce the cost of data extraction enough to expand the domains in which the analysis of ultra-dense longitudinal recordings is a viable option.

My colleagues and I made the first steps towards utilizing the HSP data by developing TotalRecall [22]. TotalRecall expands on familiar interfaces for video and audio editing, but provides browsing and visualization capabilities for a very large range of temporal resolutions. The method of continuous video visualization has been custom implemented to enhance browsing for the HSP video, and has proven invaluable for manual browsing and annotation. TotalRecall has also been designed to accommodate multiple annotators and includes organization tools for creating and assigning annotation tasks. TotalRecall continues to evolve and, regardless of whether the software is adopted by other research projects, may serve as useful example for future data management applications.

Although significant effort has been expended on developing the right interfaces and protocols for data mining, equal importance has been placed on the development of machine perception technologies that may be utilized, at the current time, to achieve the biggest reductions in annotation cost. Although the state-of-the-art in machine perception is far from perfect in most areas, the flawed algorithms that are available may still aid a human annotator and may greatly improve annotation efficiency.

The largest focus of this thesis, as well as the most technically challenging, is the development of a system for head pose annotation that uses computer vision techniques to reduce the required number of manual annotations. HeadLock is being developed specifically to operate in tandem with a human. The head estimation that HeadLock provides is designed to work with monocular video from a still camera. It was evaluated on a challenging data set, which contained low resolution targets that provide few facial features, cluttered backgrounds and varied lighting conditions. Although the evaluation revealed

many instances in which the current HeadLock system performs poorly, the results indicated that the system may still reduce the required number of human annotations by an order of magnitude.

Decreasing the number manual annotations may not result in a proportional increase in efficiency. However, by further developing and optimizing the basic approach taken by HeadLock, I believe it will be possible to improve head pose annotation efficiency by at least an order of magnitude. Using the annotation protocol that was given at the end of the *evaluation* section, this corresponds to reducing the time to annotate one minute of video from one hour to six minutes. To fulfill this promise, it will be necessary to improve the robustness and speed of the pose matching process, improve the accuracy of the 3D models, and to complete the full interface to accommodate practical annotation tasks.

The research described in this thesis began with the inception of the Human Speechome Project, and follows the progress of the project through several milestones. HeadLock addresses the current challenges of mining information from the HSP video, and contributes to the solution by providing an new approach to head pose annotation. In addition to improving the state of head pose annotation, it is hoped that this work outlines many of the challenges that must be addressed to effectively collect, manage, and analyze longitudinal audio-video recordings, whether it be for scientific research, health monitoring, smart homes, or domains that are yet to be defined.

References

- [1] Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004.
- [2] Gaurav Aggarwal, Ashok Veeraraghavan, and Rama Chellappa. 3d facial pose tracking in uncalibrated videos. In Sankar K. Pal, Sanghamitra Bandyopadhyay, and Sambhunnath Biswas, editors, *PReMI*, volume 3776 of *Lecture Notes in Computer Science*, pages 515–520. Springer, 2005.
- [3] Murray Aitkin and Granville Tunnicliffe Wilson. Mixture models, outliers, and the em algorithm. *Technometrics*, 22(3):325–331, August 1980.
- [4] Shumeet Baluja, Mehran Sahami, and Henry A. Rowley. Efficient face orientation discrimination. In *ICIP*, pages 589–592, 2004.
- [5] Michael J. Black and Yaser Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *Int. J. Comput. Vision*, 25(1):23–48, 1997.
- [6] Lois Bloom. *One word at a time: the use of single word utterances before syntax*. Mouton, The Hague, 1973.
- [7] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2):15, 1998.
- [8] Xavier L. C. Broly, Constantinos Stratelos, and Jeffrey B. Mulligan. Model-based head pose estimation for air-traffic controllers. In *ICIP (2)*, pages 113–116, 2003.
- [9] J. Bruner. *Child’s Talk: Learning to Use Language*. Norton, 1983.
- [10] Malinda Carpenter, Katherine Nagell, Michael Tomasello, George Butterworth, and Chris Moore. Social cognition, joint attention, and communicative competence from 9 to 15 months of age. In *Monographs of the Society for Research in Child Development*, volume 63 of 255. 1998.
- [11] Marco La Cascia, Stan Sclaroff, and Vassilis Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(4):322–336, 2000.
- [12] V. Comaniciu and P. Meer. Kernel-based object tracking, 2003.
- [13] Jonathan Dakss. Hyperactive: An automated tool for creating hyperlinked video. Master’s thesis, Massachusetts Institute of Technology, 1999.

- [14] G. Daniel and M. Chen. Video visualization. In Robert Moorhead Greg Turk, Jarke J. van Wijk, editor, *IEEE Visualization 2003*, pages 409–416, Seattle, Washington, USA, October 2003. IEEE Press.
- [15] P. Deutsch. Deflate compressed data format specification version 1.3. Technical Report RFC-1951, Network Working Group, May 1996.
- [16] Nicolas Gourier, Daniela Hall, and James L. Crowley. Estimating face orientation from robust detection of salient facial structures. *FG Net Workshop on Visual Observation of Deictic Gestures (POINTING)*, 2004.
- [17] Nicolas Gourier, Jérôme Maisonnasse, Daniela Hall, and James L. Crowley. Head pose estimation on low resolution images. In *CLEAR Workshop, In Conjunction with Face and Gesture*. Springer Verlag, April 2006.
- [18] ITU. Information technology - digital compression and coding of continuous-tone still images - requirements and guidelines. Technical Report ISO/IEC 10918-1, CCITT, 1992.
- [19] T.S. Jebara and A. Pentland. Parametrized structure from motion for 3d adaptive feedback tracking of faces. *cvpr*, 00:144, 1997.
- [20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- [21] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.
- [22] Rony Kubat, Philip DeCamp, Brandon Roy, and Deb Roy. Totalrecall: A system to browse and annotate hundreds of thousands of hours of video and audio. In *ICMI '07: Proceedings of the 9th International Conference on Multimodal Interfaces*, 2007.
- [23] Tai Sing Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.
- [24] Guoyuan Liang, Hongbin Zha, and Hong Liu. 3d model based head pose tracking by using weighted depth and brightness constraints. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics (ICIG'04)*, pages 481–484, Washington, DC, USA, 2004. IEEE Computer Society.
- [25] Brian MacWhinney. *The CHILDES project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Mahwah, NJ., 2000.
- [26] A. Manzanera and J. Richefeu. A robust and computationally efficient motion detection algorithm based on sigma-delta background estimation. *Proceedings Indian Conference on Computer Vision, Graphics and Image Processing*, 2004.
- [27] S. McKenna and S. Gong. Real time face pose estimation.
- [28] David L. Mills. Network time protocol (version 3): Specification, implementation and analysis. Technical Report RFC-1305, Network Working Group, March 1992.

- [29] Katherine Nelson. Concept, word and sentence: Interrelations in acquisition and development. *Psychological Review*, 81:267–285, 1974.
- [30] J. Ng and S. Gong. Multi-view face detection and pose estimation using a composite support vector machine across the view sphere, 1999.
- [31] Deb Roy, Rupal Patel, Philip DeCamp, Rony Kubat, Michael Fleischman, Brandon Roy, Nikolaos Mavridis, Stefanie Tellex, Alexia Salata, Jethran Guinness, Michael Levit, and Peter Gorniak. The human speechome project. In P. Vogt 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.
- [32] A.J. Smola and B. Schoelkopf. A tutorial on support vector regression, 1998.
- [33] Ying-Li Tian, Lisa Brown, Jonathan Connell, Sharat Pankanti, Arun Hampapur, Andrew Senior, and Ruud Bolle. Absolute head pose estimation from overhead wide-angle cameras. In *AMFG '03: Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, page 92, Washington, DC, USA, 2003. IEEE Computer Society.
- [34] Michael Tomasello and Michael Jeffrey Farrar. Joint attention and early language. *Child Development*, 57(6):1454–1463, December 1986.
- [35] Michael Tomasello and Daniel Stahl. Sampling children’s spontaneous speech: how much is enough? *Journal of Child Language*, 31:101–121, March 2004.
- [36] Jilin Tu, Thomas Huang, and Hai Tao. Face as mouse through visual face tracking. *cvv*, 00:339–346, 2005.
- [37] Ragini Choudhury Verma, Cordelia Schmid, and Krystian Mikolajczyk. Face detection and tracking in a video by propagating detection probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1215–1228, 2003.
- [38] Ying Wu and Kentaro Toyama. Wide-range, person- and illumination-insensitive head orientation estimation. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 183, Washington, DC, USA, 2000. IEEE Computer Society.
- [39] Günther Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience, second edition, 2000.