A Human-Machine Collaborative Approach to Tracking Human Movement in Multi-Camera Video

Philip DeCamp
MIT Media Lab
20 Ames Street, E15-441
Cambridge, Massachusetts 02139

Deb Roy
MIT Media Lab
20 Ames Street, E15-488
Cambridge, Massachusetts 02139

ABSTRACT

Although the availability of large video corpora are on the rise, the value of these datasets remain largely untapped due to the difficulty of analyzing their contents. Automatic video analyses produce low to medium accuracy for all but the simplest analysis tasks, while manual approaches are prohibitively expensive. In the tradeoff between accuracy and cost, human-machine collaborative systems that synergistically combine approaches may achieve far greater accuracy than automatic approaches at far less cost than manual. This paper presents TrackMarks, a system for annotating the location and identity of people and objects in large corpora of multi-camera video. TrackMarks incorporates a user interface that enables a human annotator to create, review, and edit video annotations, but also incorporates tracking agents that respond fluidly to the users actions, processing video automatically where possible, and making efficient use of available computing resources. In evaluation, TrackMarks is shown to improve the speed of a multi-object tracking task by an order of magnitude over manual annotation while retaining similarly high accuracy.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.5.2 [**Information Interfaces and Presentations**]: User Interfaces

General Terms

Performance, Human Factors, Design

Keywords

video annotation, multiple camera, object tracking, human-machine collaboration

1. INTRODUCTION

The ubiquity of digital video cameras coupled with the plummeting cost of storage and computer processing enables new forms of human behavioral analysis that promise to transform forensics, behavioral and social psychology, ethnography, and beyond. Unfortunately, the ability for state-of-the-art automatic algorithms to reliably analyze fine-grained human activity in video is severely limited in all but the most controlled contexts. Object tracking represents one of the most active and well developed areas in computer vision, yet existing systems have significant difficulty processing video that contains adverse lighting conditions, occlusions, or multiple targets in close proximity. During the 2007 CLEAR Evaluation on the Classification of Events, Activities, and Relationships[10], out of six systems applied to tracking persons in surveillance video, the highest accuracy achieved was 55.1% as computed by the MOTA metric[2]. While higher accuracy systems may exist and clearly progress will continue to be made, the performance gap between human and machine visual tracking for many classes of video is likely to persist into the foreseeable future.

While automated processing may be sufficient for some applications, the focus of this paper is to investigate the use of tracking algorithms for applications that demand a level of accuracy beyond the capability of fully automatic analysis. In practice today, when video data is available but automatic analysis is not an option, manual analysis is the only recourse. Typically, the tools for manual annotation of video are extremely labor intensive barring use in all but the most resource-rich situations. Our aim is to design *human-machine collaborative* systems that capitalize on the complementary strengths of video analysis algorithms and deep visual capabilities of human oversight to yield video tracking capabilities at a accuracy-cost tradeoff that is not achievable by full automation or purely manual methods alone.

The human-machine collaborative approach to information technology was first clearly articulated by J. C. R. Licklider[6] as a close and fluid interaction between human and computer that leverages the respective strengths of each. He envisioned a symbiotic relationship in which computers could perform the "formulative tasks" of finding, organizing, and revealing patterns within large volumes of data and paving the way for human interpretation and judgment. As computers have proliferated over the past fifty years, instances of what Licklider might consider collaborative systems have become pervasive, from the computerized braking and suspension in modern cars that help the driver stay on the road to the spell checkers that review each word as the user types it. While human-machine collaboration is now implicit in many fields, the conceptual framework still provides insight when addressing the problem of video annota-

CIVR '09, July 8-10, 2009 Santorini, GR

tion. Given that humans can perform visual tasks with great accuracy, and that computers can process video with great efficiency, finding an effective bridge between the two may yield an annotation system that process performs with much greater efficiency than manual annotation while making few concessions to accuracy.

In this paper, we present TrackMarks, a human-computer collaborative system designed to annotate large collections of multi-camera video recordings. Specifically, this paper will describe TrackMarks as applied to annotating person identity and location, although the system may be adapted to other video analysis tasks. When using TrackMarks, the human annotator begins the process by providing one or more manual annotations on single frames of video. The system then attempts to extend the user annotations into tracklets (partial track segments), filling in any sections of the data that are not completely annotated. To organize this process, the system maintains a prioritized list of annotation jobs and dynamically assigns these jobs to tracking agents (computer processes that perform tracking). A single user can trigger a larger number of parallel tracking processes. As the system generates tracklets, the user may shift to a verification and correction role. To support this role, the system supports interactive visualization of tracklets as they are being generated tightly integrated with the ability to issue corrections or additional annotations. With TrackMarks, users can fluidly interleave annotation, verification, and correction.

The development of TrackMarks was motivated by the video analysis challenges posed by the *Human Speechome Project* (HSP)[8]. The project is an attempt to study child language development through the use of very large collections of longitudinal, densely-sampled, audio-video recordings. In a pilot data collection, 11 ceiling-mounted fish-eye lens mega-pixel cameras and 14 boundary-layer microphones were installed in the home of a child. From the child's birth to three years of age, approximately 90,000 hours of 14 frames-per-second video recordings were collected, capturing roughly 70% of the child's waking experience at home during this period. In theory, given such a corpus, a scientific researcher should be able find and analyze events of interest to identify patterns of physical and social activity, compute aggregate statistics on aspects of behavior, or validate computational models of behavior and development. All of these tasks pose significant problems involving audio-video indexing, retrieval, and analysis. TrackMarks is an attempt to dramatically reduce the cost of preprocessing such video collections so that they may be used for scientific investigation.

There are existing systems that perform video annotation in a collaborative manner. Perhaps the one most similar to our own is described by Agarwala et al. in [1]. They describe a system for *rotoscoping* in which the user annotates object contours for two keyframes, the system interpolates the contours for the frames in between, after which the user can review and correct the generated annotations. This system provides a more simple interaction in which the system performs one tracking job at a time and the user reviews the results when the job is completed. In contrast, TrackMarks focuses on running multiple trackers at once and enabling the user to interact with the tracking processes while they are running.

Other approaches to tracking have included "two-stage tracking" systems, where the first stage involves the generation of high confidence tracklets, and the second stage determines how to aggregate the tracklets into longer tracks. In [9], the tracklets are combined automatically with a global optimization algorithm. In [5], Ivanov et al. describe a system that performs tracklet aggregation interactively with a human operator, which they refer to as "human guided tracking." In this system, tracklets are first generated from motion sensor data and the system aggregates the tracklets from multiple sensors as fully as it can. When the system cannot identify the correct tracklet sequence with sufficient confidence, such as when two objects come too close together to be disambiguated from the motion data, it presents the user with relevant video recordings and the user indicates which tracklets belong to a given target. While these systems may require far less manual labor than TrackMarks, they rely more on automatic tracking and require high confidence tracklets. TrackMarks has been developed to address worst case scenarios, in which sections of data may need to be annotated precisely, frame-by-frame. The more automatic systems still inform the future directions of TrackMarks, and possibilities for combining more automatic approaches will be discussed in Section 4.1.

2. THE TRACKMARKS SYSTEM

2.1 Design Goals

The purpose of TrackMarks is to identify and track multiple people from recordings taken from multiple cameras placed in connected visual regions, but was also designed around several additional objectives:

- Make annotation as efficient as possible while allowing the operator to achieve an arbitrary level of accuracy.
- Annotate camera handovers (when a person moves out of view of one camera and into view of another), occlusions, and absences. (*Absent* events are defined as cases in which a known target is not in view of any camera.)
- Provide an interface that is responsive enough to support fluid interaction between the operator and system. It is important that the automatic processes not impede the operator's ability to navigate and annotate the video. This requires that the response time for frequent operations remain under a few hundred milliseconds.
- Management for video collections larger than 100 TB. This affects database management, but also, for large corpora, it is unlikely that a present day computer will be able to access all of the data locally. When video data must be accessed over a network, it poses additional problems in maintaining responsiveness.

2.2 User Interface Overview

Figure 1 shows a screenshot of the TrackMarks interface. The top, middle panel shows a typical frame of video in full mega-pixel resolution. In this image, two bounding boxes have been superimposed on the video that indicate the positions of a child and adult. The colors of the boxes indicate the identity of each person. Video navigation is performed primarily with a jog-shuttle controller.

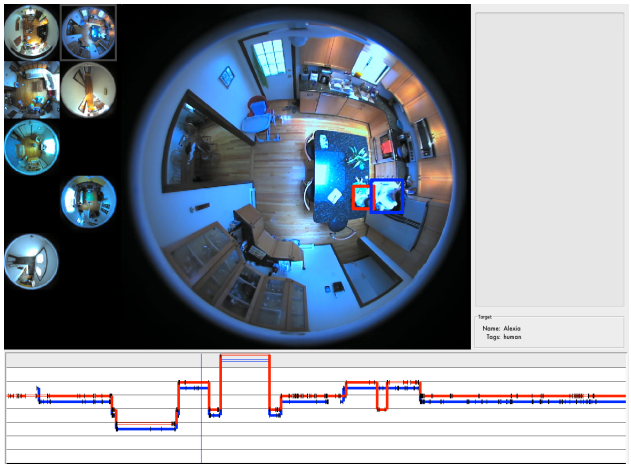


Figure 1: TrackMarks Interface

The top, left panel displays video thumbnails from the other cameras in the house. The user selects the video stream to view by clicking one of these thumbnails.

The bottom panel shows a timeline visualization of the annotations that resembles a subway map. This component summarizes the annotations that have been made, providing the user with a method of identifying and accessing portions of the data that require annotation. The horizontal axis represents time, which consists of approximately 30 minutes of data in this example. The blue, vertical bar indicates the user's position in the video stream. The timeline is divided into horizontal "channels," each representing one camera, demarcated by the thin black lines. The top channel, colored gray, represents the "absent" channel that is used to indicate that a target is not present in any of recordings. Finally, the thick, colored lines represent the tracklets. As with the bounding boxes superimposed on the video, the tracklets are colored to indicate person identity. The vertical placement of the tracklet indicates the channel, so when a tracklet line makes a vertical jump to another channel, it indicates that the target moved to a different camera at that place in time. As the system generates annotations, the timeline map adds or extends these tracklet lines and the user can monitor the progress of the system. Note that each bounding box shown on the video frame corresponds to a thin slice from one of the tracklet lines on the timeline view.

2.3 Track Representation

Track data is represented in a hierarchical structure with three levels: track points, track segments, and tracklets. At the lowest level, *track points* correspond to an annotation associated with a single frame of video. For the instance of the system described in this paper, all track points consist of bounding boxes. Track points are grouped into *track segments*, which represent a set of track points for a contiguous sequence of video frames from a single camera. Track segments may also indicate that a target is *occluded* or *absent* for an interval of time, and that no track points are available. Adjoining track segments are grouped into *tracklets*. Tracklets specify the identity of the target and combine the track data for that target across multiple cameras for a continuous time interval.

Several constraints are placed on the track data to simplify the system. First, only one annotation may be created for a given target in a given time frame. It is not possible to indicate that a target simultaneously occupies more than one location or camera, and multiple tracklets for a given target may not overlap. This precludes the use of multiple hypothesis tracking algorithms or annotating multiple views of the same object captured by different cameras, but greatly simplifies interaction with the system because the user does not need to review multiple video streams when annotating a given target.

Second, each tracklet has a single key point that is usually a track point created by the user. The tracklet *originates* from the key point, and extends forward and backward in time from that point. The purpose of the key point is to simplify tracklet editing. When deleting a track point from a tracklet, if the track point is defined after the key point, it is assumed that all of the tracklet defined after the deleted point is no longer valid and the right side of the tracklet is trimmed.

2.4 Annotation Process

This section outlines the annotation process from the view of the human annotator. To begin the process, the user selects an *assignment* to work on, where the assignment defines an objective for the user and a portion of data to process. The user browses the video and locates a target. Target identification is performed manually. The jog-shuttle controller used to navigate the video has nine buttons that are mapped to the most frequently occurring targets. The user may quickly select the identity by pressing the corresponding target button, or, more slowly, may evoke a popup menu that contains a complete list of targets as well as an option to define new targets. Given the overhead position of the cameras, it is sometimes necessary to browse through a portion of the data before an identification may be made. After identification, the user uses a mouse to draw a bounding box that roughly encompasses the object as it appears in the video. If the annotation appears correct, the user commits the annotation.

When the user commits an annotation, it creates a new track point as well and a new tracklet that consists of only that point. By default, TrackMarks automatically attempts to extend the tracklet bidirectionally. This process is described in Section 2.5. Camera handover is performed manually, and the user must create annotations at time frames in which a target enters or leaves a room. Usually, when annotating a target entering a room, the user defines a tracklet that should be extended forward in time, but should not be extended backward because the target will no longer be there. For this reason, it is also possible for the user to specify that a tracklet be extended only forward, only backward, or not at all.

While tracking is being performed, the operating may skim the video stream and verify the generated annotations. If a mistake is found, the user may click on the incorrect track point and delete it, causing the tracklet to be trimmed back to that point, or may choose to delete the tracklet altogether. More commonly, the user may draw a new bounding box around the target and commit the correction, causing the incorrect annotation to be deleted and restarting the tracking process for the target at that frame.

In addition to making position annotations, the user may indicate that a target is *occluded* or *absent*. Occlusions and

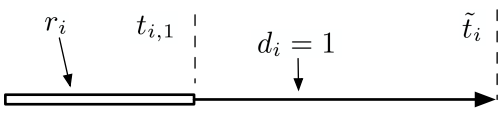
absences are indicated in an identical process, and both are referred to as *occlusion annotations*. Unlike the position annotations that are associated with a single frame, occlusion annotations may cover an arbitrary period of video. Rather than require the user find both the beginning and end of the annotation, which might require searching through a great deal of video and disrupt the user's workflow, the user marks each end of the annotation separately. Assuming that the user is navigating forward through a video stream and identifies the first time frame where a target becomes occluded, the annotator selects the target identity and presses a button on the jog-shuttle controller that indicates the selected target is occluded from that frame onward. This creates a new tracklet that starts from the user's time frame and extends as far as possible without overlapping existing annotations for the same target. When the user reaches the later time frame where the target becomes unoccluded, he can create a new annotation at that frame, effectively trimming the original occlusion annotation. In the timeline view shown in Figure 1, occlusion tracklets are distinguished from normal tracklets by using hollow lines.

2.5 Tracking Agents

While there are many possibilities for improving annotation efficiency through faster hardware or improved tracking algorithms, our work focuses on improving performance through structured collaboration. Having an efficient tracker implementation is still important, but large gains can be had by scheduling the tracking processes intelligently to minimize redundant operations, to provide rapid feedback to the user so that errors are corrected quickly, and to prevent the sizable computational resources required for object tracking from interfering with the interface. The key optimization made by TrackMarks, then, is the tracking agent subsystem that manages and executes tracking tasks.

The tracking process may be broken into four steps, in which tracking *jobs* are defined, prioritized, assigned, and executed. This is not a linear process; jobs may be revised or cancelled at any time. The tracking agent subsystem that handles this process consists of four levels: a *job creator*, a *job delegator*, *job executors*, and trackers.

The job creator receives all requests to expand tracklets and defines a job for each request. For convenience, the backward time direction is referred to as *left*, and the forward as *right*. Each job is associated with a tracklet r_i, where the purpose of the job is to add annotations to expand the tracklet in either the left or right direction, $d_i \in \{0, 1\}$. The left and right edges of the tracklet occur at times $t_{i,0}$ and $t_{i,1}$, respectively, and the job is complete when the edge t_{i,d_i} is extended to a goal time, $t_{i,d_i} = \tilde{t}_i$.



The main parameter to compute is the goal time. $\tilde{t}_i$ is selected such that the tracklet is extended as far as possible without overlapping existing tracklets with the same target. Assuming tracklet r_i is being extended to the right $d_i = 1$, the job creator finds the first tracklet r_j that exists to the right of r_i and shares the same target. If there is no such r_j, then $\tilde{t}_i$ may be set to the end of the available video stream. If r_j does exist, then $\tilde{t}_i$ is set to the left edge of that tracklet $\tilde{t}_i \leftarrow t_{j,0}$.

In the case where r_j is being extended backwards by another job, then the goal time of both jobs is set to the time half way between r_i and r_j, splitting the remaining load between the two jobs, $\tilde{t}_i, \tilde{t}_j \leftarrow 0.5(t_{i,1} + t_{j,0})$.

Tracklets may be created and altered while tracking occurs. The job creator is notified of all such events and recomputes the goal times $\tilde{t}$ for all jobs that might be affected. If the operator deletes an annotation that was created by a job that is still being processed, that job has failed and is immediately terminated.

All existing jobs are given to the job delegator, which schedules jobs for execution. In the case of TrackMarks, the time required to perform person tracking is a small fraction of the time required for retrieving and decoding the video. Subsequently, the greatest gains in efficiency result from tracking as many targets as possible for each frame of video retrieved. The job delegator exploits this by mapping all jobs into *job sets* that may be executed simultaneously by processing a single contiguous segment of video, prioritizing each job set, and assigning the highest priority sets to the available executors.

The process of grouping jobs into sets can be formulated as a constraint satisfaction problem. Each job b_i is associated with a segment of video to process, defined by a camera c_i and time interval r_i. For forward jobs with $d_i = 1$, the interval is $s_i = [t_{i,r}, \tilde{t}_i]$. For jobs with $d_i = -1, s_i = [\tilde{t}_i, t_{i,0}]$. Each job set, B, is must meet three constraints:

- Forward and backward tracking jobs cannot be combined, nor can jobs that access video streams generated from different cameras. $\forall_{b_i \in B} \forall_{b_j \in B} d_i = d_j \wedge c_i = c_j$.
- Jobs with non overlapping intervals cannot be processed together. The union of time intervals for all jobs in a set $\cup_{i, b_i \in B} s_i$ must be interval.
- A job cannot belong to a set if the tracklet edge being extended by that job is too distant from the other jobs in the set. $\forall_{b_i \in B} \exists_{b_j \in B} |t_i^r - t_j^r| < \tau$, where τ is a chosen parameter.

When a job b_i is first created, the delegator attempts to locate an existing job set, B, to which the job may be added without violating any constraints. If no such set can be found, a new set is created containing only one job. Each time a job as added or otherwise altered, the constraints are reapplied, causing job sets to be split and joined as needed. The algorithm used to solve this constraint problem involves implementation details and is of less interest than the formulation of the problem itself.

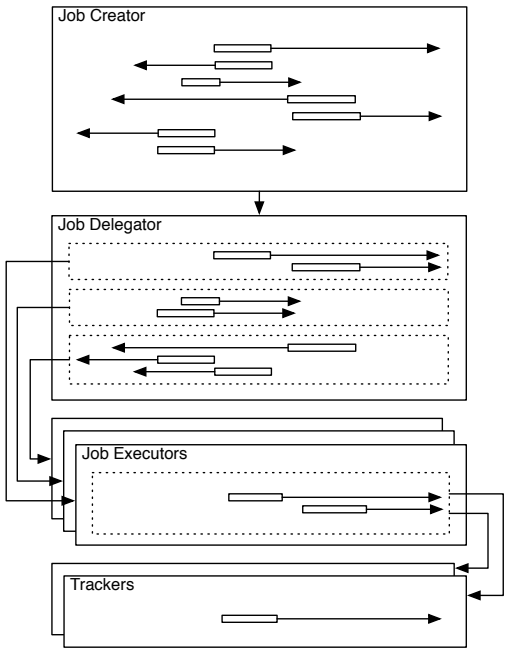


Figure 2: Tracking Agent Subsystem. The job creator manages individual tracking jobs. The delegator organizes the jobs into sets. Each executor processes one set of jobs, feeding video frames to one tracker per job.

Each time a job set is altered, the job delegator recomputes a priority for that set. The priority is based first on processing jobs sets such that they are likely to reach another job set, allowing them to merge and thus reduce computational load. Second, higher priority is given to job sets that are processing data at earlier time frames, encouraging tracking to be performed from left to right. This criteria helps to reduce the fragmentation of tracklets, and reduces the amount of time the user expends reviewing video. Other criteria might also be incorporated at this stage. Giving greater priority to the recency of the jobs might help to place trackers near the locations that the user is annotating, improving the perceived responsiveness of the system.

After the job sets are prioritized, the top n sets are assigned to job executors. The parameter n thus determines the number of tracking threads running at any point in time, and the portion of resources allocated to tracking. Each executor runs in an isolated thread, executing all jobs in the set $b_i \in B$ simultaneously. The executor initializes a tracker for each job. Job sets may be altered while the executor is running and not all jobs in the set are necessarily aligned. For forward tracking job sets, $\forall_{b_i \in B} d_i = 1$, the executor always retrieves the earliest frame of video to be processed by any job, $t_f = min_{b_i \in B} t_{i,1}$, passing that frame to each tracker that has not already processed it. The resulting behavior is that the executor may sometimes "jump backward" in the video stream and reprocess a segment of video until the earliest job catches up to the others.

The trackers are then conceived as passive components that produce one track point that indicates the location of one target for each video frame provided. The position tracker used is built on the *mean-shift* tracking algorithm [4]. Mean-shift tracking is performed by adjusting the size and location of a bounding box to maintain a consistent distribution of color features within the box. This algorithm was chosen partially for its relatively fast speed. However, it was also chosen over motion-based algorithms because it was predicted that people in a home environment would spend much more time sitting in place, as compared to people in a public space that might spend more time traveling. Additionally, when tracking a child being held by a caregiver, the two targets may be separated more reliably by color than by motion. While the implementation used relies more strongly on color features, motion features were not discounted entirely, and the tracking algorithm does incorporate a fast foreground-background segmentation step to improve performance for targets in motion.

The end result of the tracking agent system is a structured but fluid interaction between the user and system. As the user creates annotations, TrackMarks propagates them to create a set of tracklets that cover the full range of data for all targets. As it performs tracking, TrackMarks attempts to reduce the fragmentation of the data, combines tracking jobs when possible to greatly reduce the computation time required, and continuously indicates tracking progress in the user interface. When the user makes a correction, the tracking agents remove any jobs that are no longer useful and creates new jobs to propagate the correction. The tracker agents afford the user flexibility in annotating the data. In difficult cases where the tracking fails often, the user primarily focus on reviewing and correcting annotations. In other instances, the user may skim through the video and provide a few annotations, and return later to review the results. In the worst cases in which tracking fails completely, the user still has the ability to annotate manually.

3. EVALUATION

To test the efficiency of TrackMarks, three hours of multi channel video recordings were selected from the speechome corpus for transcription. The selected data was broken into six, 30 minute *assignments*. To select assignments of interest that contained high levels of activity, the data was selected from the 681 hours of data for which speech transcripts were available, allowing the activity levels of the assignments to be estimated by examining the amount of speech activity. The data from this subset was collected over a 16 month period, during which a child and family was recorded while the child was between the age of 9–24 months. The 1362 assignments were ordered in a list according to a combination of activity level and distribution of collection dates, and a sequence of six assignments was selected from near the top of that list (positions 18–24). Figure 3 shows several video frames taken from the transcribed data.

Annotators were given the same objective for all assignments: to fully annotate the position and identity of the child for the entire assignment, and to annotate all other persons within sight of the child's position, including persons visible to the child through open doorways. The annotators were instructed to fix all incorrect annotations. Annotations (drawn as bounding boxes) were deemed *correct* if the box contained at least half of the target object, and that it would not be possible to reduce the area of the bounding box by half without also removing part of the object. Targets were marked *occluded* if less than half of the target object was vis-

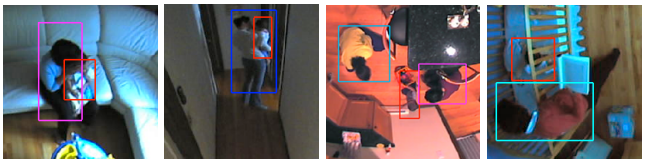


Figure 3: Example Annotations

	mean	min	max	std
A1 time (minutes)	64.1	31.0	106	27.1
A2 time (minutes)	44.2	22.5	62.7	14.8
Strict IAA MOTA	88.1%	77.4%	98.7%	7.11%
IAA MOTA	97.3%	86.3%	99.9%	5.40%
IAA MOTP	12.3px	8.33px	17.2px	3.52px
Object Count	47,482	15,381	71,320	20,277
Objects Per Frame	1.88	0.610	2.83	0.805
Manual Annotations	316	181	466	117
Forced Errors	65.7	38.0	90.0	18.9
Unforced Errors	250	143	376	98.4

Table 1: Annotation Trial Results

ible, and *absent* if the target was not present in any rooms being recorded.

The evaluation was performed by two annotators. Annotator $A1$ (first author) had previously annotated approximately five hours of video with TrackMarks. Annotator $A2$ was given two warmup assignments that are omitted from the evaluation. Results are shown in Table 1.

The first two rows present the annotation time required by both annotators, A1 and A2. The speed of the annotators fell between 28.3% and 133% of the speed required to annotate the data in realtime, with a overall mean of 55.4%. That is, 108 minutes of labor was required on average to annotate 60 minutes of recordings.

Rows three through five present the inter-annotator agreement computed with the CLEAR MOT metrics[10]. The MOT metrics provide a standardized method for computing the accuracy (MOTA) and precision (MOTP) of multi object tracking systems. The MOTA score indicates the percent of correct matches, taking into account the number of misses, false positives, and target mismatches. The MOTP score indicates the average error for all correct matches. These metrics are designed to be generalized to a wide range of tracking tasks, and require an application specific error function to compute the error between two data points in a given time frame. The error function used for this evaluation considers two data points, bounding boxes $p1$ and $p2$, to be a match if $p1$ intersects $p2$. If $p1$ and $p2$ match, then the error between them is defined as the Euclidean distance between their respective centroids. In standard applications of the MOT metrics, one set of tracks must be provided as ground truth and another set for evaluation. To adapt the metrics for inter-annotator agreement, we compute the scores twice, each time using the data provided from a different annotator as the ground truth, and average the results. Additionally, because only the *child* target was required to be annotated completely and there was greater flexibility for annotating the other targets, we compute inter-annotator agreement only on the tracks associated with the child target.

The *Strict IAA MOTA* figures, computed as defined above, show the accuracy of one annotator compare to the other, where the average was 88.1%. The primary source of disagreement was in the determination of whether a target was *visible*, *occluded*, or *absent*. While the evaluation task defined that an object could be marked *occluded* if less than half of the object was visible, this rule proved difficult to apply in practice. For example, for approximately 15 minutes of the evaluation data, the child target was in a crib and was only visible through the slats of the crib. Other portions of data contained the child repeatedly covering and uncovering his head with a blanket in both walking and lying positions, making it unclear whether the blanket should be considered an occlusion or clothing. Most of all, if the automatic tracking was performing well despite significant occlusions, annotators did not consistently mark the occlusion since that would mean deleting otherwise valid position annotations. Another source of disagreement was in determining the exact point of camera handoff, since the system forces the annotator to associate a given target with at most one camera at one time.

To get another sense of accuracy, the second *IAA MOTA* figures were computed on only the time frames where both annotators agreed that the child target was visible. Furthermore, the accuracy was not computed for time frames that occurred within five seconds of a camera handover. The resulting accuracy of 97.3% shows that the majority of the disagreement was caused by the added ambiguity of explicitly marking occlusions and camera handovers. On average, this processed culled 39% of the annotations, which might be expected for a child target that is often being held or occluded by the environment, but indicates that more stringent guidelines are needed to annotate multi-camera video consistently.

The *IAA MOTP* shows that the average distance between annotation centroids was 12.3 pixels; approximately 1.3% of the 960 by 960 pixel video frames. Because the video was recorded with a fish-eye lens, the distance represented by 12.3 pixels varies considerably depending on the pixel position, room geometry, and pose of the target. Because precision is computed only between correct matches, the precision scores computed for the strict agreement had very small deviation from those reported.

Referring back to Table 1, the bottom five rows contain statistics for the assignment tasks, all of which are averaged across the results from both annotators. *Object Count* refers to the total number of objects annotated for the entire assignment, not counting *occluded* and *absent* annotations. *Objects Per Frame* indicates the average number of objects occurring for each time frame. Because the cameras used were not synchronized at a per frame level, the number of time frames in one half hour was approximated as 25,000.

Manual Annotations indicates the total number of manual annotations created by the annotator for an assignment. Each manual annotation may be thought of as system errors, in which the system was unable to produce a correct annotation and required human intervention. We define two kinds of errors: *forced* and *unforced*. Forced errors occur on camera handovers and object transitions between *visible*, *occluded*, and *absent*. These errors are considered forced because the system is not designed to identify these events, and even with perfect tracking, they would still require manual intervention. The number of forced errors, then, is the theoretical minimum number of manual annotations required

if perfect, single-camera tracking were available. Unforced errors occur when the automatic tracking fails on a frame containing a visible target and requires manual correction. These are errors that might be reduced with a better tracker. In this evaluation, annotators were instructed to achieve accuracy and speed, not to minimize the number of annotations made. Subsequently, the number of unforced errors can only be roughly estimated by subtracting the number of forced errors from the total number of annotations. Although the unforced error figures are less certain than the others, they suggest that the amount of human intervention could be greatly reduced by more accurate tracking.

In the second evaluation, we compare the efficiency of TrackMarks to a fully manual approach. A simple system was constructed that provided an extremely simplified procedure for manual annotation. The system presents one frame of video to the annotator, the annotator locates and draws a bounding box around the child target, and the system presents the next frame of video. The video frames are presented in sequence, so the primary limit was the speed of physically drawing the bounding box. In one ten minute trial, an annotator (first author), working at a rapid but comfortable pace, produced annotations for 846 objects. In the evaluation of TrackMarks, each 30 minute assignment contained an average of 47,482 objects, suggesting that the minimum time needed to annotate such an assignment manually would be 561 minutes; about 10.4 times as much human labor as required by TrackMarks. While there are many possibilities for expediting manual annotation, it is an inherently tedious task, both physically and mentally. A more comprehensive analysis would likely need to include estimates for the amount of rest annotators require between sessions, and adjustments to the pay needed to retain annotators. Even in the absence of such an evaluation, we strongly believe that manual approaches will be limited to a fraction of the efficiency of human-machine collaboration, and that the difference will grow rapidly as technologies improve.

Last, we compare the accuracy of TrackMarks to that of a fully automated tracker. The automatic tracker evaluated was implemented with SwisTrack[7], an open source software package containing many commonly used tracker components. Out of several configurations, we achieved the best results by detecting objects using motion templates[3] and combining the objects into tracks with a nearest neighbor tracker. 140 hours of single channel video taken from the speechome corpus was processed with the automatic tracker. The half-hour block that contained the greatest number of identified objects was selected and annotated using TrackMarks. The MOT metrics were again used to determine accuracy and precision. In this evaluation, the TrackMarks annotations were used as ground truth. The annotations produced by TrackMarks consist of bounding boxes m_i around an object, while the tracks from the automatic tracker consist of points n_i that represent object centroids. Because of this discrepancy, an error function was used that differs from that used in the first evaluation. If m_i is a truth annotation, and n_i is a test annotation occurring in the same time frame, then n_i is considered a match to m_i if n_i is contained by m_i scaled by 1.5 about its center, and the error between m_i and n_i is the Euclidean distance between n_i and the center of m_i. The performance of the automatic tracker is shown in Table 2.

MOTA	48.0%
MOTP	32.8 px
Objects	67,231
Matches	37,323
Misses	29,908
False Positives	4,634
Mismatches	470

Table 2: Automatic Tracker Performance

The performance of the automatic tracker was comparable to other multi-object tracking systems applied to similar tasks[10]. The overwhelming cause of errors was *misses*, at 29,908, followed by *false positives* at 4,634, and then *mismatches* at 470, where *mismatches* were defined as instances where the generated tracklets switched from following one target to another. The MOTA achieved by the automatic tracker, 48.0%, is 49.3% lower than the inter-annotator MOTA achieved by TrackMarks, and 40.1% lower than the strict inter-annotator MOTA. The average error between matches was 32.8 pixels, compared to 12.3 pixels for the inter-annotator precision of TrackMarks.

4. DISCUSSION

For tracking tasks that require low to moderate accuracy, automatic tracking is still the most cost effective solution. However, high error rates severely limit the usefulness and possible applications for video analysis. On the other extreme, manual annotation requires a great deal of labor, limiting this approach to projects that can absorb the high cost of human annotators.

We have established TrackMarks as a viable approach to annotation that achieves a balance between cost and accuracy that creates new possibilities for video analysis. TrackMarks provides a level of accuracy far closer to manual annotation than automatic, while providing efficiency that is an order of magnitude greater than fully manual processing. Furthermore, TrackMarks makes it efficient to explicitly mark object occlusions and absences, providing an additional layer of information that can be used for analysis when more descriptive annotations cannot be created. Indeed, this feature was an important objective for the behavioral analysis research for which TrackMarks was developed.

While we have focused on person tracking as an example of video processing that can benefit from human-machine collaboration, we believe this approach will become increasingly more valuable as video analysis becomes feasible for a growing number of applications. Even as object tracking algorithms improve, new applications will arise that will make use of increasingly detailed information, such as head or hand pose. Researchers and analysts will require human vision faculties to process video for the foreseeable future, which may establish an important niche for human-machine collaborative systems like TrackMarks.

4.1 Future Directions

This research has focused more on the interaction between the computer and human, and has tried to maximize the usefulness of a relatively modest object tracker. The most obvious improvements involve increasing the accuracy and speed of the tracker processes to reduce the time and ef-

fort required of the human operator. Trackers that produce reliable confidence scores introduce a whole range of possibilities for more intelligent tracklet editing. Identifying segments of data that have a high probability of errors provides a way to bring the human's attention where it is needed most, reducing the time spent on reviewing data. Confidence also provides methods for automatically joining or splitting tracklets, In the current implementation, removing one incorrect annotation causes the entire tracklet to be trimmed to that point, the assumption being that it is more efficient to reprocess the segment of video than to require the user to find all subsequent errors in the tracklet. In many cases, cutting the tracklet based on confidence may prevent redundant processing.

However, there may be much larger gains to be found by focusing on the higher level tracking logic. This includes introducing automatic person detection, enabling the system to perform more tracking operations without manual initialization. In the case where the camera positions are well known, this leads naturally to automatic camera handover. As tracking performance improves, much of the processing could be performed offline, before the human annotator looks at the data, similar to the two-level tracking systems referred to in Section 1. In the case where the video contains longitudinal recordings of a limited number of targets, more detailed target models may be learned that help classify identity based on both appearance and behavioral routines.

5. ACKNOWLEDGMENTS

We thank Yuri Ivanov for his ideas and his help in framing this paper; Kleovoulos Tsourides for lending his time to annotate data; and Stefanie Tellex for providing an object tracker and data. This research was funded in part by a grant from the U.S. Office of Naval Research.

6. REFERENCES

[1] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004.

[2] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *J. Image Video Process.*, 2008(3):1–10, 2008.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[4] V. Comaniciu and P. Meer. Kernel-based object tracking, 2003.

[5] Y. Ivanov, A. Sorokin, C. Wren, and I. Kaur. Tracking people in mixed modality systems. In *SPIE Conference on Visual Communications and Image Processing (VCIP)*, volume 6508 of *Proc. of IEEE*. SPIE, January 2007.

[6] J. C. R. Licklider. Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1:4–11, March 1960.

[7] T. Lochmatter, P. Roduit, C. Cianci, N. Correll, J. Jacot, and A. Martinoli. SwisTrack - A Flexible Open Source Tracking Software for Multi-Agent Systems. In *IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008)*, pages 4004–4010. IEEE, 2008.

[8] D. Roy, R. Patel, P. DeCamp, R. Kubat, M. Fleischman, B. Roy, N. Mavridis, S. Tellex, A. Salata, J. Guinness, M. Levit, and P. Gorniak. The human speechome project. In P. V. et al., editor, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, pages 192–196. Springer, 2006.

[9] V. Singh, B. Wu, and R. Nevatia. Pedestrian tracking by associating tracklets using detection residuals. In *Motion and Video Computing*, pages 1–8, January 2008.

[10] R. Stiefelhagen, K. Bernardin, R. Bowers, R. T. Rose, M. Michel, and J. Garofolo. *The CLEAR 2007 Evaluation*. Springer-Verlag, Berlin, Heidelberg, 2008.