Int J Comput Vis
DOI 10.1007/s11263-012-0564-1

Efficiently Scaling up Crowdsourced Video Annotation

A Set of Best Practices for High Quality, Economical Video Labeling

Carl Vondrick · Donald Patterson · Deva Ramanan

Received: 31 October 2011 / Accepted: 20 August 2012

Abstract We present an extensive three year study on economically annotating video with crowdsourced marketplaces. Our public framework has annotated thousands of real world videos, including massive data sets unprecedented for their size, complexity, and cost. To accomplish this, we designed a state-of-the-art video annotation user interface and demonstrate that, despite common intuition, many contemporary interfaces are sub-optimal. We present several user studies that evaluate different aspects of our system and demonstrate that minimizing the cognitive load of the user is crucial when designing an annotation platform. We then deploy this interface on Amazon Mechanical Turk and discover expert and talented workers who are capable of annotating difficult videos with dense and closely cropped labels. We argue that video annotation requires specialized skill; most workers are poor annotators, mandating robust quality control protocols. We show that traditional crowdsourced micro-tasks are not suitable for video annotation and instead demonstrate that deploying time-consuming macro-tasks on MTurk is effective. Finally, we show that by extracting pixel-based features from manually labeled key frames, we are able to leverage more sophisticated interpolation strategies to maximize performance given a fixed budget. We validate the power of our framework on difficult, real-world data sets and we demonstrate an inherent trade-off between the mix of human and cloud computing used vs. the accuracy and cost of the labeling. We further introduce a novel, cost-based evaluation criteria that compares vision algorithms by the budget required to achieve an acceptable performance. We hope our findings will spur innovation in the creation of massive labeled video data sets and enable novel data-driven computer vision applications.

Keywords Video annotation · Large scale annotation · Data sets · Mechanical Turk · Crowdsource marketplaces · Tracking

1 Introduction

Sorokin and Forsyth (2008) made the influential observation that *image* labeling can be crowdsourced at low costs through platforms such as Amazon's Mechanical Turk (MTurk). This approach has revolutionized *static* data annotation in vision, and enabled almost all large-scale image data sets collected since then to be labeled (Deng et al. 2009; Russell et al. 2008; Kumar et al. 2009). Contemporary computer vision research has subsequently demonstrated the value of massive data sets of labeled images such as the results from ImageNet (Deng et al. 2009), PASCAL (Everingham et al. 2010), LabelMe (Russell et al. 2008), SUN (Xiao et al. 2010), and TinyImages (Torralba et al. 2008).

The same does not hold true for video despite a corresponding abundance of data, such as that from web-cams

A preliminary version of this work appeared in ECCV 2010 by Vondrick et al.

C. Vondrick (✉) · D. Ramanan
Department of Computer Science, UC Irvine, Irvine, USA
e-mail: vondrick@mit.edu

D. Ramanan
e-mail: dramanan@ics.uci.edu

C. Vondrick
Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, USA

D. Patterson
Department of Informatics, UC Irvine, Irvine, USA
e-mail: djp3@ics.uci.edu

Published online: 05 September 2012

and public-domain archival footage (Kahle 2010). We believe that this is due to the *dynamic* nature of video data which makes frame by frame labeling necessary but inefficient for manual labor. Inspired by popular successes such as Yuen et al. (2009), Vijayanarasimhan and Grauman (2009), Liu et al. (2008), we focus on cost effective and high quality video annotation with MTurk. We show the results of *three years of experiments* and experience in annotating massive videos unprecedented for their size and complexity, with some data sets consisting of millions of frames, costing *tens of thousands of dollars*, and requiring up to *a year of continuous work* to annotate. This extensive study has resulted in our release of VATIC (Video Annotation Tool from Irvine, California), an open platform for monetized, high quality, crowdsource video labeling.

The contributions made in this paper are motivated by our desire to uncover best-practices for monetized crowdsourced video labeling. In the remainder of this paper, we describe our video annotation tool in detail:

In Sect. 2, we briefly review related work in designing image and video annotation tools.

In Sect. 3, we present insights into the design of a user-interface in which workers track objects through a continuous video shot (to solve the problem in Fig. 1). To support our claims, we present user studies that demonstrate contemporary annotation software is suboptimal. We have found that video annotation is considerably more complex than image annotation, likely due to the fact that temporal data is difficult to visualize and edit.

In Sect. 4, we describe how to best use crowdsourcing to annotate videos. In order to collect high-quality annotations, we find it crucial to validate good workers and turn away the majority of MTurk workers. In this sense, we do not use MTurk directly as a crowdsourced platform, but rather as a market to identify reliable workers.

Fig. 1 An example of the difficult problem that our interactive system addresses. The *red boxed* player becomes totally occluded while many players quickly change pose from standing to a prone position. The referees commonly enter and leave the scene. The camera is not stationary. The ball exists in the pile of people, but even a state-of-the-art vision algorithm is unable to determine its position

In Sect. 5, we analyze trade-offs particular to balancing computer and human effort in video annotation by extending work that minimized labeling cost only along the dimension of human effort (Vijayanarasimhan and Grauman 2009; Vijayanarasimhan et al. 2010). Although the "Turk philosophy" is to completely replace difficult computer tasks (such as video labeling) with human effort, this is clearly *not* efficient given the redundancy of video. In contrast to LabelMe video (Yuen et al. 2009), we show that one can interpolate *nonlinear* least-cost paths with efficient dynamic programming algorithms based on image data and user annotated endpoints.

In Sect. 6, we analyze the total cost of labeling for various combinations of human workers and cloud-computing CPU cycles. We further demonstrate that our cost analysis can be used as a error metric for evaluating vision algorithms; rather than evaluating a tracker with disconnected measures such as time-to-failure, we evaluate trackers using the dollar amount in savings afforded when used in a monetized, interactive crowdsourced platform.

Our hope is that our discoveries will spur innovation in the creation of affordable, massive data sets of labeled video. To encourage this, our final contribution is the release of a simple, reusable, and open-source platform for research video labeling.[1]

2 Related Work

With the rising popularity and success of massive data sets in vision, the community has put considerable effort into designing efficient visual annotation tools. Deng et al. (2009) introduced a crowdsourced image annotation pipeline through ImageNet. Torralba et al. (2010) presented LabelMe as an open platform for dense polygon labeling on static images. Everingham et al. (2010) describe a high quality image collection strategy for the PASCAL VOC challenge. Von Ahn and Dabbish (2004) and Von Ahn et al. (2006) discovered that games with a purpose could be used to label images. Ramanan et al. (2007) show that exploiting temporal dependence in video can automatically build a data set of static faces. Welinder et al. (2010) propose a quality control mechanism for annotation on crowdsourced marketplaces. Vittayakorn and Hays (2011) describe quality control measures without collecting more data. Endres et al. (2010) study some of the challenges and benefits of building

[1] The software and data sets can be downloaded from our website at http://mit.edu/vondrick/vatic.

image datasets with humans in the loop. Yet, the same principles that assist and motivate users to annotate static images do not apply to dynamic videos.

Consequently, significant work has been completed in order to build specialized interfaces tailored for video annotation. Yuen et al. (2009) introduced LabelMe video, an online, web-based platform that is able to obtain high-quality video labels with arbitrary polygonal paths using homography preserving linear interpolation, and can generate complex event annotations between interacting objects. Mihalcik and Doermann (2003) describe ViPER, a flexible and extensible video annotation system optimized for spatial labeling. Huber (2011) designed a simplified interface for video annotation. Ali et al. (2011) present FlowBoost, a tool that can annotate videos from sparse set of key frame annotations. Agarwala et al. (2004) propose using a tracker as a more reliable, automatic labeling scheme compared to linear interpolation. Buchanan and Fitzgibbon (2006) discuss efficient data structures that enable interactive tracking for video annotation. Fisher (2004) discuss the labeling of human activities in videos. Smeaton et al. (2006) describe TRECVID, a large benchmark video database of annotated television programs. Laptev et al. (2008) further show that using Hollywood movie scripts can automatically annotate video data sets.

While these tools are effective at building large data sets, they are not necessarily economical. The state-of-the-art is optimized to obtain high quality labels, but as a practical matter, we must also be concerned with cost. In order to scale up to the next generation of data sets, we must build a system that can annotate high quality, massive videos without exhausting our community's funding and tiring our workers. In this paper, we propose such a system for large scale, high quality and economical video annotation platform.

3 User Interface

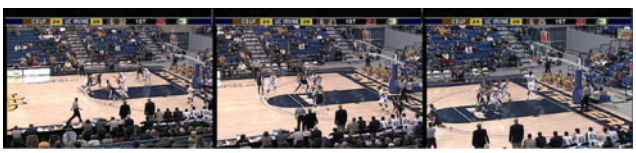
We wish to design an interface that allows workers to annotate every object of interest in a video. Our users should be able to both track objects, represented as spacetime trajectories of bounding boxes, as well as mark attributes, represented as discrete flags associated with a trajectory for a time interval. However, since humans have troubling visualizing space simultaneously with time, designing this graphical interface presents subtle, challenging problems that, if not properly addressed, make video annotation unnecessarily labor intensive. Despite this additional complexity, most contemporary video annotation interfaces are suboptimal because they assume that the same principles that help users annotate space can be applied to time. In this section, we present a more efficient interface for video annotation, shown in Fig. 3. We demonstrate, through extensive user studies, that popular design choices are sub-optimal and we show that, despite intuition, more constrained and simpler interfaces provide a superior annotation experience.

3.1 User Studies

Throughout this section, we evaluate different modes of annotation by conducting user studies. Since we reject the status quo in video annotation interface design, we completed the study to support our claims. In each study, we asked annotators to label videos shown in Fig. 2. We picked these videos because they each represent common problems in video annotation: the scripted video has large objects quickly moving with linear yet dynamic motion; the basketball video is extremely difficult due to small, similar looking objects with frequent occlusions and nonlinear motion; and the VIRAT video (Oh et al. 2011) has slow moving and stationary cars that follow linear paths.

We located research subjects both by hiring dedicated users and discovering capable workers on MTurk. We found subjects on MTurk by contacting our highest earning workers and offering them the opportunity to participate in our user study. Upon accepting our offer, they were redirected to a private website and required to read instructions before completing the study. We compensated MTurk workers at about $7.00 per hour. Our dedicated workers are experts in computer vision and were compensated fairly in accordance with standard university pay scales. In both cases, we asked annotators to label every video under varying conditions. We randomized the order that annotators labeled videos, a necessary step to reduce a learning and memorization bias; once

(a) Scripted



(b) Basketball

(c) VIRAT

Fig. 2 The three segments from our study that dedicated annotators labeled in order to compare fixed rate vs. user defined key frames. (**a**) Is a group of people quickly walking around in a complex manner. (**b**) Consists of a very difficult clip from the basketball game with ambiguous motion and nonlinear paths. (**c**) Is a parking lot with a couple of cars driving in the street and following linear paths

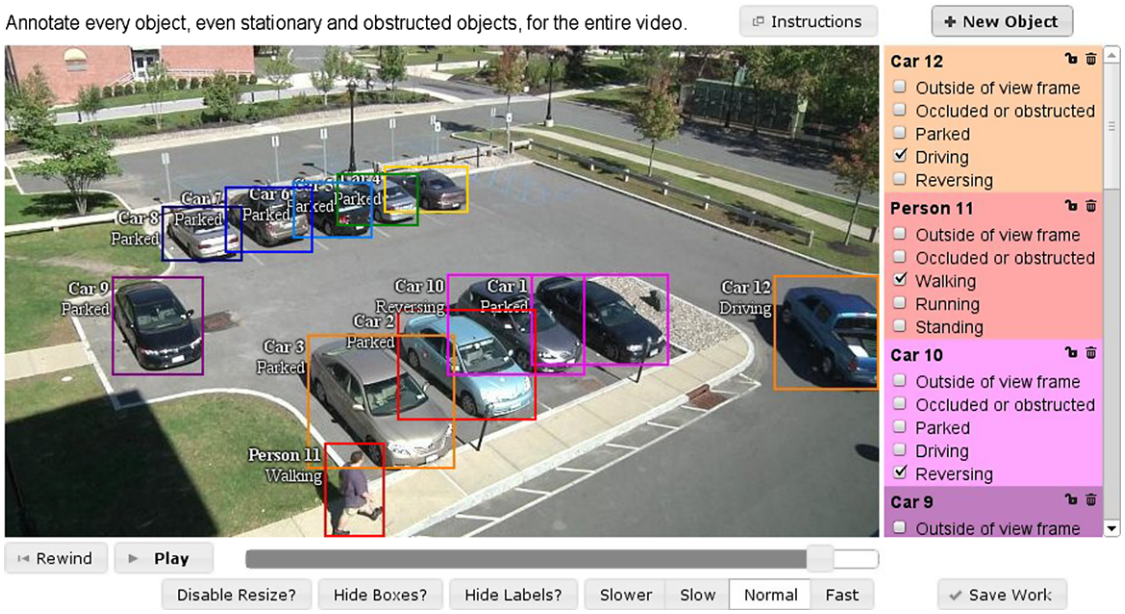


Fig. 3 Our online, interactive video annotation user interface. Users can play the video, draw bounding boxes around objects of interest, and track each object throughout its lifetime. Each object can have multiple attributes that further describe its actions. Workers can adjust the play back speed, seek throughout the timeline, and mark objects as occluded or off the screen. Since scenes quickly become cluttered, users can lock objects to prevent accidental modifications to their paths. Keyboard shortcuts are available

an annotator has wrestled with a video sequence, labeling it again becomes significantly easier. We then timed how long a user took to annotate each video to evaluate which modes are more efficient, and we also solicited qualitative feedback, such as "which interface felt faster?" or "which interface did you prefer?" Our results are surprising and demonstrate that contemporary video annotation platforms are sub-optimal.

3.2 Key Frame Schedules

Since manually labeling every frame in a video is clearly inefficient, our system requires users to annotate only a sparse set of key frames. We then use a variety of interpolation methods (discussed in Sect. 5) to predict the annotations on the remaining frames. In the following experiments, we use simple linear interpolation to validate aspects of our interface. Users are allowed to playback the video where they see the linearly-interpolated paths in real-time. As one might expect, the choice of key frames can have a significant impact on the user experience. The more key frames the user must label, the longer it takes for the user to annotate a video. Therefore, an optimal key frame strategy will minimize the number of key frames that a user must annotate. There are two primary types of key frames: fixed rate and user defined.

Fixed key-frames: A simple key frame schedule is to annotate each object on a fixed rate interval where the user labels every T frames. For difficult videos like a basketball game with dynamic motion, we require a high frequency of key frames, so we must set T to be low, while for easier videos like VIRAT, we can set T to be much higher. Indeed, fixed rate key frames allow the user to annotate video mechanically: once the video pauses, simply update the annotations. However, in order to capture high quality labels, the annotation frequency must be high enough to handle every situation. If an object has chaotic motion for only a few seconds but linear motion otherwise, T must be low enough to handle to the chaotic motion, resulting in wasted clicks during the linear motion.

User-defined keyframes: To overcome these issues, we could instead adopt a user defined key frame schedule. This approach would allow the user to pause the video on any frame and update annotations. When an object moves in a straight line (such as a car driving down the street), users can label the end points of the path and rely on linear interpolation to recover the remaining annotations. This allows the user to dynamically adjust the key frame frequency: for mostly stationary frames, perhaps only a few annotations are necessary, but for chaotic frames, the user can label frequently. Assuming the user can quickly pick intelligent key frames, this approach can lead to an optimal key frame schedule. Most contemporary video annotation systems such as LabelMe video or ViPER deploy user-defined key frames.

User-study results: Although most existing annotation systems employ user defined key-frames, our results demonstrate that *fixed rate key frames are significantly faster than user defined key frames*. Essentially, users are not accurate at

Table 1 A comparison of annotation time on fixed rate vs. user defined key frames. *Though almost all existing annotation systems employ user defined key-frames, our results demonstrate that fixed rate key frames are superior*. We asked dedicated workers to label the three segments shown in Fig. 2. Times are in seconds. Ratio is fixed over user. Averages are arithmetic mean for time, and geometric mean for ratios. Subjects with asterisks did the user defined keyframes second, so they may have memorized the video when doing user defined key frames. Notice that basketball speed is strongly correlated with the order of experiments. Our results on the scripted data are statistically significant (at a p-value of .04)

Subject	Scripted				Basketball				VIRAT			
	User	Fixed	Ratio	Saved	User	Fixed	Ratio	Saved	User	Fixed	Ratio	Saved
A	599	**463**	0.77	136	1,457	**1,323**	0.91	134	**220**	244	1.11	−24
B	653	**247**	0.38	406	4,555	**2,275**	0.50	2,280	**176**	**178**	1.03	−2
C	476	**275**	0.58	201	1,216	**830**	0.68	386	338	**215**	0.64	123
D	772	**432**	0.56	340	1,505	**1,497**	0.99	8	489	**302**	0.62	187
*E	605	**371**	0.61	234	**935**	1501	1.61	−566	269	**231**	0.85	38
*F	654	**472**	0.72	182	**1,672**	1,858	1.11	−186	372	**326**	0.87	46
*G	235	**193**	0.82	42	**591**	696	1.18	−105	165	**120**	0.73	45
*H	**312**	331	1.06	−19	**656**	748	1.14	−92	172	**164**	0.95	8
Mean	538	**348**	0.66	190	1,573	**1,341**	0.96	232	275	**223**	0.83	53

estimating the optimal locations for user-defined key frames because the in-between interpolation strategy can be non-intuitive (even for simple linear interpolation). As a result, users spend considerable mental effort in deciding when to pause the video, as well as effort in correcting errors. These results are summarized in Table 1. In the user-defined key frame experiment for the scripted video, users watched the video multiple times to correct the interpolation between the key frames. Under a fixed rate approach, users only had to watch the video once, increasing efficiency by 33 %. We found similar results for VIRAT: users had difficulty estimating optimal keyframes for cars which can move nonlinearly due to subtle accelerations. On average, users were 17 % faster when labeling with fixed-rate key frames. The basketball clip displayed the least improvement (with fixed rate saving 4 %). Since the motion was fast, confusing, and nonlinear, annotators would annotate frequently, essentially resorting to a fixed-rate keyframe schedule even in the user-defined experiment. For example, Subject B annotated every frame (taking 75 minutes) despite demonstrating an understanding of interpolation on the other videos. He was among the fastest workers on every video except basketball. We still found that workers preferred the fixed rate key frames since the pausing was automatic. Our study demonstrates that despite the flexibility of user defined key frames, a fixed rate schedule is more efficient. The overhead of excess annotations with fixed rate is offset by the cost of repeatedly watching the video with a user defined schedule.

3.3 Multiple Object Annotation

A significant annotation burden is labeling a video with possibly hundreds of objects maneuvering chaotically throughout a scene, such as in Fig. 4. Some objects may be moving independently (such as cars and people), while other objects may be strongly dependent on each other (such as the joints of a person). Since we want a dense labeling, we must determine the most efficient method to label every object under both conditions.

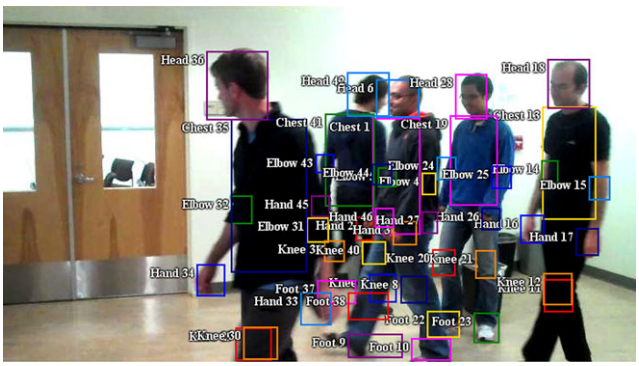


Fig. 4 Our user study demonstrates that our interface can efficiently label the skeleton of a person even in complex videos by labeling one joint at a time

All-objects: A common approach is to annotate every object at once. In this method, the user annotates all of the objects in the first frame, advances to the next key frame, updates all of the annotations, and repeats this process for the entire video. This approach allows the user to watch the video only once since when a frame is labeled, the user never needs to return to it. If the user is able to able to observe the motion of every object in a video as it plays, then labeling every object simultaneously may save time. Indeed, annotators always initially prefer this dense labeling strategy since it seems to save effort.

Table 2 A comparison of annotation time on labeling one object at a time, objects in logical groups, or all objects at once. *Though almost all existing annotation systems are designed for labeling all object of interest simultaneously, we find that it is much more efficient to label objects one at a time*. We asked a small group of our best MTurk workers to label the joints of people in Fig. 2a and the players in Fig. 2b under all annotation modes. The second number in each column is the ratio compared to annotating one object at a time. All times are in seconds. Averages are arithmetic mean for times and geometric mean for ratios. A *dash* means the user declared it to be impossible or gave up; hence many users even refused to label all objects at once

Subject	Human Joints						Basketball						Preferred	Order
	One		Group		All		One		Group		All			
A	**1639**	1.00	1763	1.07	–	–	476	1.00	358	0.75	**253**	0.53	One	OGA
B	**2071**	1.00	2100	1.01	2440	1.17	**1382**	1.00	1618	1.17	2403	1.73	One	AOG
C	**1867**	1.00	2106	1.12	–	–	681	1.00	813	1.19	**599**	0.87	One	GAO
D	**2399**	1.00	–	–	–	–	494	1.00	**448**	0.91	960	1.94	One	AOG
Mean	**1994**	1.00	1989	1.07	2440	1.17	**759**	1.00	809	0.99	1085	1.12		

Single-objects: An alternative method is to require workers to only annotate one object at a time. The user would label a single object for its lifetime, then return to the beginning of the video to annotate the next object. This approach would require the user to watch the video once for each object. Many researchers dismiss single object labeling because it seems wasteful. However, we hypothesize that, for videos with many objects, it may be more efficient since it avoids confounding the user with the simultaneous tracking of multiple objects.

Groups of objects: As a compromise, a hybrid approach is to label objects by groups. Instead of annotating all objects or just one object, users can label all objects in a semantic group. For example, friends walking down the street often are in close proximity and maintain similar velocities. In this method, the user would annotate each person in one group first, rewind the video, and annotate people in the next group. Another example of this scenario is the labeling of human body parts. Each body part is treated as an object, and parts belonging to the same person form natural groups.

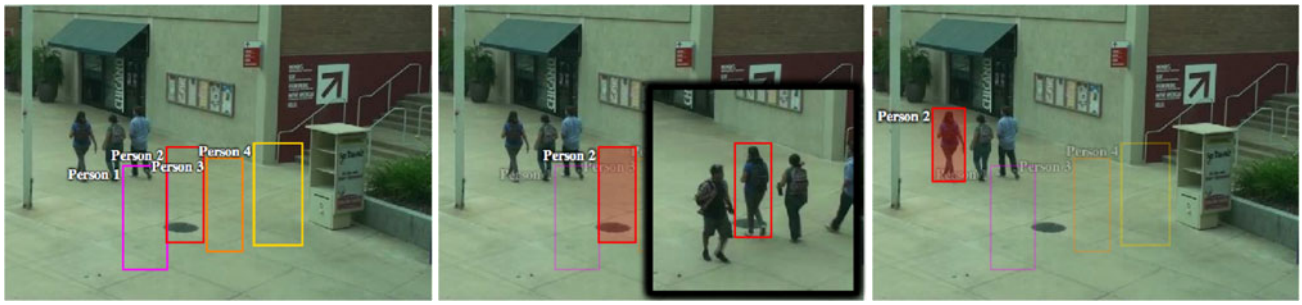
User-study results: Our user study reveals that annotating one object at a time is not only superior, but also strongly preferred by MTurk workers. Our results are summarized in Table 2. We asked workers to annotate both the players in the basketball game in Fig. 2b, as well as the joints (e.g., hands, feet, elbows, etc.) of the people in the scripted video clip from Fig. 2a. Although person joints may appear to be a prime candidate for grouping since they are physically dependent on each other (when the hand moves, so must the elbow), *our results strongly suggest the opposite: annotating one object/body-part at a time is not only more efficient, but it also preferred by the users*. Some users even refused to complete the study if they were required to annotate in groups.

Impact of results: When conducting our user studies in a controlled environment, we discovered new users always initially incorrectly annotated in a sub-optimal grouping strategy by attempting to label every object at once. Some users would eventually realize that this strategy was poor, but the thought of making multiple passes through the video did not occur to most. For those that did learn, they resorted to annotating in groups. Only rarely and after frequent use of our system did users eventually converge to labeling one object at a time. This reveals that users would benefit from explicit instructions guiding them to a single-object strategy immediately, or moreover, a restricted interface that forces them to label single objects at a time.

3.4 Maintaining Track Identity Across Frames

A common mistake among annotators is to confuse the identity of an object between frames. This failure results in large errors because two tracks will swap. For example, the basketball players all look similar; they are only distinguished by the number on their uniform, but the low resolution nature of the video makes the number unreadable. Consequently, users often switch the identity of a basketball player when the motion becomes confusing. We wish to design our interface to minimize this risk.

Video Playback: We found human annotators heavily rely on the motion of objects in order to correctly decode the scene. In early versions of our tool, workers were only shown key frames and could not play the video. While this approach would be fast since the worker only needs to annotate static images, maintaining track identity was impossible for the basketball game and barely feasible for simpler videos. The local appearance of an object is necessary but not sufficient to maintain object identity across frames. The user must *watch* the video play in order to correctly track. We believe this is the case because a large portion of the human ability to track objects is by following the changes between frames. Upon this discovery, we made the ability to play the video an integral component of our user interface.



(a) Which person is the red box tracking? (b) A tooltip reminds the worker. (c) The worker can update the box.

Fig. 5 Maintaining track identity between frames can be difficult. By providing a drop down tooltip for each object that summarizes its identity, workers can reliably keep the box on the same object. The tooltip plays a slideshow of the key frames already annotated for that object, allowing the user to "see into time" while keeping the playhead on the frame of interest

Spacetime Tooltip: Despite the playback ability, users would still frequently become confused about the identity of an object from the previous key frame. In our user studies, we observed that users often forget which object they were originally tracking when the motion is complex or unpredictable. Users would then have to watch the video multiple times as they attempted to decode the motion. To prevent this frequent and unnecessary overhead, we now display a small tooltip, shown in Fig. 5, next to each object that reminds the user of the tracked object's appearance and motion. Upon clicking on a bounding box, a movie summarizing the object and its previous annotations will play. This tooltip allows the user to quickly play through the video without moving the playhead. Since the tooltip movie skips most frames and plays at an accelerated frame rate, the user is able to effortlessly visualize the trajectory and appearance of the object, thereby giving the user access to history of the object without modifying the state of the current interface. We found users were enthusiastic with the addition of this feature and adopted its use upon discovery.

3.5 Attributes and Visibility

In addition to labeling the trajectory of objects, we also wish to annotate binary temporal attributes about the object. For example, at any given frame, we may wish to know whether a person is walking or running, their activity, whether they are occluded, or if they are visible in the frame. In order to obtain these labels, we display check boxes next to the object that the user can mark at key frames. Similar to spatial annotation, frame-by-frame labeling is inefficient, so we must develop an appropriate interpolation scheme for binary attributes.

For both spatial and attribute annotations, we must distinguish between three types of frames: positive annotations (the user explicitly indicates that this object is in state X), negative annotations (the user indicates that it is not in state X), and lack of annotations (the state is interpolated/extrapolated). If the user does not explicitly label a frame, he may agree with the interpolation or he may be waiting for a more opportune moment to change its label. Our user interface must be able to distinguish between these inherently ambiguous cases.

Timeline Independence: Attribute key frames must be independent of spatial annotations. If the user adjusts the bounding box, the user does not necessarily agree with the attribute labeling. From the feedback during our user studies, we found that users preferred to annotate objects first spatially and attributes second. The cognitive effort required to simultaneously annotate both a bounding box and its attributes was too taxing. Consequently, each attribute should have its own timeline, allowing the worker to make one pass for each attribute and one pass for bounding boxes.

Interpolation: When the user seeks to a frame, we predict a binary label for each attribute. We found a simple yet effective approach is to assume the attribute label has the same label as the immediate keyframe previous in time. This strategy works well for attributes that describe the objects appearance as well as whether the object is inside the view frame.

3.6 Constrained Interface

We believe our interface is successful because we limit the number of available choices and constrain the worker to a closed world. Work in psychology and human-computer interaction reveals that minimizing interruptions and choices can significantly reduce user anxiety and increase efficiency (Schwartz 2005; Bailey and Konstan 2006; Mark et al. 2005). In our interface, we follow similar principles. For example, users must annotate on fixed key frames, the types of objects are predefined, and we only support rectangular bounding boxes. We have observed that annotators are tempted to try to do too much at once. Users instinctively feel that annotation is more efficient if they do everything

simultaneously, yet our user studies demonstrate that this cognitively overloads the user and, contrary to our instinct, slows us down. Instead, a restricted interface allows users to efficiently annotate: by only making one decision at a time, we are able to more efficiently annotate video.

Precision-Cost Trade-off: A more flexible interface, such as LabelMe video (Yuen et al. 2009), might allow more powerful annotations, but at the expense of increased annotation effort. In contrast to our framework, LabelMe video supports user defined key frames, free text entry for the type of objects, arbitrary polygons, and the ability to link objects together by semantic events. These features all allow more rich annotations. If we desire a rich and detailed annotation corpus, then the flexibility is necessary. But, if we can accept lower precision annotations, then restricting the interface will result in a significant savings of time and effort, as our user studies have demonstrated. As we wish to deploy our system on Amazon's Mechanical Turk with massive data sets, saving the annotators time is highly desirable since it reduces our costs.

4 Mechanical Turk

We ran our annotation system within Amazon Mechanical Turk (MTurk), an online labor marketplace that allows employers to easily hire workers to complete tasks. MTurk is ideal for jobs that are difficult for computers but trivial for humans. As employers, we create *Human Intelligence Tasks* (HITs) and set prices for each task before posting to the MTurk servers. Workers around the world, as in Fig. 6, then browse offered jobs and accept those that interest them. Task completion is not guaranteed and is the result of typical market dynamics (Ross et al. 2010). Upon validation of completed work, Amazon releases escrowed payments to the workers.

In the remaining sections, we describe aspects of our system that relate to crowdsourcing on MTurk. There are two main challenges: a) How do we split up a video annotation job into small, distributed tasks for individual workers? and b) How do we ensure high-quality results? We begin with the former, but spend most of our efforts addressing the crucial latter question.

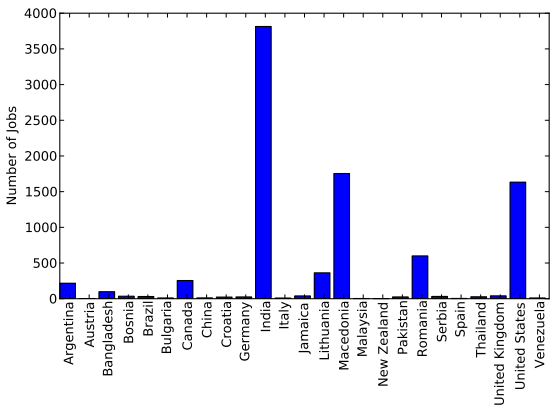


Fig. 6 Where are the workers? Most jobs for VIRAT were completed from workers in India, Macedonia and the United States

4.1 Shot-Based Annotation

We first break every video up into many small, overlapping segments, typically of about ten seconds each. If the video consists of multiple shots, we use a standard scene recognition algorithm (Oliva and Torralba 2001) to create segments along scene boundaries. We then publish each segment onto MTurk and pay a worker to annotate every object in a segment using our interface previously described.

After all segments have been labeled, we must stitch the annotations between segments to create continuous paths that span the original video. Since each segment overlaps another (typically by a second), we can use these redundant annotations to correspond tracks with each other. We establish correspondence between the set of tracks from segment S and the adjacent segment T. We map every path $i \in S$ to another path $j \in T$ using the mapping function f. By searching for the f that minimizes the assignment costs $C(i, j)$, we can compute an optimal assignment between segments:

$$\min_f \sum_{i \in S} C\big(i, f(i)\big) \quad \text{where } f : S \to T \tag{1}$$

Equation (1) is equivalent to a minimum-weight bipartite matching problem, which we solve using the Hungarian algorithm (Munkres 1957). Without loss of generality, let the overlap region between paths to be on the interval $0 \le t \le T$. Let i_t and j_t be the bounding box at time t for each path respectively. We define the cost of assigning path i to j to be low when they sufficiently overlap for a majority of frames:

$$C(i, j) = \sum_{t=0}^{T} \begin{cases} 0 & \text{if both are visible and overlap} \\ 0 & \text{if both are not visible} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

Leftover tracks are matched to dummy nodes with a fixed cost. We then use the pointers f to link tracks between shots. This approach works well, although it may require some manual cleanup when objects are small or there are mistakes in the annotations.

4.2 Macro vs Micro Tasks

A common principle in crowdsourcing is to design tasks that are *micro*, i.e. the worker can solve them quickly, easily, and effortlessly. This conventional wisdom stems from observations that workers often solve HITs when they are bored, do not want to exert significant cognitive effort, and lack

time for involved tasks. In early versions of our tool, we followed this principle and divided work in order to minimize the effort required by each individual worker. In these experiments, we instructed a worker to annotate only one object for an entire segment. After completing the annotation, a different worker would annotate another object. To prevent duplicate annotations on the same object, workers saw previous annotations from other users. While this protocol encourages rapid participation from many workers, we discovered that micro-tasks are *not* suitable for high quality video annotation.

The problem with micro tasks: The crucial flaw with the given approach is that workers need to rely on accurate and unambiguous annotations by previous workers. We found that small errors propagated and compounded themselves. For example, it may be unclear when a particular track begins or ends due to partial occlusion. A new worker may begin annotating an unannotated object, only to discover in a few frames that a previous worker annotated the same object with a slightly late starting frame. At this point, they *should* restart according to our directions, but most workers feel compelled to continue the annotation to minimize wasted effort. Because workers often look to previous annotations as a guide, such mistakes are mimicked and compounded across subsequent work.

Macro tasks: Rather, video annotation requires *macro* tasks in order to obtain high quality labels. *Instead of annotating one object per task, the worker should take "ownership" of a video segment and annotate every object*. The cognitive effort expended into visually decoding a scene is wasted if each object is annotated by a different worker. Although our user studies demonstrate that workers should annotate one object a time, we found that the same worker must annotate every object in the video. By granting ownership of an entire segment to only one worker, we are able to distribute small errors independently across multiple segments in a video, rather than compound small errors within a segment, resulting in higher quality annotations. Figure 7 demonstrates that there are workers who are willing to work on tasks that require hours of attention.

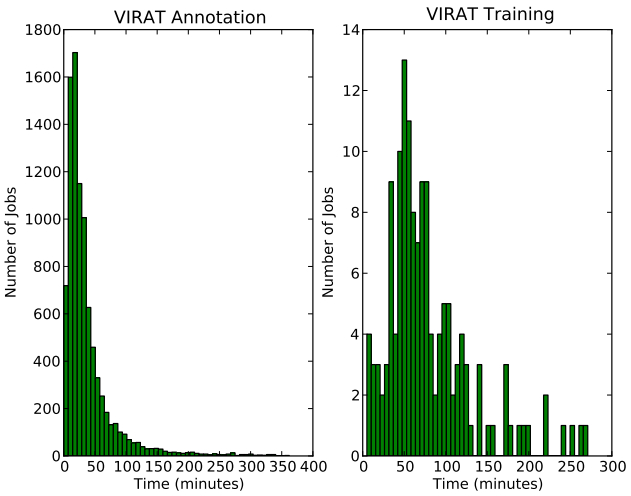


Fig. 7 How long did each worker spend annotating? Both plots are for the VIRAT video data set. The *left* shows labeling times for an actual annotation session after workers had passed our quality control measure. The *right* shows worker times when they were learning how to annotate video and attempting to pass our quality control measures. For a ten second clip, most workers took up to an hour to annotate every object in VIRAT. The total annotation time for VIRAT was 8 man months for 27 hours of video

4.3 Discovering Good Workers

Video annotation is hard: In this section, we outline our protocol for generating high-quality video annotations. Compared to image annotation, video annotation is a surprisingly difficult task for a variety of reasons. Firstly, object tracks can be ambiguous in both the spatial and temporal domains. For example, it is hard to resolve starting and ending frames for objects undergoing partial occlusions. Secondly, annotation can be tedious. When annotating a crowd of people walking in low resolution, it can be difficult to avoid swapping track identities. And finally, some workers have difficulty understanding the concept of object continuity with respect to an annotation interface. Users sometimes instantiate a new track when an object reappears from an occlusion, or in the extreme case, some users marked a new track for every single frame. Because video annotation is hard, we found that most workers, despite accepting the task, do not have the necessary patience or skill to be accurate annotators.

The good, the bad, and the ugly: The typical approach for quality control in crowdsourced scenarios is to self-validate by asking for multiple annotations for the same data point. The underlying assumption is that most workers are honest, and that a small number of dishonest workers that can be identified out through validation. As we argued above, many if not most honest workers will also be poor annotators, mandating the need for a refined strategy for quality control. To that end, we classify workers into three types, and describe methods for identifying each: *good* workers who are honest and skilled; *bad* workers who are honest but unskilled; and *ugly* workers who are dishonest and cheaters. We find that it is crucial to eliminate both bad and ugly workers for high-quality crowdsourced video annotation.

Eliminating the bad: When a new worker visits our task, we silently redirect them to a "gold standard" annotation challenge shown in Fig. 9. Since we secretly know the ground truth annotation, we can automatically evaluate their performance to check if they are skilled annotators. We select a different gold standard video for each data set that is representative of the type of problems the human annotator must overcome (e.g., frequent occlusions, camera motion, or fast motion). Given a set of ground truth tracks G and candi-

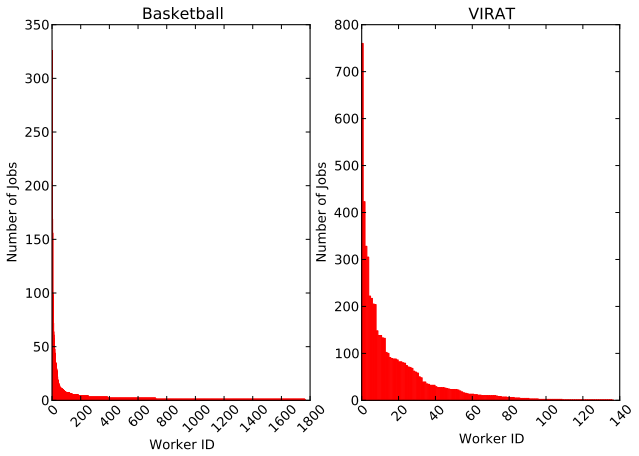


Fig. 8 How many jobs did each worker complete? On the *left*, we show how many jobs each worker completed on the 200,000 frame basketball game without any quality control. On the *right*, we show the number of jobs completed per worker across the 3,000,000 frame VIRAT data set with after a quality control evaluation test. Notice that the basketball video is an order of magnitude shorter in length, yet its protocol required an order of magnitude more workers. Our quality assurance protocol eliminates the long tail of workers only attempting a few jobs. By identifying the workers that are "good" at video annotation, we are able to obtain very high quality labels

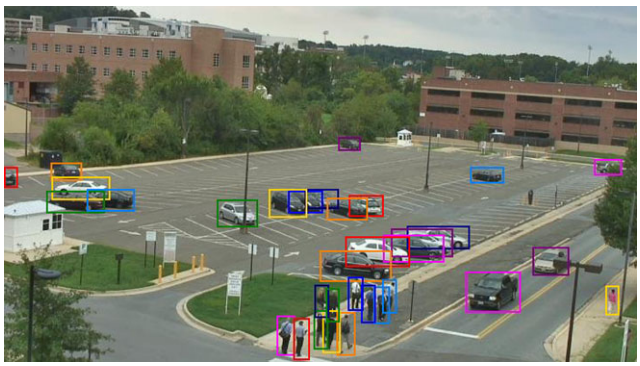


Fig. 9 The gold standard video a worker must label to demonstrate their annotation ability before they are allowed to submit work for VIRAT. The video is slightly more challenging than the majority of VIRAT in order to select the best MTurk workers

date paths S, we establish correspondence using the assignment problem in Eq. (1). If G and S do not sufficiently agree under the best matching, then the worker fails the challenge, but is allowed to try again an unlimited number of times. Because interested but misguided "bad" workers may put forth considerable effort and still fail, we prevent them from submitting work, which never sets them up for a rejection. Once the worker passes the gold standard challenge, they are given access to the true, unlabeled data set. Crucially, workers are never informed that they have been evaluated by a gold standard. Figure 8 shows that most of the long-tailed distribution of workers falls under the "bad" category, and are eliminated.

Eliminating the ugly: The above quality control protocol will successfully separate the good from the bad, but it does not ward off the "ugly" worker. An adversarial worker might provide a perfect gold standard annotation then revert to low quality, automated submissions completed by a robot. While contemporary crowdsourcing employs thousands of workers, our gold standard challenge only accepts hundreds of workers. This means that rather than verifying work, we can verify the worker. We manually verify each worker by randomly sampling and viewing a couple of jobs, a process that took less than half an hour in total. In practice, we discovered one "ugly" worker who consistently submitted gamed annotations. We blocked him from working on our task and recreated his jobs. In essence, our validation protocol allows us to use Turk not directly as a crowdsourced platform, but as a marketplace to identify reliable and skilled annotators, which we found crucial to providing high-quality annotations.

Possible limitations: One limitation of eliminating workers is that throughput may decrease. Most of the time, a few workers perform a majority of the tasks, and many workers perform a few or a single task. Because "bad" workers tend to lie on the tail of the distribution in Fig. 8, we are still able to take advantage of a core group of validated workers who performed hundreds or thousands of jobs. We experimentally have noticed that our jobs have a long warm up time as our system identifies good workers, but once workers have been matched to our jobs, completion is rapid. An unexpected benefit was consumer loyalty; validated workers were eager for any additional work, and also strongly supported us on bulletin boards.

4.4 Worker Compensation

Payout: In a macro-task protocol, videos containing more objects are more difficult to label. To compensate workers proportionally with video difficulty, we paid workers a fixed rate per object. For example, on videos of intermediate difficulty (such as VIRAT), we paid 5 cents per object in a 10 second shot, shown in Fig. 10. For more difficult videos like the basketball game, we paid 15 cents per object, while for extremely difficult videos such as the basketball, we paid 50 cents. We also paid the worker a small completion bonus, typically a few cents, if they successfully annotated every object. Since we do not know the true number of objects in each video, we always paid the completion bonus regardless of the number of objects annotated. This payout scheme assumes that we trust the worker to not overlabel hallucinated objects; we found that was reasonable for "good" workers that passed our gold-standard validation. In general, we found this payment schedule to produce high quality, dense annotations. If we paid independently of the number of objects, workers will only annotate a few objects

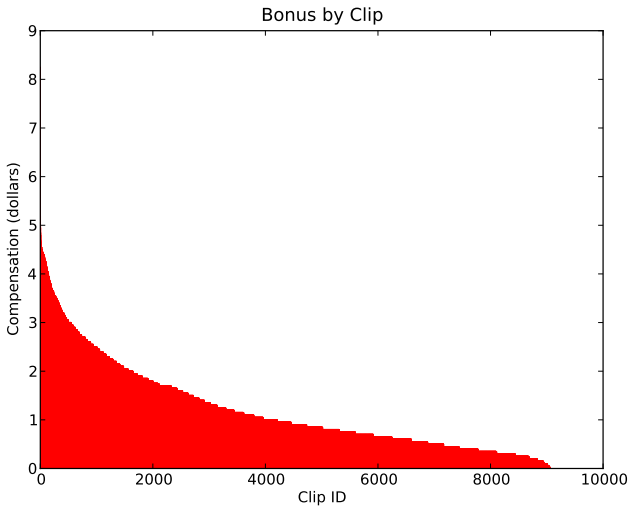


Fig. 10 How much do we spend on each clip? At $0.05 per object, most VIRAT video clips cost between $1 and $2, resulting in a cumulative cost of tens of thousands of dollars. Since video annotation is a skilled task, we must hire expert workers

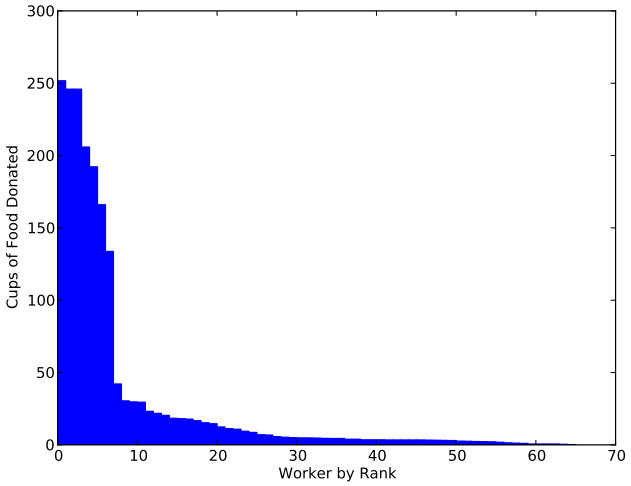


Fig. 11 For a subset of jobs, we allowed workers to donate their pay to the Untied Nation's World Food Programme. Workers could choose to donate none, half, or all of their pay. A total of 65 workers donated 2,000 cups of food (worth $500), demonstrating its viability

since there is no incentive to provide a dense labeling. Similarly, if we do not include the completion bonus, workers will only annotate the easy objects. By including both the completion bonus and the per object pay, workers will maximize their pay and provide dense annotations.

Rejection: As video annotation requires specialized skill, we must attract the best workers. With the rise of online communities that advocate for the workers such as TurkOpticon.org and TurkerNation.com, maintaining a strong reputation amongst the workers is crucial. When a worker completes a task for a requester, they can report their experiences to these communities in an effort to warn others against unfair practices. In early versions of our tool, we automatically rejected unsatisfactory work. Since most workers cannot annotate video sufficiently, most work was rejected, ultimately resulting in negative ratings that tarnished our reputation. Once we received enough negative press, workers were so scared of a rejection that we inadvertently turned away the best workers. Indeed, MTurk workers take a single rejection harshly. One worker even threatened to involve the IRB over his rejection:

> "I feel strongly about my 20 cents…I expect to *[sic]* paid in the next 24 hours or I WILL let the IRB know ASAP."—an Ivy league student

While this worker was very upset about twenty cents, some workers reported that they care more about their personal statistics rather than the pay. MTurk records how many jobs each worker has completed and their acceptance ratio. These statistics are paramount:

> "I am fine if you do not want to pay me for these, but rejecting spoils my qualifications."—MTurk worker

We now accept every task regardless of its quality. A strong reputation among workers is worth the overhead cost. Notice that our gold standard challenge prevents bad workers from entering our system and setting themselves up for rejection; since we do not allow poor quality workers to even enter our system, we cannot hurt their statistics. Moreover, we use this paranoia to our advantage by threatening to reject assignments. We inform workers that we reserve the right to reject poor work and this is effective at motivating workers to produce high quality labels.

Motivational Feedback: We found giving our workers automatically generated motivational feedback was effective at encouraging workers to continue annotating. As we argued above, workers are cautious to work on new jobs due to fear of rejection. When we accept an assignment, we also generate a small string (e.g., "Keep up the fantastic work!") to provide feedback. Workers reported that they were thankful for this positive confirmation:

> "I really appreciate the positive Feedback… its nice to know I'm doing what you need."—MTurk worker

This feedback encouraged workers to continue annotating for us because they felt confident that their future work would be accepted.

Charity Incentive: As a further experiment analyzing the motivation of a worker, we allowed workers to forgo compensation and instead donate their pay towards charity. Upon completing a task, workers were given an option to either donate none, half, or all of their pay to the United Nation's World Food Programme. These results are summarized in Fig. 11. Intriguingly, our results suggest that workers are willing to not receive a direct payment for their services, so long as they see their time is valued as evidenced by the do-

nation amount. We hypothesized that we may also observe a higher quality output for work with donated funds, as this seemingly would disincentivize an "ugly" worker looking to game the system. However, we did not see an increase in accuracy, probably due to the fact that our aggressive validation protocol already eliminated the "ugly" workers. We believe that further exploration of charity or other economic incentives may produce higher quality work.

5 Interpolation

Vital to our analysis is the ability to properly interpolate between a sparse set of annotations. Our labeling tool requests that a worker labels the enclosing bounding box of an entity every T frames. Offline, we interpolate the object path between these key frames using a variety of algorithms. Because this is done offline, we can afford to use computationally expensive schemes. We define b to be the coordinates of a bounding box:

$$b = \begin{bmatrix} x_1 & x_2 & y_1 & y_2 \end{bmatrix}^T \tag{3}$$

We write b_t for the bounding box coordinate of an entity at time t. Without loss of generality, let us define the keyframe times to be time 0 and time T. We define the interpolation problem to be: Given b_0 and b_T, estimate b_t for $0 < t < T$.

5.1 Linear Interpolation

The simplest approach is linear interpolation:

$$b_t^{lin} = \left(\frac{t}{T}\right) b_0 + \left(\frac{T-t}{T}\right) b_T \quad \text{for } 0 \le t \le T \tag{4}$$

Yuen et al. (2009) makes the astute point that constant velocity in 3D does not project to constant velocity in 2D due to perspective effects. Assuming the tracked entity is planar, they describe a homography-preserving shape interpolation scheme that correctly models perspective projection. However, we found simpler linear interpolation to work well for many common scenes where there does not exhibit much depth variation.

Both of these interpolation algorithms are very efficient since they avoid the need to process any pixel data. However, both assume constant velocity which clearly does not hold for many entities (Fig. 12). In the remainder of this section, we describe a dynamic programming based interpolation algorithm that attempts to *track* the location of the entity given the constraints that the track must start at b_0 and end at b_T.

5.2 Discriminative Object Templates

To score a putative interpolation path, we need a visual model of the tracked object. We use all the annotated keyframes within a single video shot to build such a model. Assume N such keyframes exist, which in turn yield N bounding boxes which contain the entity of interest. Our first approach was to construct an average pixel-based template, and score it with sum-of-squared differences (SSD) or normalized correlation. We also experimented with more sophisticated templates built on histogram of oriented (HOG) (Dalal and Triggs 2005) features. We found poor results with both.

(a) Basketball (b) VIRAT

Fig. 12 Nonlinear motion requires more sophisticated interpolation strategies for estimating entity position given the end locations. We employ a tracker in order to find the actual path through visual analysis

We believe the suboptimal results arose from the fact that such templates are not designed to find objects in the cluttered backgrounds that we encountered. To compensate for this fact, we extract an *extremely* large set of "negative" bounding boxes collected from the N keyframes, making sure that they do not overlap the entity of interest in those frames. We then attempted to score a putative bounding box by competing an object template with an average background template. We again found poor results, this time due to the fact that our video backgrounds are complex and are poorly modeled with an average template.

Finally, we converged on the approach of learning a discriminative classifier trained to produce high scores on positive bounding boxes and low scores on the negatives. For each bounding box b_n we compute a feature descriptor composed of HOG and color features:

$$\phi_n(b_n) = \begin{bmatrix} HOG \\ RGB \end{bmatrix} \tag{5}$$

where $RGB \in \mathbb{R}^{3+6}$ consists of the 3 means and 6 covariances of the three color channels computed from all pixels in window b_n. When trained with a linear discriminative classifier, these color features are able to learn a quadratic decision boundary in RGB-space. Before extracting features, we resize the image patch at b_n to the "canonical" object size estimated from the average of the N labeled bounding boxes. Given a collection of features along with labels $y_n \in \{-1, 1\}$ identifying them as positives or negatives, we learn a linear SVM weight vector w that minimizes the following loss

function:

$$w^* = \underset{w}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \max\bigl(0, 1 - y_n w \cdot \phi_n(b_n)\bigr) \quad (6)$$

We use liblinear (Fan et al. 2008), which appears to be fastest linear SVM solver available. For typical size problems, training took a few seconds.

5.3 Constrained Tracking

We write the constrained endpoints given by the key frames as b_0^* and b_T^*. We wish to use the template w to construct a low-cost path $b_{0:T} = \{b_0 \cdots b_T\}$ subject to the constraints that $b_0 = b_0^*$ and $b_T = b_T^*$. We score a path by its smoothness and the local classifier scores:

$$\underset{b_{1:T}}{\operatorname{argmin}} \sum_{t=1}^{T} U_t(b_t) + P(b_t, b_{t-1}) \quad (7)$$

$$\text{s.t.} \quad b_0 = b_0^* \quad \text{and} \quad b_T = b_T^* \quad (8)$$

We define the unary cost U_t to the SVM score truncated by α_1 so as to reduce the penalty during an occlusion:

$$U_t(b_t) = \min\bigl(-w \cdot \phi_t(b_t), \alpha_1\bigr) + \alpha_2 \bigl\|b_t - b_t^{lin}\bigr\|^2 \quad (9)$$

We introduce a prior on linear interpolation to allow our tracker to gracefully degrade to linear interpolation when the path becomes extremely difficult. In practice, α_2 is very small and often zero. We note that we are able to efficiently compute the dot product $w \cdot \phi_t(b_t)$ using convolution kernels capturing both the HOG template and the weights for the color features.

We then define the pairwise cost to be proportional to the change in position:

$$P(b_t, b_{t-1}) = \alpha_3 \|b_t - b_{t-1}\|^2 \quad (10)$$

Note that the constraints in (8) can be removed simply re-defining the local costs to be:

$$U_0(b_0) = \inf \quad \text{for } b_0 \neq b_0^* \quad (11)$$

$$U_T(b_T) = \inf \quad \text{for } b_T \neq b_T^* \quad (12)$$

5.4 Efficient Optimization

Given K candidate bounding boxes in a frame, a naive approach for computing the minimum cost path would take time $O(K^T)$. It is well known that one can use dynamic programming to solve the above problem in $O(TK^2)$ by the following recursion (Bellman 1956):

$$cost_t(b_t) = U_t(b_t) + \min_{b_{t-1}}\bigl[cost_{t-1}(b_{t-1}) + P(b_t, b_{t-1})\bigr] \quad (13)$$

where $cost_t(b_t)$ represents the cost of the best path from $t = 0$ to b_t. We initialize $cost_0(b_0) = U_0(b_0)$. By keeping track of the argmin, one can reconstruct the minimum cost path ending at any node.

Note that the above recursion can be written as a min-convolution (Felzenszwalb and Huttenlocher 2004), allowing us to compute the optimum in $O(TK)$ using distance-transform speed-ups:

$$cost_t(b_t) = U(b_t) + \min_{b_{t-1}}\bigl[cost_{t-1}(b_{t-1}) + \alpha_2 \|b_t - b_{t-1}\|^2\bigr] \quad (14)$$

6 Results

The computer vision community has validated the power of our public online labeling framework by collectively placing thousands of videos on MTurk. Figure 14 shows a small sample of the types of videos that our system has annotated. Our system is robust to many annotation problems—such as frequent occlusions, motion blur, drastic camera motion, variations in pose, and cluttered backgrounds—and can annotate videos of any difficulty. These experiments demonstrate that our framework will successfully scale to building massive video data sets.

In order to conduct our study on the economics of video annotation, we selected four different data sets of varying difficulty shown in Fig. 13. First, we examine "easy" videos of people performing athletic drills where they are easily distinguished from the background. Second, we look at "medium" videos from the VIRAT challenge video surveillance data set. VIRAT is unique for its enormous size of over three million frames and up to hundreds of annotated objects in each frame. Third, we look at a "difficult" task of annotating basketball players who tend to undergo a fair number of occlusions in cluttered backgrounds (as in Fig. 1 and Fig. 13d) throughout a two hundred thousand frame video. Finally, we consider the task of annotating "very difficult" entities such as a basketball (as in Fig. 12), which is hard to track due to frequent occlusions by players and the existence of large amounts of motion blur relative to its small image size. We use these data sets to examine cost trade-offs between automation and manual labeling as a function of the difficulty of the data. Our labeled basketball video data is unique for its size and complexity, and we have made it available to the community for further research on activity analysis.

We deploy our previously described user interface to have workers annotate our video data sets on MTurk. We then use these dense sets of annotations as ground truth. We hold out different intervals of ground truth in order to determine how well the tracking and interpolation methods predict the missing annotations. We score our predictions with the same criteria as the PASCAL challenge:

(a) Athletic Drills: Trivial due to easily distinguished foreground.

(b) VIRAT Cars: Intermediate due to stationary cameras and little motion

(c) VIRAT People: Difficult due to small size and frequent motion

(d) Basketball Players: Difficult due to frequent occlusions and similar looking objects

(e) Basketball Ball: Extremely difficult due to cluttered backgrounds

Fig. 13 Some of the data sets that we have annotated using VATIC. Despite the varying difficulty in videos, annotation quality is very good. All annotations shown are from MTurk workers

a prediction must overlap with the actual annotation by at least 50 % to be considered a detection (Everingham et al. 2010).

6.1 Diminishing Returns

We first confirm our hypothesis that the "Turk philosophy" (human computation is cheap and subsumes automated methods) does not hold for video because it is wasteful for users to annotate every frame. Figure 15 shows a diminishing returns property in which increased human labeling (x-axis) results in smaller and smaller reductions in error rates (y-axis). Moreover, the rate of the diminishing is affected both by the difficulty of the video (easy, medium and difficult) and the choice of interpolation algorithm. For medium videos, we can achieve 10 % error with a user annotation rate of 0.001 clicks per frame or 1,000 frames between clicks. For medium-difficultly videos, we require at least 0.05 clicks per frame regardless of the mode of interpolation. Finally, for difficult videos, we need 0.2 clicks per frame for the best accuracy. Our results suggest that interpolation of any kind can exploit the redundancy in video to reduce annotation effort by an order of magnitude compared to a naive, "brute-force" MTurk approach.

6.2 CPU vs Human Cost

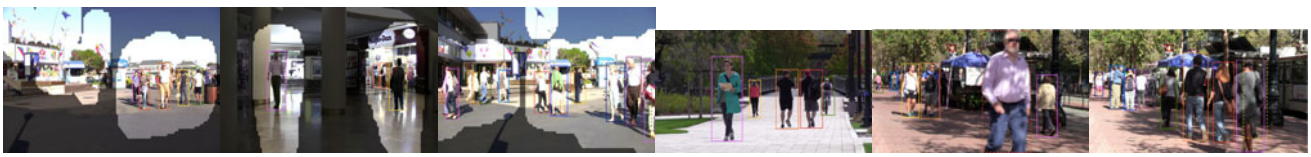
As part of our best-practice analysis, we would like to answer the question: Given $X, how should one divide/manage human effort versus CPU effort (required to interpolate annotations) so as to maximize track accuracy? To do so, we make three simplifying assumptions. First, we assume that linear interpolation is virtually free in terms of CPU usage since it requires a minimal amount of floating point operations compared to our tracking-based interpolation. We next assume that tracking-based interpolation will require a fixed amount computation regardless of the annotation frequency. This is reasonable because the length of an interpolated interval and the number of intervals to interpolate

(a) MINDS Eye: Easy due to scripted actions and deliberate motions (Anonymous, 2012)

(b) People Exercies: Intermediate due to stationary cameras and scripted actions (Demiröz et al, 2012)

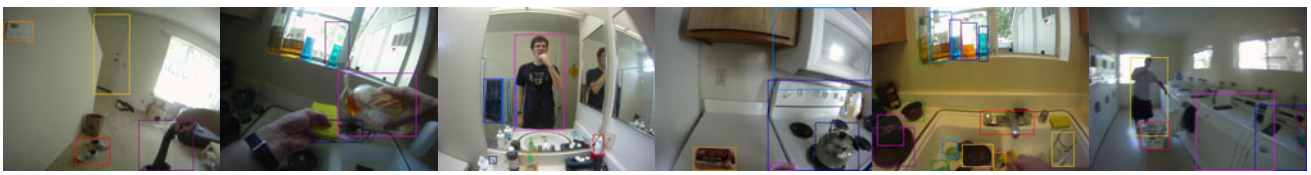
(c) Animals: Intermediate due to frequent occlusions and cluttered foreground (Liu and Lazebnik, 2011)

(d) Kinect Camera: Intermediate due to camera motion (Aydemir et al, 2012)



(e) Pedestrians: Intermediate due to frequent occlusions (Chen et al, 2011)

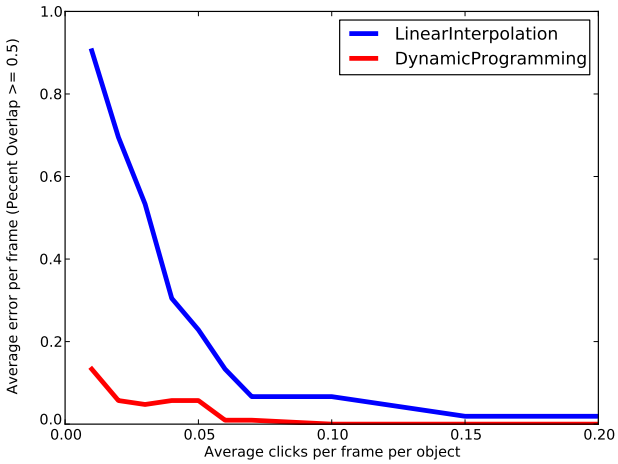
(f) Aerial: Difficult due to extreme camera motion and video artifacts (Oh, 2011)



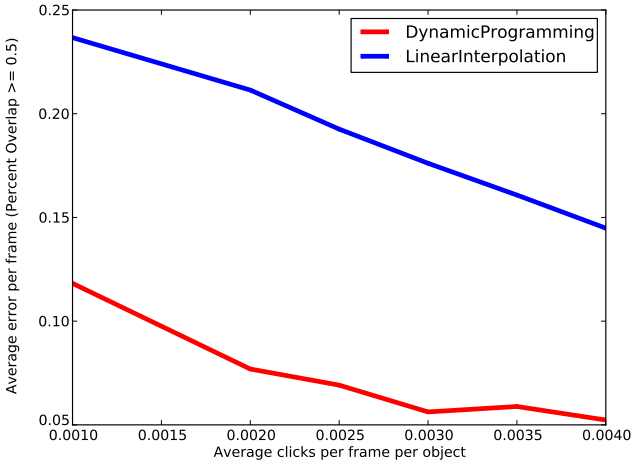
(g) First Person Camera: Difficult due to extreme camera motion and many small objects (Pirsiavash and Ramanan, 2012)

Fig. 14 Examples of the videos that the community has annotated using VATIC. Our system is robust to common annotation problems and can scale to most types of video (Anonymous 2012; Demiröz et al. 2012; Liu and Lazebnik 2011; Aydemir et al. 2012; Chen et al. 2011; Oh 2011; Pirsiavash and Ramanan 2012)

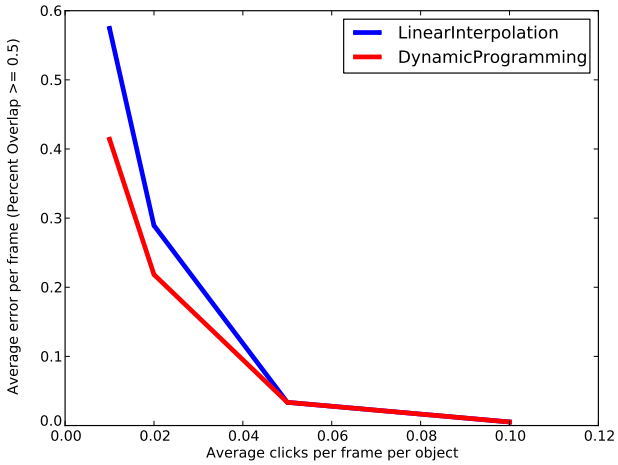
scale inversely with each other. This means that for a fixed length-video, tracking-based interpolation will cost some A amount of dollars regardless of the annotation frequency. Finally, we assume that we can generate interpolation algorithms that incur $\alpha \cdot A$ dollars for CPU usage by randomly flipping a α-biased coin to determine if a particular interval should be interpolated linearly or with a tracking algorithm.



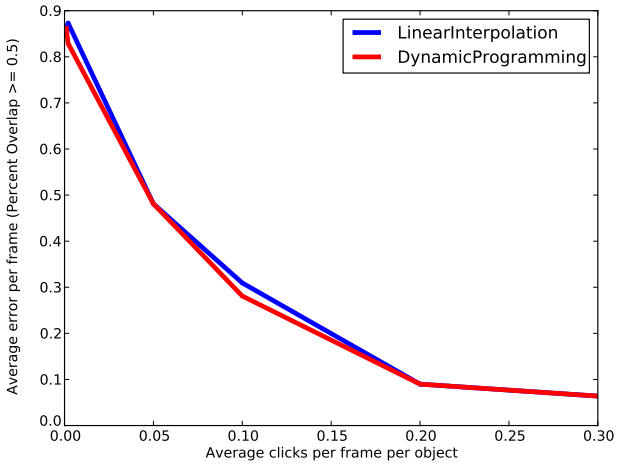
(a) Athletic Drills (easy)



(b) VIRAT Cars (medium)



(c) Basketball Players (difficult)



(d) Ball (very difficult)

Fig. 15 Performance of dynamic programming and 2D linear interpolation on easy, medium, difficult, and very-difficult data sets. Dynamic programming excels when features are easily extracted, such as in athletic drills or VIRAT. But, dynamic programming performs equally well as linear interpolation when the object is highly occluded (such as a basketball)

We use rates from Amazon's Elastic Compute Cloud (EC2) platform to compute a monetized CPU cost. On average, our tracking algorithm takes 0.12 seconds per frame per object on a single core Intel Xeon 2.67 GHz processor. Amazon charges US$0.085 per hour (effective) for a comparable computer. The two hour basketball game will take 40 hours to process all the players and referees with total cost of US$3.40. We then use our worker compensation rates from MTurk to compute the cost of paying a human to label every frame. We paid workers US$0.15 per basketball player to annotate 300 frames. On average, a worker annotated every 12 frames. A frame-by-frame labeling would cost us US$5,947. We do a similar analysis for the athletic drills, VIRAT, and the basketball. We finally plot the error-by-cost graphs in Fig. 16.

6.3 Performance Cost Trade-off

We now consider our motivating question: how should one divide human effort versus CPU effort so as to maximize track accuracy given a X$? A fixed dollar amount can be spent only on human annotations, purely on CPU, or some combination. We express this combination as a diagonal line in the ground plane of the 3D plot in Fig. 16. We plot the tracking accuracy as a function of this combination for different X$ amounts. We describe the trade-off further in Fig. 17.

We note that researchers often desire zero error in their dataset: what is the cost for a perfect annotation? Figure 17 reveals that we can obtain nearly zero error on athletic drills for $50 and on VIRAT for $10,000 by running a tracking algorithm. However, for difficult videos, the tracking algorithm struggles and linear interpolation is sufficient for a

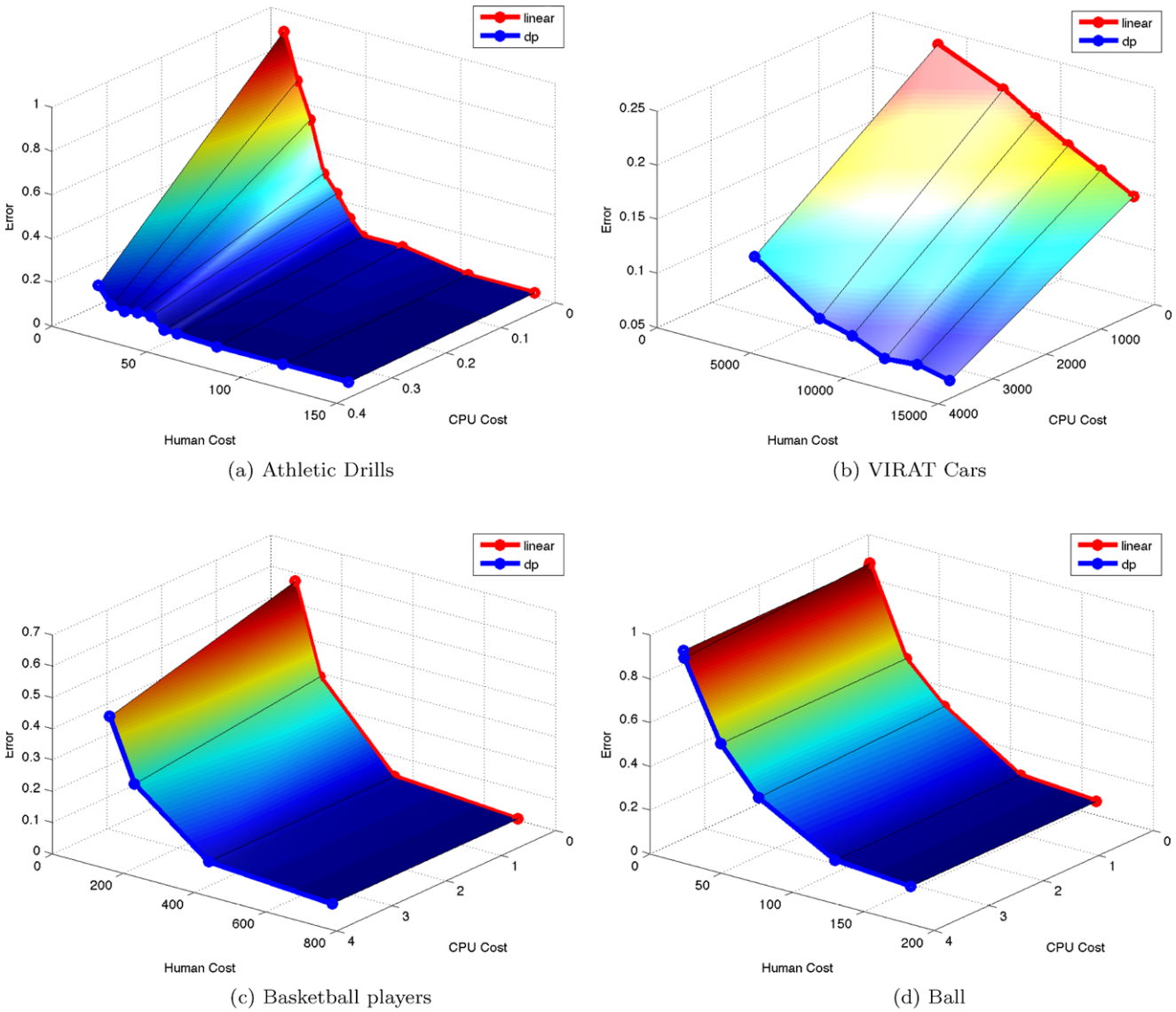
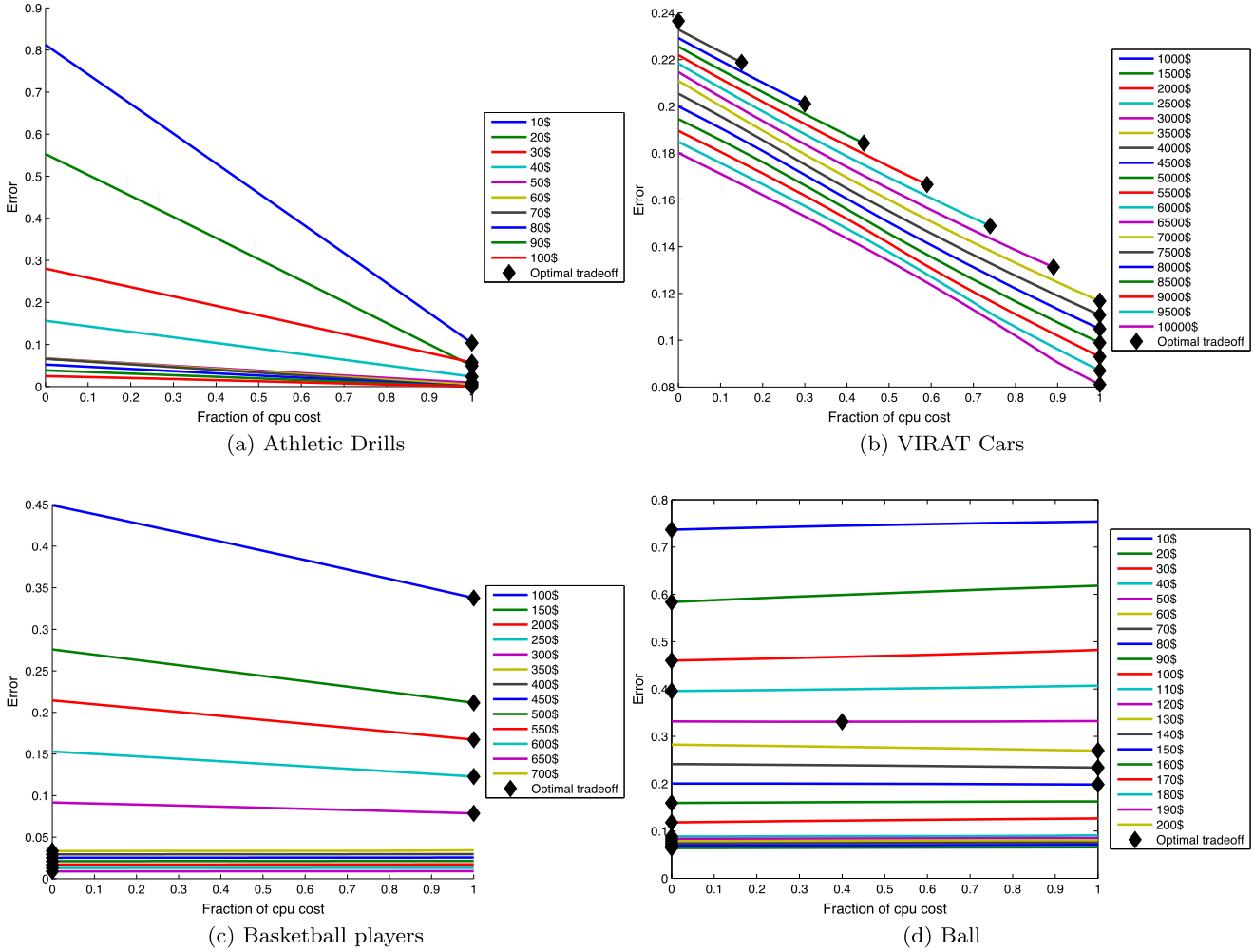


Fig. 16 Cost trade-off between human effort and CPU cycles. As the total cost increases, performance will improve. Cost axes are in dollars. *Blue* and *red* indicate low and high error respectively

high quality annotation allowing the basketball players to be annotated for $350 and the basketball for $200. Clearly, as tracking algorithms improve, the cost of a perfection annotation will decrease.

6.4 Interactive Evaluation Metric

Our motivation so far has been the use of crowdsource marketplaces as a cost-effective labeling tool. We argue that crowdsourcing also provides an interesting platform research on *interactive* vision. It is clear that the state-of-the-art techniques are unable to automatically interpret complex visual phenomena (yet). We hypothesize that allowing a modest amount of human intervention will allow us to successfully deploy vision algorithms *now*, allowing us to incrementally address and quantify progress for difficult scenarios. Traditional tracking metrics, such as time-to-failure, do not reveal the whole story: an approach may only need one more annotation in order to produce a perfect track. Indeed, the mere complexity of the data we analyze might be dwarfing our perception of progress in computer vision. To overcome this issue, we propose to fairly evaluate algorithms by how much they reduce annotation effort. We can quantify the performance of an algorithm by its economics: how much does this algorithm reduce the error for a fixed price point?



(a) Athletic Drills (b) VIRAT Cars

(c) Basketball players (d) Ball

Fig. 17 We show the cost trade-off between human effort and CPU cycles for different dollar amounts on different videos. In the "easy" field-drill video (**a**) and "medium" VIRAT data set (**b**), the optimal trade-off is to maximize CPU usage for a fixed dollar amount. Even though VIRAT has a significant amount of linear motion and stationary cars, our tracker is fast enough that it is still economical to execute. In the "very-difficult" ball footage (**d**), the optimal trade-off is to minimize CPU usage for a fixed dollar amount, essentially reducing to linear interpolation. Our most interesting tradeoff occurs for the "difficult" video of basketball players and referees (**c**). At current market prices, if nearly zero error is required, then linear interpolation is cost effective. But, if we can accept some error in our annotations, then running the dynamic programming based tracker is the most economical interpolation scheme

To demonstrate this, we analyze how our dynamic programming based tracker performs against other trackers. While there are many trackers available (for a survey, see Yilmaz et al. 2006), we will consider variations from our method by changing which features the visual object model employs. Figure 18 compares various aspects of our tracker: linear interpolation versus dynamic programming, color versus HOG descriptors, and the amount of spatial invariance in HOG. In all cases, the time-to-failure is fractions of a second, but our evaluation metric demonstrates that trackers that process pixel data with more robust features are, rightly so, superior. Interestingly, we show that HOG, as introduced in Dalal and Triggs (2005), is not optimized for tracking. This is because parameters are tuned for category-level invariance, but a tracking framework requires only instance-level invariance. Our metric reveals that the less spatially invariant HOG is superior for tracking.

Our cost-based analysis demonstrates that it pays to reduce the spatial invariance of HOG by computing histograms over smaller neighborhoods of 4×4 pixels rather than 8×8. To demonstrate this, we consider the cost of annotating the basketball game with 30 % error. The cost of executing the less invariant tracker is three times more expensive at \$10.20 (due to larger convolution kernel) for the entire basketball game. Therefore, to obtain 30 % error with HOG4, we must average 0.01 clicks per frame, costing \$59.47 in human labor for a combined cost of \$69.67. If in-

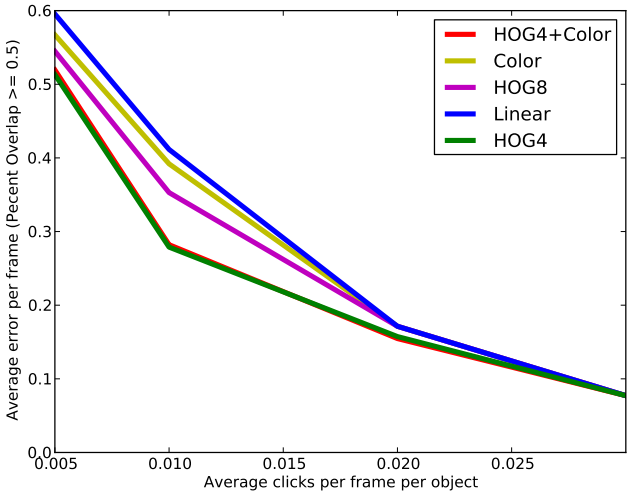


Fig. 18 A comparison of different trackers on a thousand frames of the basketball players video. HOG8 refers to the original HOG descriptor which computes bins gradient orientations over 8×8 pixel neighborhoods. HOG4 uses smaller 4×4 neighborhoods, and so is less spatially-invariant. Our method introduces a novel evaluation metric that more fairly evaluates trackers, as compared to traditional metrics such as time-to-failure. Our cost-based analysis reveals that it pays (literally!) to use smaller spatial neighborhoods, but it does not pay to use color

stead we choose to trade computation for humans and run the more spatially invariant HOG8 at the same error point, we only spend $3.40 on computers but $83.25 on humans with a cumulative cost of $86.64. In other words, HOG4 saves $17 over HOG8 on the basketball game when 30 % error is acceptable.

7 Future Work

Clients on the world wide web are distributed by nature and our framework would allow us to harness the unused processing power on the worker's computers. Instead of leaving the central server to do all the tracking, the workers' personal computers can execute the trackers for us. This approach allows us to not only buy the worker's time, but also some of the worker's CPU cycles. At the time of publication, however, running an intensive tracking algorithm is infeasible due to lacking efficiency in web browsers and JavaScript. As adoption of new web technology grows, we hope to explore this avenue further.

Even though our user studies demonstrate that fixed rate key frames are superior to user defined key frames, they are still sub-optimal since the annotation frequency is fixed per video. Instead, we should adopt an adaptive key frame schedule that adjusts the annotation frequency depending on the complexity of the video. If the object is stationary or easily recovered by a tracker, the annotation frequency should automatically decrease, but when the object undergoes unpredictable motion, the frequency must increase. We have laid the theoretical groundwork for active learning based video annotation in Vondrick and Ramanan (2011), but we have not addressed the user interface issues (a crucial area of vision often overlooked) surrounding active learning. As we believe active learning can provide a significant improvement over the results presented in this paper, we plan to integrate active learning into the system described here.

Finally, we believe that more research into compensation incentives can improve the quality of our video annotation system. In this paper, we began to explore non-standard payment schedules through our charity incentive, per-object bonus, and completion bonuses. We believe there are other compensation strategies—such as delayed payments with interest or lotteries—that will separate the "good" workers from the "bad" and "ugly." We hope to investigate these alternatives incentives in the future.

8 Conclusion

We have introduced a large scale video annotation platform capable of economically obtaining high quality labels for complex videos. We first built an efficient user interface for video annotation by informing our design choices through extensive user studies. Our experiments demonstrate, despite intuition, that contemporary interfaces are sub-optimal and that simplified, restricted interfaces can save significant effort. We next deployed our system on Amazon's Mechanical Turk where, instead of relying on the wisdom of the crowd, we are able to collect high quality labels by pinpointing a small group of expert workers who are capable of video annotation. By never rejecting work, we have built a relationship of trust with the workers. We then demonstrate that tracking algorithms benefit video annotation when under a constrained budget since they can exploit visual analysis to recover nonlinear paths. Our results reveal that the "Turk philosophy" does not hold for video annotation and computers should assist humans.

In order to reach the next generation of massive data sets, we cannot solely rely on low-wage crowdsourced marketplaces. Instead, we must also design intelligent annotation protocols that yield high quality and economical labels. In furthering this goal, we have presented a set of best practices, backed by three years of analysis, that shape our video annotation platform. We hope these contributions will spur the continued creation of massive video data sets and lead to innovation in data driven computer vision throughout the next decade. Indeed, data will always play a central role in computer vision research.

Acknowledgements We thank Sangmin Oh, Allie Janoch, Sergey Karayev, Kate Saenko, Jenny Yuen, Antonio Torralba, Justin Chen, Will Zou, Barış Evrim Demiröz, Marco Antonio Valenzuela Escárcega, Alper Aydemir, David Owens, Hamed Pirsiavash, our user study participants, and the thousands of annotators for testing our software and offering invaluable insight throughout this study. Funding for this research was provided by NSF grants 0954083 and 0812428, ONR-MURI Grant N00014-10-1-0933, DARPA Contract No. HR0011-08-C-0135, an NSF Graduate Research Fellowship, support from Intel, and an Amazon AWS grant.

References

Agarwala, A., Hertzmann, A., Salesin, D., & Seitz, S. (2004). Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics, ACM*, *23*, 584–591.

Ali, K., Hasler, D., & Fleuret, F. (2011). Flowboost–appearance learning from sparsely annotated video. In *IEEE computer vision and pattern recognition*.

Anonymous (2012). http://www.visint.org/.

Aydemir, A., Henell, D., Jensfelt, P., & Shilkrot, R. (2012). Kinect@ home: crowdsourcing a large 3d dataset of real environments. In *2012 AAAI spring symposium series*.

Bailey, B., & Konstan, J. (2006). On the need for attention-aware systems: measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, *22*(4), 685–708.

Bellman, R. (1956). Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, *42*(10), 767.

Buchanan, A., & Fitzgibbon, A. (2006). Interactive feature tracking using kd trees and dynamic programming. In *CVPR 06*, Citeseer (Vol. 1, pp. 626–633).

Chen, J., Zou, W., & Ng, A. (2011). Personal communication.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

Demiröz, B., Salah, A., & Akarun, L. (2012). Çevresel zeka uygulamalari için tüm yönlü kamera kullanimi multi-omnidirectional cameras for ambient intelligence.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). ImageNet: a large-scale hierarchical image database. In *Proc. CVPR* (pp. 710–719).

Endres, I., Farhadi, A., Hoiem, D., & Forsyth, D. (2010). The benefits and challenges of collecting richer object annotations. In *CVPR workshop on advancing computer vision with humans in the loop*. New York: IEEE Press.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

Fan, R., Chang, K., Hsieh, C., Wang, X., & Lin, C. (2008). LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Felzenszwalb, P., & Huttenlocher, D. (2004). *Distance transforms of sampled functions* (Cornell Computing and Information Science Technical Report TR2004-1963).

Fisher, R. (2004). The pets04 surveillance ground-truth data sets. In *Proc. 6th IEEE international workshop on performance evaluation of tracking and surveillance* (pp. 1–5).

Huber, D. (2011). Personal communication.

Kahle, B. (2010). http://www.archive.org/details/movies.

Kumar, N., Berg, A. C., Belhumeur, P. N., & Nayar, S. K. (2009). Attribute and simile classifiers for face verification. In *ICCV*.

Laptev, I., Marszalek, M., Schmid, C., & Rozenfeld, B. (2008). Learning realistic human actions from movies. In *IEEE conference on computer vision and pattern recognition, 2008. CVPR 2008* (pp. 1–8). New York: IEEE Press.

Liu, C., Freeman, W., Adelson, E., & Weiss, Y. (2008). Human-assisted motion annotation. In *IEEE conference on computer vision and pattern recognition, CVPR 2008* (pp. 1–8).

Liu, W., & Lazebnik, S. (2011). Personal communication.

Mark, G., Gonzalez, V., & Harris, J. (2005). No task left behind? Examining the nature of fragmented work. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 321–330). New York: ACM Press.

Mihalcik, D., & Doermann, D. (2003). *The design and implementation of ViPER* (Technical report).

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, *5*(1), 32–38.

Oh, S. (2011). Personal communication.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C. C., Lee, J. T., Mukherjee, S., Aggarwal, J. K., Lee, H., Davis, L., Swears, E., Wang, X., Ji, Q., Reddy, K., Shah, M., Vondrick, C., Pirsiavash, H., Ramanan, D., Yuen, J., Torralba, A., Song, B., Fong, A., Roy-Chowdhury, A., & Desai, M. (2011). A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR*.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.

Pirsiavash, H., & Ramanan, D. (2012). Detecting activities of daily living in first-person camera views. In *CVPR*.

Ramanan, D., Baker, S., & Kakade, S. (2007). Leveraging archival video for building face datasets. In *IEEE 11th international conference on Computer vision, 2007. ICCV 2007* (pp. 1–8). New York: IEEE Press.

Ross, J., Irani, L., Silberman, M. S., Zaldivar, A., & Tomlinson, B. (2010). Who are the crowdworkers? Shifting demographics in mechanical Turk. In *Alt.CHI session of CHI 2010 extended abstracts on human factors in computing systems*.

Russell, B., Torralba, A., Murphy, K., & Freeman, W. (2008). LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*(1), 157–173.

Schwartz, B. (2005). *The paradox of choice: why more is less*. New York: Harper Perennial.

Smeaton, A., Over, P., & Kraaij, W. (2006). Evaluation campaigns and trecvid. In *Proceedings of the 8th ACM international workshop on multimedia information retrieval* (pp. 321–330). New York: ACM Press.

Sorokin, A., & Forsyth, D. (2008). Utility data annotation with Amazon Mechanical Turk. *Urbana*, *51*, 61, 820.

Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: a large dataset for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(11), 1958–1970.

Torralba, A., Russell, B., & Yuen, J. (2010). Labelme: online image annotation and applications. *Proceedings of the IEEE*, *98*(8), 1467–1484.

Vijayanarasimhan, S., & Grauman, K. (2009). What's it going to cost you? Predicting effort vs. informativeness for multi-label image annotations. In *CVPR*.

Vijayanarasimhan, S., Jain, P., & Grauman, K. (2010). Far-sighted active learning on a budget for image and video recognition. In *CVPR*.

Vittayakorn, S., & Hays, J. (2011). Quality assessment for crowdsourced object annotations. In J. Hoey, S. McKenna, & E. Trucco (Eds.), *Proceedings of the British machine vision conference* (pp. 109–110).

Von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 319–326). New York: ACM Press.

Von Ahn, L., Liu, R., & Blum, M. (2006). Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 55–64). New York: ACM Press.

Vondrick, C., & Ramanan, D. (2011). Video annotation and tracking with active learning. In *NIPS*.

Vondrick, C., Ramanan, D., & Patterson, D. (2010). Efficiently scaling up video annotation with crowdsourced marketplaces. In *Computer Vision–ECCV 2010* (pp. 610–623).

Welinder, P., Branson, S., Belongie, S., & Perona, P. (2010). The multidimensional wisdom of crowds. In *Neural information processing systems conference (NIPS)* (Vol. 6, p. 8).

Xiao, J., Hays, J., Ehinger, K., Oliva, A., & Torralba, A. (2010). Sun database: large-scale scene recognition from abbey to zoo. In *CVPR*.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: a survey. In *Acm computing surveys (CSUR)*.

Yuen, J., Russell, B., Liu, C., & Torralba, A. (2009). LabelMe video: building a video database with human annotations. In *International conference of computer vision*.