Hands by hand: crowd-sourced motion tracking for gesture annotation

Ian Spiro, Graham Taylor, George Williams, and Christoph Bregler
Department of Computer Science, Courant Institute
New York University
{ian,graham,george,chris}@movement.nyu.edu
http://movement.nyu.edu/hands

Abstract

We describe a method for using crowd-sourced labor to track motion and ultimately annotate gestures of humans in video. Our chosen platform for deployment, Amazon Mechanical Turk, divides labor into HITs (Human Intelligence Tasks). Given the informational density of video, our task is potentially larger than a traditional HIT that involves processing a block of text or a single image. We exploit redundancies in video data in such a way that workers' efforts can be multiplied in effect. In the end, a fraction of frames need to be annotated by hand, but we can still achieve complete coverage of all video frames. This is achieved with a combination of HITs using a novel user interface, combined with automatic techniques such as template tracking and affinity propagation clustering. We show in a case study how we can annotate a video database of political speeches with 2D positions and 3D hand pose configurations. This data is then used for some preliminary analytical tasks.

Figure 1. An example annotation from the pilot dataset: Vladimir Zhirinovsky making an open-palmed, symmetrical, double-handed loop.

1. Introduction

This paper describes our efforts in utilizing Amazon Mechanical Turk (MT) for developing a video based gesture analysis system. There have been several efforts to use MT for various domains that need large example databases, such as natural language processing [15], and image labeling tasks [16], and recently for use with video of moving rigid objects [21].

We focus our efforts on acquiring annotations of human motion features like hand motion and complex 3D configurations like hand shape and 3D pose. Sometimes called match-moving, this is a typical task in high-end visual effects studios and an entire suite of internal and external tools [8, 9, 2] are used by professionally trained "tracking-artists." Academic researchers also have a need for this and high-end annotation tools [10, 13] have been developed and used by gesture annotation experts previously. In both cases, a large budget is usually necessary to annotate a small set of video.

In support of our research, we generally attempt to build systems that can extract patterns and statistics from large video databases, and can be used for various tasks in gesture analysis, to aid social and behavioral sciences, and for general search and visualization tools centered around understanding human motion. We previously developed a prototype system that can learn statistical models of individual "motion style" for subjects that are engaged in speaking actions. We have successfully applied this to a variety of tasks, including automated identification of personal speaking style, and to more general body motion styles [20]. That specific system uses no explicit representation of hand motion, or gesture types. This paper describes our efforts toward building a richer training database that contains detailed labels of hand motion and pose information.

The challenge is to transform a complex, labor intensive task such that it can be broken up into smaller units and completed by a variety of users without expert knowledge

of the domain. We need to avoid treating every individual frame as a task, as our database has on the order of millions of frames and this would be cost-prohibitive by any conservative estimate. We employ a staggered approach that alternates between human input and clustering techniques, such as affinity propagation [5] and pattern tracking techniques [11]. The UI attempts to replicate only the bare essentials of high-end tools. In some cases we introduce new ways of doing the task. This is a non-trivial user interface design challenge, targeting the interface to crowd-sourced workers such that it can be learned quickly with minimal training, but provides us with quality data.

This paper details the system design, which overall is a set of vision and machine learning tools running on a server, and Adobe Flash on the front-end, facing workers who have been sourced through Mechanical Turk. We also describe the design and deployment of HITs. This MT-based system complements our other research efforts that use fully automatic vision and statistical learning approaches to extract similarly meaningful representations from video data [20].

2. Related Work

2.1. Annotation Tools

The task of tracking the position of features or objects in a video, and "matching" the 3D pose and configuration of rigid and articulated objects in 3D is, in the visual effects industry, known as "match-moving" or "rotoscoping." Methods date back over 100 years when animators traced film on light tables to produce animations. Most recent high-end tools are based on supplying the user with an interface that is similar to 3D key-frame animation (such as seen in Autodesk Maya and other tools). The user can "scrub" the video back and forth, and can click on locations to set key-frames. The frames in between are either interpolated or semi-automatically tracked with general pattern trackers ([8, 9] to name a few). Most recent advances that are closest to our philosophy are techniques that blend hand annotations and automatic tracking and model estimation in an interactive way [3, 2]. Another community, which includes gesture and multi-modal communication researchers, use a different set of annotation tools. The most popular ones are ANVIL [10] and MacVisSTA [13]. Both tools are more targeted for time-based annotation of text tags, but have some capability of annotating spatial information. Neither tool permits annotation of 3D configuration information. All tools discussed so far have in common a high level of complexity that requires a user to undergo some training. In some cases the level of training needed to use these tools is extensive. This is not possible for MT users: they need to understand how to use an annotation tool in a very limited time. Another problem with the high-end tools is that they are generally not platform independent. For MT we don't want to require that users have a specific operating system. Therefore we employ a web-based interface primarily written in Flash.

LabelMe [14] and Sorokin [7] provide web-based toolboxes for image annotations. And most recently a video extension [21] has been reported. For our specific domain, these toolboxes do not provide the necessary functionality yet, since our annotations generally require handling of non-rigid objects and nuanced, high-speed motions.

2.2. Gesture Tracking

With the domain of human gesture tracking, there is a wide variety of approaches available that can automatically track and recognize human gestures and human motion (including our own work [19, 20]). It is beyond the scope of this paper to review all related tracking techniques and we refer to [4] for a survey. Fully automatic tracking of gestures is not solved yet for the general case. Part of this effort is to build a training database that will be used to develop and further improve such gesture tracking systems.

3. Our Approach

Our goal in this case study is to annotate videos of people while they are gesturing. This could include a wide range of motions involving any parts of the body, but for this initial attempt, we focus specifically on hand motions. We split this into two stages: 1) Annotate the locations of the hands in all frames, and 2) determine the pose and 3D configuration of each identified hand image. Figure 2 outlines our pipeline. We alternate between human based hand annotations and automatic tracking/clustering. As a preprocess, our videos first need to be cut into shorter sub-intervals, such that each HIT will require approximately the same amount of human effort. All videos remain on our server, and the human annotator communicates with the system through the Mechanical Turk site, which has embedded in it our own Flash based interface.

3.1. Embedded Flash User Interface

Flash provides access to a library of standard UI widgets. These widgets can be fully customized with Actionscript and combined with advanced logic to create new controls. Though the system requires a proprietary browser plugin, this plugin is available on all major consumer operating systems and behaves consistently across platforms. The main alternative to Flash is Javascript. For lightweight, HTML-oriented manipulations, this would be the preferred approach. But once the system becomes dependent on subtle, visual feedback– in particular things that would require advanced CSS properties in the dynamic HTML paradigm– the likelihood of breaking across platforms goes up. For the sake of rapid, research-oriented experimentation, it is ideal

to develop and test on a single platform. In the case of a Flash application, a user will either have the correct plugin or not. And in the latter case, they simply pass over the task and leave it for someone else. This is distinctly different from a traditional website that is intended to work for the maximum number of users and include fallback mechanisms, such as plaintext versions of a page for browsers that do not support a particular web technology.

3.2. Step 1: Tracking HIT

Our first goal is to locate the hands in all frames of the video. An exhaustive approach would be to show the user every individual frame and instruct her to click the center of each hand, advancing to the next frame after each click. But in the case of real human motion, the track of the hands may obey strong spatio-temporal constraints. There are typically spans of time where a hand stays right around a particular spot. In other time spans, a hand will track linearly between two key points. By adding a basic key-framing system to the interface, it is possible for the user to annotate a large number of frames with a relatively small number of clicks.

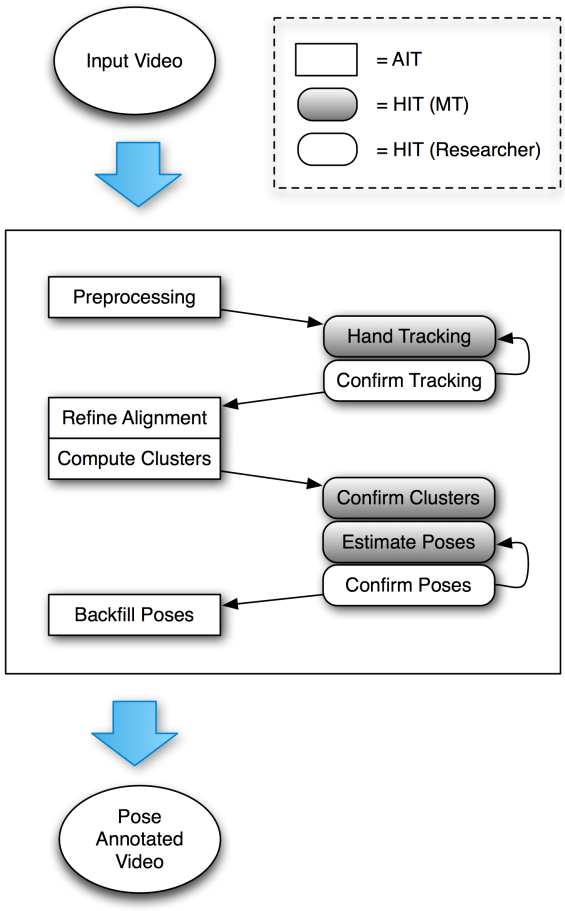

Figure 2. Information flow between artificial intelligence tasks and human intelligence tasks.

An early version of the interface had a set of clickable buttons for scanning back and forward along the movie's time dimension. If one attempts the annotation task, it is immediately apparent that having to move the cursor back and forth between these buttons and the main annotation panel is cumbersome. By linking keyboard input to the scan buttons, this problem is solved. The user can quickly skip ahead with the press of a key, while keeping the mouse fixed in its previous location– generally a good starting point for the next annotation click.

If the user can only skip ahead by a single frame, the user may begin to annotate every single frame, like in the original, exhaustive case. Another variation of the interface was designed that provides two sets of buttons and two corresponding sets of keyboard shortcuts. One set of controls skips forward and back by a single frame, while the other skips forward and back by ten frames. The users were instructed to use the skip buttons as appropriate, but ultimately to annotate with sufficient density to keep the hand within a bounding circle. Many users performed the task perfectly, adding additional keyframes where necessary to capture quick changes in position. Some users took full advantage of the skip-ten feature, and annotated exactly every ten frames, despite the instructions. Many motions are nonlinear in nature and if the user samples this with insufficient density, the tracking will be poor.

The simplest approach to improving annotations was to cut the skip interval in half. This tended to slow down power users but caused the average user to put in more keyframes.

In practice, the work performed by crowd-sourced labor needs to be validated by a person. For simplicity, this was performed 'in-house' by a researcher, though it could also be performed by MT in the future. In the case that tracking for a segment was insufficient, the work was rejected and

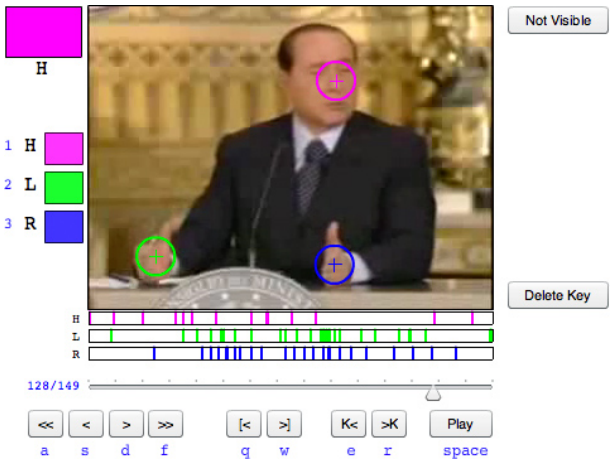

Figure 3. Our tracking interface for Mechanical Turk.

the frames were sent back into the MT queue for a second attempt at processing.

3.3. Step 2: Pose Clustering

After the tracking HIT, we are left with a collection of image patches, all representing the same subject's hands with a consistent camera, scene, and lighting. Given these consistencies, it is feasible to use simple matching techniques, such as normalized cross correlation (NCC). For all patches of a particular hand, we compute all NCC scores. This produces two results. First, by taking the maximum NCC within a window [11] two time-adjacent patches can be more precisely aligned. Second, these maximum scores can be used as rough affinity scores in a clustering approach. (Another option for matching are optical flow based or region based tracking techniques that perform a full affine warp or more complex warps, but in our experience those techniques frequently fail on complex hand gestures with small image support).

To compute clusters of similar patches, we use affinity propagation [5] on the matrix of NCC scores, taking advantage of Frey and Dueck's publicly available implementation. By setting high confidence thresholds, in a typical video sequence we can reduce a set of patches down to a set of templates that is a fraction of the original number of frames. We can now send these entire clusters to MT for evaluation in the next stage.

Steps 1 and 2 illustrate a key philosophy of the pipeline: Some tasks are better suited for people, while others are more appropriate for an algorithm. People are far better at accurately locating hands than any currently existing algorithm, but would find the clustering task performed by NCC and affinity propagation to be challenging and tedious.

3.4. Step 3: Pose HIT and Configuration Annotation

The next task is to determine the pose of every hand patch. Rather than sending every individual patch for evaluation, we send entire clusters as groups. The clustering is not perfect, so we also ask the user to first validate the cluster before matching the pose. The user can mark individual patches within the cluster as being incorrect, but then proceed to identify the pose for the remaining, correct patches.

One component of the pose-matching task concerns the positions of the fingers within the hand. Considering the raw dimensionality of finger position alone, it is a system of 14 joints, each with 1 or 2 degrees of freedom. It is not surprising that hand pose has such high dimensionality, given that ours hands' versatility is a key advantage of our species. At the same time, the input videos we are concerned with involve people speaking, using the hands only incidentally. To reduce the complexity of the finger positioning, we developed a set of base poses (as seen in the middle of figure 4). By separating finger position from overall hand pose, we can simplify the user interface. The remaining question is how to orient the hand in three-space to align it with the input pose. This can be achieved with three rotations.

The interface provides the user with several base pose buttons that represent finger positioning but not angle. After the user selects a pose, she can tune three knobs to rotate the hand into place. With each adjustment of a rotation knob, the user immediately sees the resulting image and can compare it directly to the input patch. Several options were considered for producing the images for this real-time feedback. One option was to take photos of a human subject's real hand in different, canonical positions. But this would require manual alignment and might not look consistent from frame to frame. Another option was to custom-build a rendering system in Flash to produce graphics. In this case, the time it would take for implementation was prohibitive. We opted instead to use Poser Pro software [6] to render a set of fixed poses. Poser is highly scriptable and using the built-in Python module it was possible to automate the positioning and export of many hand images. We created 7 finger poses, 10 axial rotations, and 7 side-side rotations for a total of 490 images. The third rotational dimension is obtained for free by simply rotating the final 2D image in plane using built-in Flash support including anti-aliasing. By importing the pre-rendered hand images into Flash, we create a Poser "Light" tool that is specifically catered to hand posing. Once again, the results of this task need to be validated by another person and for simplicity this was done by a re-

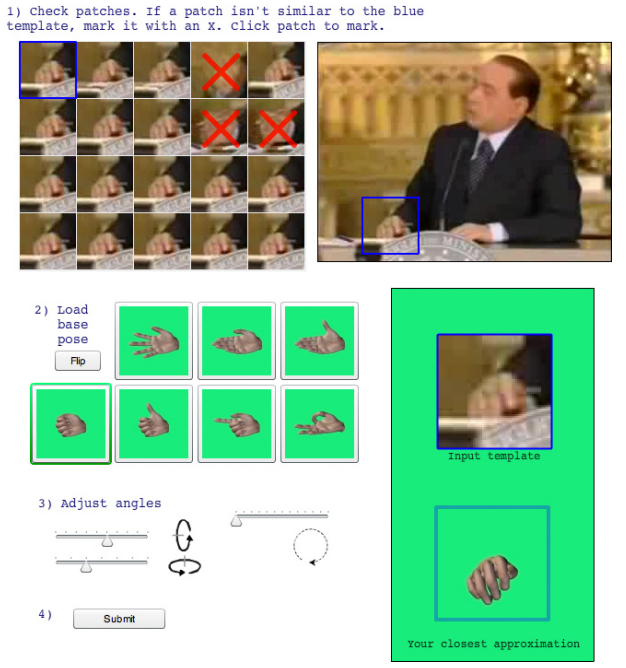

Figure 4. Our cluster verification and pose estimation interface for Mechanical Turk.

searcher, but could be handled by MT in the future.

3.5. Step 4: Post Processing Poses

After step 3, it is possible that many hand patches have not been estimated. If a hand was placed in the wrong cluster and identified as such by a user in the cluster validation phase, we do not know its pose. One approach would be to send any rejected patches back to MT for a second attempt. But this becomes expensive because now a full HIT must be devoted to a single patch. Our approach is to backfill the data using NCC-based post-processing.

For any unidentified patch, we compute the NCC distance to the image patch of each pose cluster center. We then can compute the weighted average of the closest K clusters using the NCC values and pose annotations of that specific cluster. (To compute the weighted average, we need to convert the pose angles into the twist representation for better interpolation properties [12]). Optionally we can do this on existing pose annotations for further fine tuning. Based on a preliminary sampling of 200 random frames, this technique had an accuracy of 77% correct pose estimation. We are currently using these predicted values in some experiments (see below), but do not use them as "ground-truth" training data.

4. Case Study

We previously collected a database of 189 videos of politicians and public figures giving speeches. Our efforts concerned processing all videos with automatic motion estimation and machine learning techniques (similar to bag-of-feature representations that do not need explicit labels, only coarse motion-style targets) and we got promising results for various recognition and clustering tasks [19, 20].

For the next data collection phase, we intend to fully annotate this database with hand locations and pose. At the time of this paper submission, our MT based system produced $23,667$ labelled video frames on a smaller pilot database of 10 videos. All our analyses in this case study are based on this smaller pilot database, but we expect to have the entire database annotated soon.

This pilot database was labeled with 158 "Tracking HITs" containing $6,769$ keyframe click annotations and 1612 "Posing HIT" annotations. We experimented with different pay rates for the two types of HITs, and got good results with $\$0.25$ for each 150 frame Tracking HIT and $\$0.05$ for each Posing HIT that contained on average 9.5 patches. The total cost was around $\$120$. If we had used a single frame approach, asking users to label the hand position and pose for each individual frame, this would require as many HITs as input frames. Assuming this task costs at least as much as a single Posing HIT, the total cost is more than $\$1000$ and annotating the full video database would cost over $\$10,000$.

Both, the low price of HITs and the good quality of the results surprised us. We had a savings of an order of magnitude over a frame-based MT approach or compared to using a professional service[1].

We have begun to use this data for two applications: for interesting visualization applications by domain experts, and to increase the annotation richness of training data for our automatic gesture tracking and classification system.

For assessment purposes, we built a preliminary suite of visualization tools that can reveal frame-based, gesture-based, and subject-based aspects of the pilot database. Figure 5 shows visualizations of the different "kinespheres" for different subjects. Figure 6 shows a few automatic matching examples of typical gestures in one example video to a specific subject or to the entire database. This allows us to analyze body-language in video more quantitatively, for example in counting the occurrence of specific gestures, poses, and other attributes in a speech.

[1] If we contracted with a professional studio, it would have taken each tracking artist between $4-20$ seconds for the tracking task of two hands per frame and $8-40$ seconds for the posing task [1]. At a rate of $\$50/h$, the pilot database would cost $\$6,000-\$30,000$.

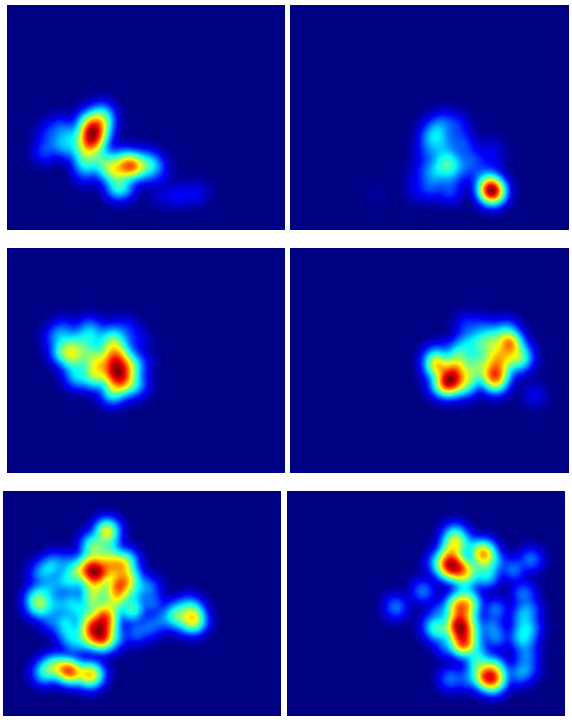

Figure 5. The hand gesture kinesphere for different subjects. (Video is not mirrored, so left hand kinesphere appears on the right and vice versa.) Barack Obama (first row) has a stronger bimodal distribution with his right hand. Hillary Clinton (second row) shows this effect more on her left. Vladimir Zhirinovsky (last row) has a wide distribution of gestures and the largest kinesphere.

Feature extraction and dimensionality reduction Dimensionality reduction is often used for visualization of high-dimensional datasets. In our case, we use such a visualization to observe structure amongst and within gesturing subjects. Two primary aims are to 1) note similarities and differences amongst subjects and 2) discover atomic "movemes" characteristic to, and shared amongst individuals. Both tasks can be performed by watching the (annotated) videos, but this is extremely time-intensive.

For each video frame, we first convert the output of the tracking and pose annotations into a 24-dimensional vector representing velocity, pose orientation (twist), and pose type (1-of-K encoding where $K = 7$) for each hand. While we could directly apply dimensionality reduction to this representation, we first perform a type of feature extraction that extracts an overcomplete, latent binary representation from the original data [17]. The reasoning for this step is twofold. First, our feature extraction method explicitly captures the dynamics of the gesture data. Each binary latent vector actually represents a local window in time. Secondly, the latent representation is *distributed*, where many different latent variables can interact to explain the data. This abstract representation is a more salient input to the dimensionality reduction step. For example, we do not have to worry about weighting velocity vs. pose type or orientation: this has already been figured out by the dynamical model. Nor do we need to worry about how the heterogeneous input types (i.e. the velocity and twists are real-valued but the pose type is multinomial) affect dimensionality reduction. In our experiments, we use 100 binary variables and a third-order dynamical model. Therefore the output of the feature extraction stage are length-100 binary vectors for each (overlapping) window of 4 frames. To this representation, we then applied t-SNE [18] to reduce the vectors to dimension 2 (Figure 7). We note that while person labels have been used in preparing this plot (to add color), both feature extraction and dimensionality reduction are completely unsupervised.

Figure 6. Example gesture matches within subjects and across subjects. Each column has similar gestures. Any window of motion can be compared to all other windows of the same length using an L2-distance. By applying non-maximal suppression, we find good candidate matches, that can then be verified visually.

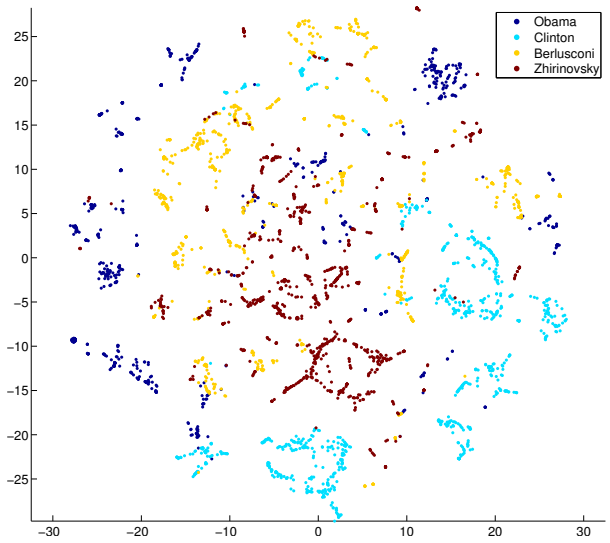

Figure 7. Dimensionality-reduced annotations. Data points represent overlapping 4-frame windows. We note that frames of a particular subject tend to cluster together. At a finer grain, we note the emergence of "strokes" in 2D which correspond to spatially and temporally local "movemes" in video.

5. Deployment at the Snowbird workshop

We recently deployed our system as part of an effort to analyze the nonverbal communication of academics. We digitally recorded all of the contributing and invited speakers at the Learning Workshop, held in Snowbird, Utah between April 6-9, 2010. We recorded 28 speakers, with talks ranging from 10-40 minutes each. After each block of speakers, video excerpts were chosen, then sent into the first section of the pipeline described in this paper. (We did not run the pose estimation stage.)

We were able to obtain accurate hand and head tracks for each of the speakers within hours of their talks. This allowed us to perform a simple analysis of the academics which included 1) clustering motion tracks; 2) ranking speakers based on the "energy" exhumed in each talk; and 3) locating specific gestures (e.g. hand-waving) across speakers. We presented our analysis during the concluding session of the workshop, to demonstrate the automation and quick turn-around of the system. This experiment demonstrates the flexibility and efficiency with which data can be collected for "on-site" gesture analysis using crowdsourced motion tracking.

6. Discussion

The system here described is a proof of concept for optimizing large, video-based annotation tasks. Our pilot study shows the potential multiplier effect obtained by combining human intelligence and artificial intelligence in a single pipeline. By proceeding with experiments on our entire motion database, we hope to show that we can get even better than a linear speedup. If an input video is large enough, we believe performing hand tracking on a small part of the input, will provide enough training data to build an automatic detector for filling in other frames. Likewise, as input size increases, pose clusters will grow, and it will take less human effort per pose on average.

For the next iteration, we envision a more generalized pipeline in which certain tasks can be attempted interchangeably by an artificial or a human intelligence. For example, an AI hand detector could attempt to locate the hands in a video frame, and only if the confidence value is low, does that particular task need to be sent to a person. Both HIT and AIT results could go to the same approval queue. As the model's training and performance increases over time, human effort can be reallocated toward approval tasks, away from the more time-intensive tracking and pose identification.

7. Acknowledgements

We would like to thank Kirill Smolskiy for his help in collecting sample videos and the Office Of Naval Research (ONR N000140910789, ONR N000140910076), Autodesk, and Google for supporting this research.

References

[1] High-End VFX Studio Time Estimates for Match-Moving. 2010. Private communication. 5

[2] C. Bregler, K. Bhat, J. Saltzman, and B. Allen. ILM's Multitrack: a new visual tracking framework for high-

Figure 8. An example search in the Snowbird dataset for a hand-waving gesture, which searches data produced a few hours prior.

end VFX production. In *SIGGRAPH 2009: Talks*, page 29. ACM, 2009. 1, 2

[3] A. Buchanan and A. Fitzgibbon. Interactive feature tracking using kd trees and dynamic programming. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2006. 2

[4] D. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan. Computational Studies of Human Motion: Part 1, Tracking and Motion Synthesis. *Foundations and Trends in Computer Graphics and Vision*. 2

[5] B. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972, 2007. 2, 4

[6] http://poser.smithmicro.com/poserpro.html. Poser Pro: Complete 3D Figure Design and Animation. 4

[7] http://vision.cs.uiuc.edu/annotation/. Vision At Large: large scale data collection for computer vision research. 2

[8] http://www.adobe.com/products/aftereffects/. Adobe After Effects. 1, 2

[9] http://www.vicon.com/boujou/. Bojou Matchmoving Software by Vicon. 1, 2

[10] M. Kipp. Anvil-a generic annotation tool for multimodal dialogue. In *Seventh European Conference on Speech Communication and Technology*. Citeseer, 2001. 1, 2

[11] J. Lewis. Fast normalized cross-correlation. In *Vision Interface*, volume 10, pages 120–123. Citeseer, 1995. 2, 4

[12] R. M. Murray, Z. Li, and S. S. Sastry. *Mathematical Introduction to Robotic Manipulation.* CRC Press, Baton Rouge, 1994. 5

[13] R. Rose. *MacVisSTA: A System for Multimodal Analysis of Human Communication and Interaction*. PhD thesis, Citeseer, 2007. 1, 2

[14] B. Russell, A. Torralba, K. Murphy, and W. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008. 2

[15] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008. 1

[16] A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. *Proc of First IEEE Workshop on Internet Vison at CVPR 2008*. 1

[17] G. Taylor, G. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In *Adv. in Neural Inf. Proc. Sys.*, pages 1345–1352, 2007. 6

[18] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[19] G. Williams, C. Bregler, P. Hackney, S. Rosenthal, I. McDowall, and K. Smolskiy. Body signature recognition. Technical report, TR-2008-915, New York University. 2, 5

[20] G. Williams, G. Taylor, K. Smolskiy, and C. Bregler. Body motion analysis for multi-modal identity. In *Int. Conf. Pattern Recognition*, 2010. 1, 2, 5

[21] J. Yuen, B. Russell, C. Liu, and A. Torralba. Labelme video: building a video database with human annotations. *ICCV, Kyoto, Japan*, 2, 2009. 1, 2