Video Retrieval by Mimicking Poses

Nataraj Jammalamadaka
IIIT-Hyderabad, India
nataraj.j@research.iiit.ac.in

Andrew Zisserman
University of Oxford, UK
az@robots.ox.ac.uk

Marcin Eichner
ETH Zurich, Switzerland
marcin.eichner@vision.ee.ethz.ch

Vittorio Ferrari
University of Edinburgh, UK
ferrari@vision.ee.ethz.ch

C. V. Jawahar
IIIT-Hyderabad, India
jawahar@iiit.ac.in

ABSTRACT

We describe a method for real time video retrieval where the task is to match the 2D human pose of a query. A user can form a query by (i) interactively controlling a stickman on a web based GUI, (ii) uploading an image of the desired pose, or (iii) using the Kinect and acting out the query himself. The method is scalable and is applied to a dataset of 18 films totaling more than three million frames. The real time performance is achieved by searching for approximate nearest neighbors to the query using a random forest of K-D trees. Apart from the query modalities, we introduce two other areas of novelty. First, we show that pose retrieval can proceed using a low dimensional representation. Second, we show that the precision of the results can be improved substantially by combining the outputs of independent human pose estimation algorithms. The performance of the system is assessed quantitatively over a range of pose queries.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content analysis and indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval;
H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; I.4.9 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Experimentation, Performance

Keywords

Human pose search, Video processing

1. INTRODUCTION

The goal of this paper is the real time retrieval of human poses from a large collection of videos. The last decade has seen considerable progress in 2D human pose (layout) estimation on images taken in uncontrolled and challenging environments. There are now several algorithms available [1, 4, 16, 22] that can detect humans and estimate their poses in typical movie frames, where humans can appear at any location, at any scale and wear clothing of varying sizes, colors and texture; the illumination can vary and the backgrounds can be cluttered.

There are numerous reasons why detecting humans and obtaining their pose is useful. A fundamental one is that often the pose, or a pose sequence, characterizes a person's attitude or action. More generally, applications range from video understanding and search through to surveillance and games. In this paper we enable for this first time the real time retrieval of poses in a scalable manner. Being able to retrieve video material by pose provides another access mechanism to video content, complementing previous work on searching for shots containing a particular object or location [20], person [2, 12, 19], or action [3, 11].

The real time system we propose is applicable on top of most 2D pose estimation algorithms. The only condition is that their output can be mapped into the simple pose representation we use, which enables rapid Euclidean distance based nearest-neighbor matching. We demonstrate our system here on the output of two existing pose estimation algorithms, Eichner & Ferrari [4] and Yang & Ramanan [22]. The functionality of the system is illustrated in figure 1, which shows the three query modalities that we have developed and the type and quality of the output.

While an early system for pose search was first proposed by [10], in this paper we extend the idea and methods substantially. First, we introduce a low dimensional pose representation and approximate nearest neighbor matching, making our system much more efficient both in terms of computational cost and memory consumption. This enables real-time search in a large scale database (over 3 million frames). Second, our system has a better retrieval engine: by combining independent pose estimation algorithms, we can reject frames where the pose estimate is likely to be incorrect. This leads to higher precision in the top returns (but not higher recall). Third, our approach allows for new query modes: query-by-stickman (with an interactive GUI), and query-by-Kinect, in addition to the query-by-image of [10].

The following section overviews the design and functionality of the system and the rest of the paper then gives details of the various stages and their experimental performance.

ICMR '12, June 5-8, Hong Kong, China

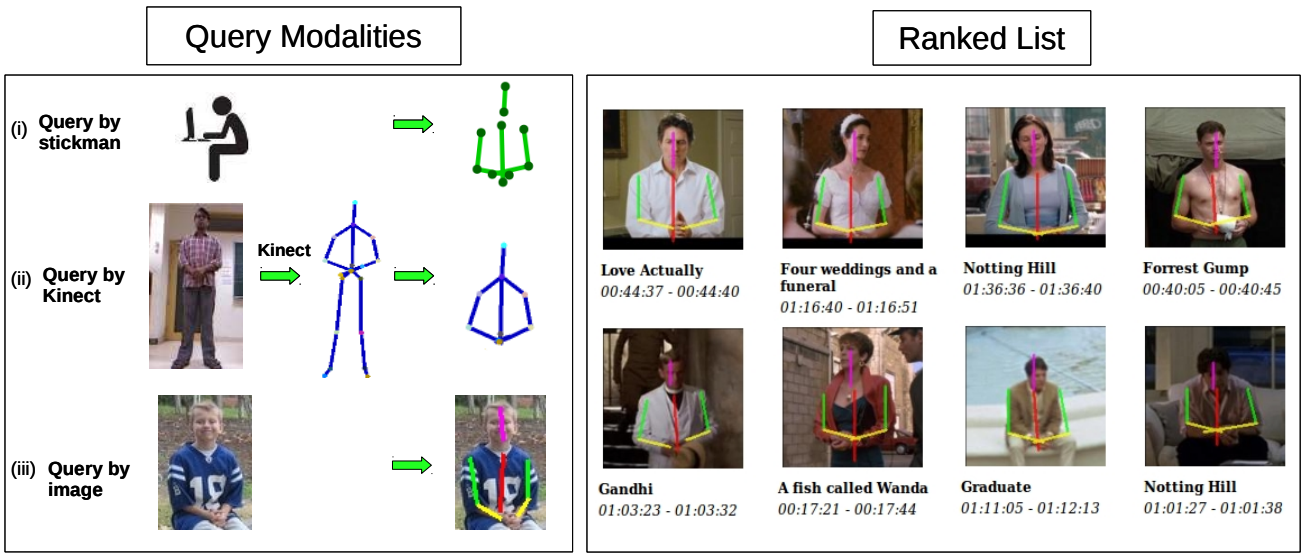


Figure 1: Pose retrieval overview. An illustration of the three query modalities: interactive stickman GUI, pose from Kinect, and pose estimated from an image. The output ranked list is obtained instantaneously as the query is varied. This can be tailored in various ways, for example to select only poses in different shots or within a particular movie. The results of the query pose are shown here at the video level.

2. SYSTEM OVERVIEW

We overview here the query modes and output of the retrieval system. The system is able to search over fifty thousand poses from 3 million frames of 18 movies in 5 ms, running on a single 2.33 GHz machine. To enable such real time performance all the processing steps, query and presentation of retrieved results need careful consideration. In particular much of the processing is carried out off-line with only the pose matching carried out at run time.

The following subsections describe the querying and output of the system and the principal processing blocks.

2.1 Query modes

For a user to query the system, we have developed three querying modes (figure 1). (i) The first is a so-called *stickman*. The interface is intuitive where the user can move the joints of a iconic stick figure freely and the pose retrieval system responds to these movements in real-time by retrieving the matching frames from the database. Imagine that the user is moving the stickman from the famous 'Titanic' pose (arms stretched side ways and parallel to the ground) to a 'Hands up' pose, passing through a series of intermediate poses. Then the pose retrieval system follows this sequence of poses, and continuously updates the results. This mode is very useful as a web-based application and can be deployed anywhere with a basic computer and an Internet connection. (ii) In the second mode, the user himself can move by waving his/her arms around and making different poses. The pose retrieval system tracks his movements using the Kinect sensor and displays the retrieved results in real-time. This mode is suitable for entertainment applications. (iii) Finally in the third mode, the user can upload an image of a person and then request the pose retrieval system to display similar poses in the database. For mobile application, this mode is very useful.

2.2 Output presentation

Ranked thumbnails of poses from the database are displayed according to their similarity to the query pose. Hovering over the thumbnail shows the full frame, and clicking on the thumbnail plays the video starting from that shot.

The user can choose the level at which to perform retrieval among (i) frame level, (ii) shot level, or (iii) video level retrieval. For example, shot level means that only a single result from each shot is displayed, whereas for frame level a number of results could be from the same shot. Thus shot and video level give more diverse results in terms of actors, scenes etc. The user can also choose whether to search the whole database or a particular movie. Retrieving from the whole database and at shot level are the default choices.

2.3 Off-line processing stages

Video processing.

Frames are grouped into shots, and an upper body detector is run on all the frames to detect people. The upper body detections are then grouped through the temporal sequence into a track. Finally two human pose estimation algorithms [4, 22] are run on each upper body detection in all the tracks, to estimate a stickman pose (section 3).

Data cleanup.

Unfortunately the precision of these pose estimation algorithms is not high and this significantly affects the perceived quality of the pose retrieval system. To improve precision, we propose a filtering stage based on the agreement of the two independent pose estimates from the algorithms (section 4). Poses which survive this filtering step are then considered for retrieval.

Pose representation and matching.

We have developed a low dimensional 12D vector pose representation suitable for matching two poses (section 5). The pose representation is crucial to the performance of the retrieval system as it simultaneously affects both the accuracy and speed of the system. The 12D vectors are recorded in a forest of randomized K-D trees for fast approximate nearest neighbor retrieval (section 6).

2.4 On-line processing stages

Pose retrieval.

Each query mode provides a 12D vector specifying the query pose. Nearest neighbor pose matches in the database are then obtained using the approximate nearest neighbor algorithm. This returns K approximate nearest neighbors, which are then sorted according to their Euclidean distance from the query vector. Depending on the chosen level of retrieval, information about the frames, shots and videos respectively are returned.

Client-server architecture.

For the retrieval system we use a standard client-server architecture (section 7). While the client interfaces with the user, the server performs the back-end operations of fast search and ranking, stores the randomized K-D tree structure, and serves the thumbnails, frames and videos.

3. DATA AND VIDEO PROCESSING

The videos are processed off-line over a series of three stages: (i) shot detection using a standard color histogram method [13]; (ii) upper-body detection to localize the people in each video frame and track them within the shot; (iii) human pose estimation (HPE) algorithms to determine the human pose within the upper-body detection area.

Data.

In the current system we have 18 videos, 17 are Hollywood movies and one is a season of the TV serial 'Buffy the Vampire Slayer'. In all, there are about 3 million frames. The statistics are given in table 1.

Video Name	♯Frames	♯Shots	♯Tracks	♯HPEs
Apollo13	192792	1795	9581	80567
About a Boy	138524	1411	8463	83413
*Buffy	318138	4232	24599	206047
Forrest Gump	204378	1101	7921	90216
*Four-wedding	185650	1177	12983	120029
Gandhi	263746	2117	13396	140045
Graduate	151027	498	938	78670
Groundhog Day	142411	838	1416	101342
*Living-in-Ob	129605	801	17719	94835
*Lost-in-Tran	146211	1046	21210	79256
Love Actually	193794	1990	11625	148566
*My-Cus-Vin	35299	303	241	19993
Notting Hill	171311	1586	8509	122643
Pretty Woman	172175	1165	24586	116540
Rainman	187345	1447	25463	105553
*Seeking-susan	148844	772	19538	61032
*Wanda	155116	981	19293	81294
Witness	161584	1234	769	55449
Total	3,097,950	24,494	228,250	1,785,490

Table 1: Dataset statistics. We consider 18 movies for the retrieval system. For each video, the number of images, shots, tracks and human pose estimates are reported. Movies with * are abbreviations for 'Buffy the Vampire Slayer' , 'Four weddings and a Funeral', 'Living in oblivion', 'Lost in Translation', 'My Cousin Vinny', 'Desperately Seeking Susan' and 'A fish called Wanda'.

3.1 Upper body detection and tracking

An upper body detector (UBD) is run on every frame of the video. An UBD algorithm detects and gives a bounding box around the people in the image. These are often a prerequisite for human pose estimation algorithms [1, 4, 17].

Here, we use the publicly available detector of [5]. It combines an upper-body detector based on the model of Felzenszwalb *et al.* [9] and the OpenCV face detector [21]. Both detectors run in a sliding window fashion followed by non-maximum suppression, and output a detection window (x, y, w, h). To avoid detecting the same person twice, each face detection is then regressed into the coordinate frame of the upper-body detector and suppressed if it overlaps substantially with any upper-body detection. As shown in [6] this combination yields a higher detection rate at the same false-positive rate, compared to using either detector alone.

In order to reduce the false positives, the detections in a shot are grouped into tracks and short tracks are discarded. These tracks are obtained using the temporal association strategy described in [10]. Visually, the tracks form a continuous stream of a person's bounding box in the shot.

3.2 Human pose estimation

We use here the algorithms of Eichner and Ferrari [4] and Yang and Ramanan [22]. Both of these have publicly available implementations. The algorithm [4] is run on each upper body detection in a track. The upper body detection is used to determine the scale of the person. The algorithm [22] is run over the whole frame to obtain multiple pose estimates. As many of the pose-estimates are false-positives, we use the upper body detections to filter them out. In detail, all detections returned by [22] which overlap less than 50% with any UBD detection are discarded. Overlap is measured using the standard "intersection over union" criterion [7].

Both methods are based on the pictorial structures framework [8], which models the body parts as variables in an energy minimization problem. Parts are parameterized by their image position. The energy function scores a configuration of parts, and contains unary and pairwise potentials. The unary potentials evaluate the local image evidence for a part in a particular position. The pairwise potentials carry priors on the relative position of parts. The model structure is a tree, which enables fast and exact inference. The algorithm infers the configuration of parts with the lowest energy.

Eichner and Ferrari [4] automatically estimate body part appearance models (color histograms) specific to the particular person and image being analyzed, before running the pictorial structure inference. This improve the accuracy of the unary potentials, which in turn results in better pose estimation. Their model also adopts the person-generic edge templates as part of unary potentials, as well as the pair-

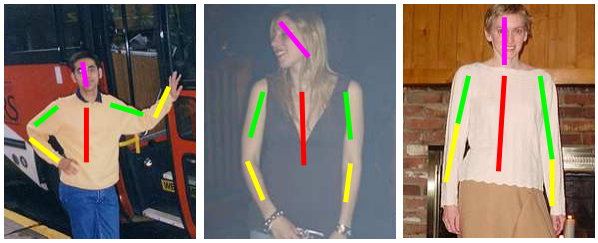


Figure 2: Pose estimates by HPE algorithms. The six upper body parts estimated by the algorithms are color coded as pink for head, red for torso, green for upper arms and yellow for the lower arms.

wise potential Ψ of [15]. The model has six parts, directly corresponding to physical limbs: head, torso, and upper and lower arms illustrated in the figure 2.

In Yang and Ramanan [22] instead, model parts are the mid and end points of the upper body limbs. Unary and pairwise potentials are learnt using an effective discriminative model based on latent SVMs [9]. This algorithm searches over multiple scales, locations and, implicitly, over rotations.

4. IMPROVING THE PRECISION OF POSE ESTIMATES

For the pose retrieval, we are interested only in correct pose estimates. That is, the pose estimates where all the parts are correctly estimated. A bad pose estimate (where one or more parts is wrongly estimated) can severely lower the perceived performance of the retrieval system if it appears high up in the ranked list. Here we use the assessment criterion of [4] and define a part as correctly estimated if its segment end-points lie within 50% of the length of the ground-truth annotation.

Unfortunately, for both the HPE algorithms the percentage of fully correct pose estimates, 9.1% and 14.4% (table 2), is very low. Note that we are applying a very severe test for a correct pose – that *all* parts are correct. In contrast, the quantitative measure used to assess performance in [4, 22] counts the *average* number of correctly estimated parts (PCP). For example, a pose with 5 out of 6 parts correctly estimated has a PCP score of $5/6 = 0.83$, but scores zero under our criterion.

Intersection of pose estimates.

In order to improve the percentage of correct pose estimates, we consider the agreement between the two pose estimation algorithms. The intuition behind this is that, for a given upper body detection, while the wrong pose estimates of both algorithms can be very different, the correct pose estimates of both the algorithms would be the same. Since the algorithms are based on different features and inference algorithms, they are rather complementary in their failure modes. Hence we expect that our agreement criterion should reliably identify the correct pose estimates.

We consider that two poses agree if the PCP between them is perfect (i.e. 1.0). Suppose the algorithms of Eichner and Ferrari [4] and Yang and Ramanan [22] generate pose estimates $A1$ and $A2$ respectively, then $A1$ and $A2$ are in agreement if the PCP between them is perfect (PCP=1.0), and if so the pair $(A1, A2)$ is added to the intersection set.

Method	Precision	Recall
Eichner and Ferrari [4]	9.1	12.1
Yang and Ramanan [22]	14.4	19.1
Intersection with PCP=1.00	41.1	3.5
Intersection with $PCP \geq 0.83$	16.7	5.1
Intersection with $PCP \geq 0.67$	7.8	5.5

Table 2: Improving precision using the intersection test. The table shows the precision and recall of HPE for 5000 random samples from the movie dataset. Using the intersection test, the recall values are low, but the precision is considerably improved.

Experimental evaluation.

To evaluate the 'Intersection of pose estimates' method, we compare the precision and recall of Eichner and Ferrari [4], Yang and Ramanan [22] and their intersection with respect to ground-truth. The aim of this evaluation is to ascertain if the intersection operation improves the precision. For the ground-truth, frames are sampled randomly from the movies dataset, of which 5000 frames with all body parts visible are manually annotated with a stickman. For each of these ground-truth frames, the upper body detector and the two pose estimation algorithms are run. Then the intersection operation described above is performed to obtain an intersection set S. We deem an element $e = (A1, A2)$ in the set S to be positive if both $A1$ and $A2$ are in full agreement (PCP=1.0) with the ground-truth stickman.

Table 2 shows that the intersection operation improves the precision significantly over those of the precisions of [4, 22]. Here precision is the total number of positives in S divided by the cardinality of S and recall is the total number of positives in S divided by the total number of ground truth stickmen. The precision improves from 14.4% to 41.1% while maintaining a tolerable recall. We also tried the variations of the intersection operation with agreement criterion of at least five parts ($PCP \geq 0.83$), and at least four parts ($PCP \geq 0.67$). As table 2 suggests, while reducing the PCP threshold marginally improves the recall, it worsens the precision significantly. Hence, in the implemented system we use PCP=1.0. Using this method, 54k poses are selected as the intersection set.

5. POSE REPRESENTATION AND MATCHING

We explain here the pose representation and matching components of the system.

5.1 Pose representation

From the human pose estimation algorithm, we obtain pose estimates consisting of line segments corresponding to the upper body parts namely head, torso, right and left arms. These depend on the size of the person and their location. To compare two pose estimates we require a representation that is invariant to scale and location in the image. Previous representations used for retrieval purposes were high dimensional, for example the three descriptors proposed by [10] have 15360, 1449 and 1920 dimensions. These descriptors are very high dimensional because they represent either full probability distributions over possible

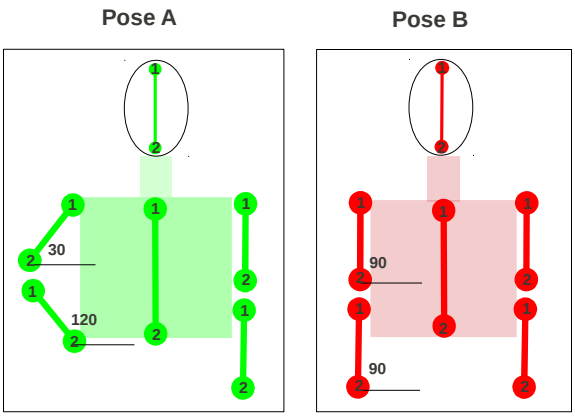


Figure 3: Pose Representation: The two poses A and B differ in two parts, the upper and lower left arms. The pose representation based on the angles clearly distinguishes pose A from pose B. The endpoints of each stickmen are consistently numbered.

part positions and orientations, or soft-segmentations of the parts.

In contrast, we use a simple and effective representation based on a single absolute orientation of each part. The orientation of parts are independent of the location and the size of the human and are not affected by variations in the relative length and positions of parts.

When comparing two poses, we would like to form a distance measure based on the sum of differences of angles of corresponding parts. To achieve this we encode the angle θ of each part as the 2D vector $(\cos\theta, \sin\theta)$. For six parts this results in a 12 dimensional feature vector. Then the Euclidean distance between two such representations gives the cosine of the angular difference. This is elaborated in more detail below. Figure 3 illustrates the pose representation.

5.2 Pose matching

Consider any part i of the poses A and B. Let the angles $(0^\circ \leq \theta \leq 360^\circ)$ be θ_A^i and θ_B^i respectively. We measure the dissimilarity between the poses of the parts of A and B as the negative cosine of $|\theta_A^i - \theta_B^i|$. The negation ensures that the dissimilarity monotonically increases with $\Delta\theta = |\theta_A^i - \theta_B^i|$.

In this work, we are interested in large scale nearest neighbor search. Most standard algorithms for this task are based on the Euclidean distance and require a feature vector for each sample. By encoding the angles as the vectors $\mathbf{v}_A = (cos(\theta_A^i), sin(\theta_A^i))$ and $\mathbf{v}_b = (cos(\theta_B^i), sin(\theta_B^i))$, $-cos(\theta_A^i - \theta_B^i)$ is obtained as the squared distance between the vectors i.e. $(\mathbf{v}_a - \mathbf{v}_b)^2 = 2(1 - cos(\theta_A^i - \theta_B^i))$.

6. POSE RETRIEVAL

After the database has been preprocessed off-line with the stages detailed above, it is ready to be searched by the user. The user enters a query (section 2.1) which is converted to the same 12-D representation as the poses in the database (section 5.1).

Experiment A: Recall, for $(m * 100)$ NN				
multiple m	1	5	10	20
Recall	10.0	76.0	88.7	95.2
Experiment B: Recall using N trees				
Num trees	1	2	5	10
Recall	94.1	95.2	90.1	85.4
Experiment C: Recall vs database size				
DataBase size	100K	250K	500K	
Recall	97.1	96.2	95.2	
Experiment D: Search speed ratio				
DataBase size	50K	500K	5.0 M	
Search speed ratio	1	11	123	

Table 3: Approximate nearest neighbor search vs Exhaustive search. The recall of the top 100 ground truth matches is averaged over 1000 queries.

For searching the query in the database we use the approximate nearest neighbor (ANN) method proposed by Silpa-Anan *et al.* [18]. The method has one of the best recall rates among algorithms that index high dimensional data with significant search speed gain over exhaustive search [14]. The method organizes the database as a collection of randomized K-D trees. To construct a K-D tree, the data is split at the median value of a pre-assigned dimension d and the two halves are passed down to the left and right subtrees. This procedure is recursively followed to further split the data. In [18] the splitting dimension is randomly chosen among the T dimensions with the largest variances. The technique constructs a set of randomized trees by running the random selection multiple times.

Given the randomized trees, the K nearest neighbors of a query are obtained as follows. The query point is sent to all of the trees. Each tree then returns a different set of approximate nearest neighbors (due to the randomness in the construction of each tree). First the query is passed down through all the trees to obtain the initial set of nearest neighbors. Then the search is systematically expanded into other nodes by backtracking from the leaf nodes and exploring the nearby alternative paths. This operation is efficiently performed by maintaining a list of nodes from all the trees in a priority queue. The node which is closest to the query is explored first. This procedure is repeated until K nearest neighbors are obtained. The nearest neighbors returned by the algorithm (with duplicates removed) are then sorted based on the Euclidean distance to the query.

Implementation details.

We discuss the important parameters of the number of trees used and the number of nearest neighbors returned from each below in the evaluation. To construct a forest of two K-D trees over a database of $54,000$ pose estimates requires 350 MB of memory and a retrieval time of $5ms$ for 2000 poses.

Experimental evaluation.

Here we evaluate how well the ANN algorithm performs compared to exhaustive search. To this end 1000 random pose estimates are sampled from the whole movie database as queries. Each query is searched in the database using both exhaustive and approximate nearest neighbor search. In the exhaustive search, the query pose is compared using the Eu-

clidean distance to all the elements in the database and the best 100 pose estimates are retained. Next, the search is repeated with the ANN algorithm. These ANN algorithms suffer from low recall. To address this, the standard practice is to retrieve more points and retain the desired number of nearest neighbors closest to the query. In this experiment, the desired number of nearest neighbors is 100 and multiples of 100, $100 * m$ *where* $m > 1$, are retrieved using ANN. The performance is measured using *recall at* 100, i.e. the proportion of the ground-truth nearest neighbors that are in the top 100 neighbors returned by the ANN algorithm.

Experiment A: In the first experiment, the recall of the ANN is observed while varying the multiple m. The multiple m is varied with values $\{1, 5, 10, 20\}$, but the number of trees is fixed to 2. As shown in table 3 ('Experiment A'), the recall at 100 rapidly improves with m.

Experiment B: In the second experiment the number of trees, N, is varied, but K is fixed at 2000. Thus on average, K/N neighbors are requested from each tree. As shown in table 3 ('Experiment B'), the performance is best for two trees.

Experiment C: In the third experiment the size of the database is varied, for $K = 2000$ and $N = 2$. As shown in table 3 the recall is largely unaffected by the database size.

Experiment D: In the fourth experiment the size of the database is varied, for $K = 2000$ and $N = 2$. As shown in table 3 ('Experiment D'), the speed gain over exhaustive search is orders of magnitude better and significantly improves with the size of the database.

In conclusion, following these empirical results, in our system we construct a forest of two trees, retrieve $K = 2000$ approximate nearest neighbors and take the top 100 from amongst these.

7. CLIENT-SERVER ARCHITECTURE

The system is implemented using a standard client-server architecture. The client interfaces with the user and the server performs the back-end operations. The functionalities of these components are described in detail for the stickman query mode, and summarized for the other two query modes.

7.1 Client

The client provides an interface for the user to interact with the system. It is responsible for taking the query from the user, sending it to the server and displaying the retrieved results returned by the server. Figure 4 shows the pose retrieval web demo that we have built. The upper-left corner shows the tabbed interface for three query modes query-by-stickman, query-by-Kinect and query-by-image respectively. In the query-by-stickman mode, the user can choose any pose by moving the arms of the interactive stickman in the image. As the user moves the stickman, the client continuously sends coordinates of stickman as queries to the server. The server responds in real time with the database poses that best match the query. This provides a gratifying and interactive pose search experience. In the query-by-Kinect mode as the user moves, the skeleton of the user output by the Kinect sensor is uploaded to the server. We observe that the Kinect's pose estimation is stable when the user is in full view of the Kinect. To further improve the stability, the skeleton is uploaded to the server only when the Euclidean distance between successive poses is greater than 5 pixels. In the query-by-image mode, the query image is uploaded to the server. The server then detects human poses in the image and relays back the best human pose to the client. The user is also given the option select the database and level (frame/shot/video) to retrieve at. The thumbnail of the matching result, the position of the shot and the movie to which it belongs are displayed. If the user clicks on any thumbnail, the corresponding video shot is played.

The client is implemented as a web application using the AJAX framework with javascript as the language. The client independently handles the user interaction, communication with the server and the results display. The input given by the user is passed onto the server as an XML request. Then the incoming XML data, a ranked list, from the server is processed. The corresponding thumbnails are requested from the server and displayed. To view the video clip of any retrieved result, the user is given an option to click on the corresponding thumbnail.

7.2 Server

The server processes a request from the user and returns the top ranked results that best match with the query. After receiving a query from the client, the server: (i) constructs the 12D pose representation, and (ii) retrieves the best matching results from the database and returns relevant information. Upon further request, it (iii) returns the video clip of the requested shot.

If the input is a stickman, the 12D representation is simply computed by measuring the orientation of the body parts. If the input is an image, the server applies the human pose estimation algorithms and then derives the 12D descriptor just as if it were a database video frame. If the input is a Kinect skeleton, the locations of the neck, head, base of the spine, two wrists, two elbows and two shoulders are used to construct the stickman (see the Kinect example in figure 1) The stick corresponding to the torso, for example, is constructed by forming a line segment with the end points as location of the neck and the base of the spine.

The server then retrieves 100 best approximate nearest neighbors. For each element in the ranked list the thumbnail path, the video name, the position of the shot to which it belongs to are sent back as result. If the user has selected shot or video level retrieval, then the nearest neighbors are grouped (on shot or video), and the single pose estimate most similar to the query in each group is returned.

8. EXPERIMENTAL EVALUATION

The pose retrieval system is quantitatively evaluated by posing 10 queries to the system and measuring the precision of the retrieved results on the first page of the web application, i.e. the top 12 returns. The queries are chosen to cover a diverse set of poses that people often make. Table 4 shows the queries posed to the system and the corresponding precisions, and figure 5 shows the first five retrieved results for the three top performing queries.

The precision for the best query is 91.7%, but the two worst queries have a precision of zero. The four queries with the highest precision are poses for which there are many examples in the database. Note how these include diverse poses, with arms next to the torso, overlapping with the torso, and stretched out from the torso. This illustrates the versatility of the system. The major impediment to the performance of the system at the moment is the failures of the pose estimation algorithms.

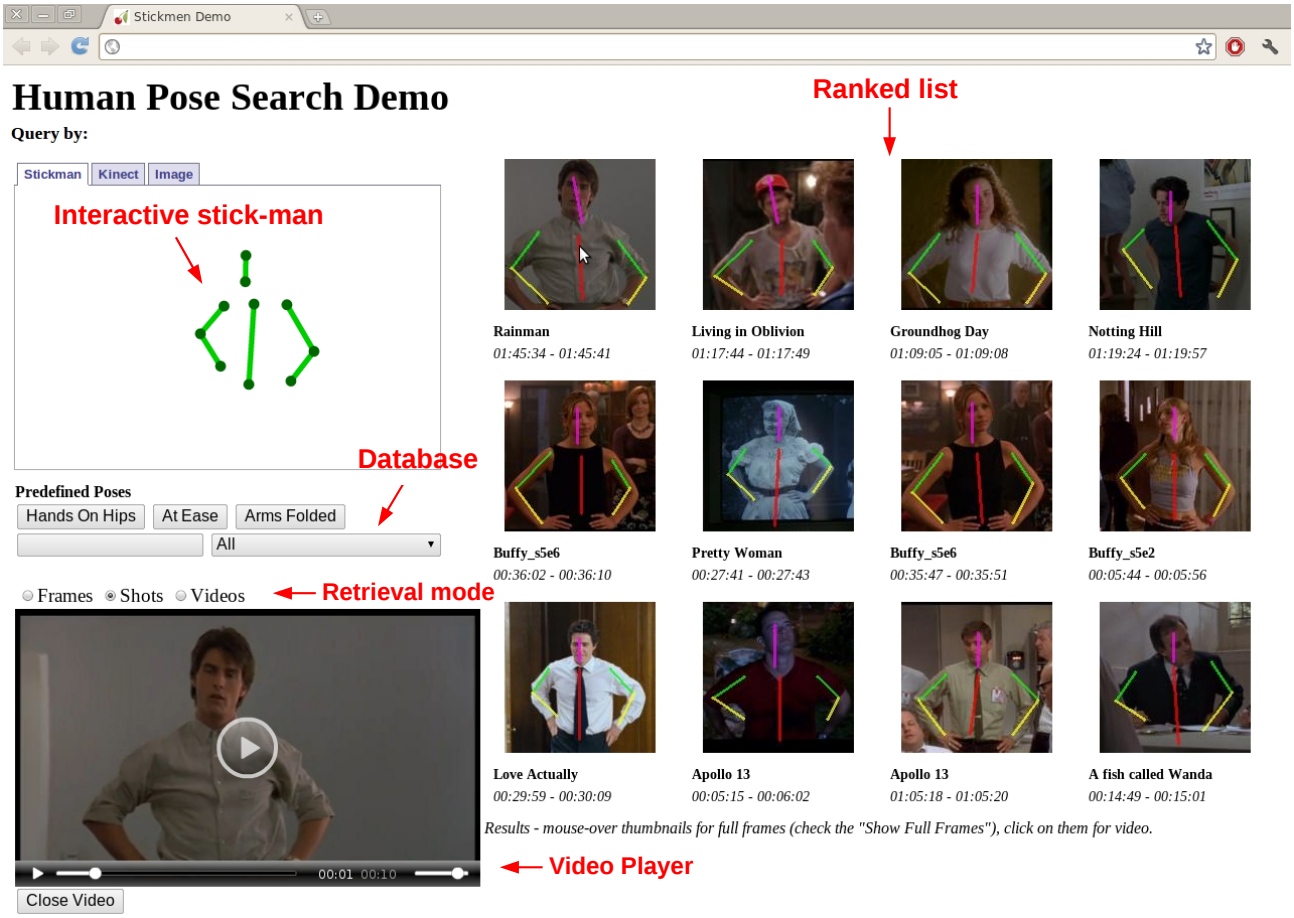


Figure 4: Screen-shot of the pose retrieval system. The elements of the web page such as the three query modes, options for selecting the database and the level of retrieval and video player, amongst other things, are indicated by text annotations and pointers in light red color. The *interactive stickman* can be moved around and the results are instantly updated as a *ranked list*. Clicking on a thumbnail plays a video of that shot.

9. CONCLUSION

We have designed a general, scalable, real time pose retrieval system for accessing frames, shots and videos. Even with the current state of pose estimation algorithms, with 50k examples there are many interesting poses that can be discovered and retrieved. We expect the performance of the system simply to improve over time as the performance of pose estimation algorithms improves.

Although we have demonstrated the idea for videos, a similar system could equally well be developed for large scale image datasets.

Acknowledgments

We are grateful for financial support from the UKIERI, and ERC grant VisRec no. 228180.

10. REFERENCES

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. *CVPR*, 2009.

[2] O. Arandjelovic and A. Zisserman. Automatic face recognition for film character retrieval in feature-length films. In *CVPR*, 2005.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005.

[4] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *BMVC*, 2009.

[5] M. Eichner and V. Ferrari. Calvin upper-body detector v1.03. `http://www.vision.ee.ethz.ch/~calvin/calvin_upperbody_detector/`, 2010.

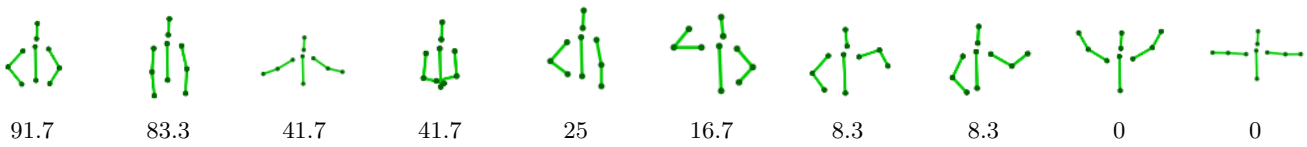
[6] M. Eichner, M. Marin, A. Zisserman, and V. Ferrari. Articulated human pose estimation and search in (almost) unconstrained still images. *ETH Technical report*, 2010.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[8] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005.

[9] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[10] V. Ferrari, M. Marin, and A. Zisserman. Pose search: retrieving people using their pose. *CVPR*, 2009.



91.7	83.3	41.7	41.7	25	16.7	8.3	8.3	0	0

Table 4: Pose retrieval evaluation: The poses displayed in the first row are used to evaluate the performance of the system. The numbers in the second row, are the precision of the top 12 results for that pose.

Figure 5: Pose retrieval examples

[11] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[12] P. Li, H. Ai, Y. Li, and C. Huang. Video parsing based on head tracking and face recognition. In *CIVR*, 2007.

[13] R. Lienhart. Reliable transition detection in videos: A survey and practitioner's guide. *International Journal of Image and Graphics*, 2001.

[14] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[15] D. Ramanan. Learning to parse images of articulated bodies. *NIPS*, 2006.

[16] B. Sapp, C. Jordan, and B. Taskar. Adaptive pose priors for pictorial structures. In *CVPR*, 2010.

[17] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. *ECCV*, 2010.

[18] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*, 2008.

[19] J. Sivic, M. Everingham, and A. Zisserman. Person spotting: Video shot retrieval for face sets. In *CIVR*, 2005.

[20] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[21] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 2004.

[22] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR*, 2011.