Using Unlabeled 3D Motion Examples for Human Activity Understanding

by

Ankur Gupta

B. Tech., The Indian Institute of Technology Kanpur, 2005
M. Sc., The University of British Columbia, 2010

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

July 2016

© Ankur Gupta, 2016
Abstract

We demonstrate how a large collection of unlabeled motion examples can help us in understanding human activities in a video. Recognizing human activity in monocular videos is a central problem in computer vision with wide-ranging applications in robotics, sports analysis, and healthcare. Obtaining annotated data to learn from videos in a supervised manner is tedious, time-consuming, and not scalable to a large number of human actions. To address these issues, we propose an unsupervised, data-driven approach that only relies on 3d motion examples in the form of human motion capture sequences.

The first part of the thesis deals with adding view-invariance to the standard action recognition task, i.e., identifying the class of activity given a short video sequence. We learn a view-invariant representation of human motion from 3d examples by generating synthetic features. We demonstrate the effectiveness of our method on a standard dataset with results competitive to the state of the art. Next, we focus on the problem of 3d pose estimation in realistic videos. We present a non-parametric approach that does not rely on a motion model built for a specific action. Thus, our method can deal with video sequences featuring multiple actions. We test our 3d pose estimation pipeline on a challenging professional basketball sequence.
Preface

This dissertation is based on the research work conducted in collaboration with multiple researchers at the Laboratory for Computational Intelligence at UBC.

A version of Chapter 3 has appeared in these two publications:


The author identified the problem, formulated the solution, and designed the experiments for both of these publications. For the CVPR paper, implementation was done by J. Martinez and the author. J. Martinez also contributed to problem formulation and drafting the manuscript. For the BMVC paper, A. Shafaei helped with the implementation, ran the experiments, and provided feedback on the mathematical formulation.

A part of Chapter 4 has appeared in the following publication:


The author contributed with identifying the challenge, formulating the solution, and implementing feature generation. J. He implemented the flexible matching (DTW-based) methods. The design and the data collection for the Video-based 3d motion Retrieval benchmark (V3DR) used in this paper was primarily done by J. Martinez and the author.

J. Little and R. Woodham contributed with ideas and provided feedback at all the stages of the project. They also edited all the above manuscripts.
# Table of Contents

Abstract .................................................. ii  
Preface ..................................................... iii 
Table of Contents .......................................... iv  
List of Tables ............................................... viii 
List of Figures ............................................. ix  
Glossary .................................................... xx  
Acknowledgments ........................................... xxi 

## 1 Introduction .......................................... 1  
1.1 Motivation ............................................ 3  
  1.1.1 Tackling dataset bias in action recognition ........ 3  
  1.1.2 Identifying and localizing human activities ........ 4  
  1.1.3 Utilizing unstructured data ....................... 5  
1.2 Contributions ......................................... 6  
  1.2.1 Cross-view action recognition via feature synthesis .... 6  
  1.2.2 Video-based mocap retrieval .................... 7  
  1.2.3 3d pose estimation in sports videos .............. 9  
1.3 Outline ................................................. 11
## 2 Background and Related Work

2.1 Cross-view action recognition

2.1.1 Reasoning in 3d space

2.1.2 Statistical approaches

2.2 Retrieval of human motion

2.2.1 Descriptors for pose and motion retrieval

2.2.2 Exemplar-based 3d pose estimation

2.2.3 Comparing and aligning temporal sequences

2.3 3d human pose estimation in videos

2.3.1 Statistical priors to human motion

2.3.2 Motion synthesis

2.3.3 Direct regression to 3d pose

2.4 Summary

## 3 Unsupervised Cross-View Action Recognition

3.1 Overview

3.2 Dense Trajectories from Mocap (MDT)

3.2.1 Generating multiple projections

3.2.2 Hidden point removal

3.2.3 Trajectory generation and postprocessing

3.3 Learning from synthetic data

3.3.1 Generating correspondences

3.3.2 Learning codeword transformations

3.4 Synthesizing cross-view action descriptors

3.5 Experiments

3.5.1 Dataset and evaluation

3.5.2 Dense trajectories for action classification

3.5.3 Mocap feature synthesis

3.5.4 Mocap retrieval-based augmentation

3.5.5 Results

3.6 Discussion

3.6.1 Limitations and the future work
6.2.3 Fine tuning pose estimation in videos

Bibliography
List of Tables

Table 3.1 Comparison of the overall performance of our approach and the state of the art on the IXMAS dataset. We show the accuracy averaged over all the camera pairs. We highlight the best value with boldface and underline the second best value. The results for the leave-one-subject-out evaluation mode, as well as the usual mode (common in the literature) are shown. Rahmani and Mian [RM15] build on the initial version of our method [GSLW14], but learn a non-linear transformation between mocap trajectory features from different viewpoints to a canonical view.

Table 4.1 Per-class and overall mean Average Precision (mAP) on the detection modality of the video-based mocap retrieval benchmark. We show an average performance using both IXMAS and YouTube queries. # ex. is the number of files in the database containing the given action. Chance corresponds to the expected performance of uniformly random retrieval. We highlight the best value in each category with boldface and underline the second best value. Again, we observe that using pose + motion (P+T), significantly improves the retrieval performance over motion-based features (T). Also, the flexible alignment techniques (SS-DTW and SLN-DTW) perform the best on most action classes in comparison to cross-correlation (CC).
List of Figures

Figure 1.1 A comparison of different tasks for localizing human activity in videos vis-à-vis the requirement of training data. We roughly measure the annotation cost as the number of clicks required per frame. Detection and tracking enable locating a person in the 2d space and has the least cost. A more complicated task of 2d pose estimation can be used for action recognition [JGZ+13] and video retrieval [EMJZF12], but the annotations are expensive. The 3d pose estimation task provides an even more detailed description of the activity, but we cannot obtain the annotations using a single view.

Figure 1.2 The cross-view action recognition problem. We show the kicking action as seen from two widely different viewpoints. Note that the projection of appearance as well as motion varies across cameras. If we train our model using the examples from Camera 1, it is not going to generalize well to Camera 2 videos. Therefore, we need an effective way to transfer knowledge across views.
Figure 1.3 Video-based mocap retrieval and alignment. We illustrate a flexible alignment between a short query video and a longer motion capture (mocap) sequence. The video matches only a part of the mocap, so the end-points of the two sequences do not align. The connections show the frame level correspondences. Note that the flexible alignment (many-to-one or one-to-many matches) ensures that motion at different speeds can be matched correctly.

Figure 1.4 3d pose estimation from a professional basketball sequence taken from a broadcast video. This is a challenging problem due to depth ambiguity, artifacts such as motion blur, and the complexity of player movements. We show two frames of a sequence with the corresponding 3d pose obtained by our system.

Figure 2.1 Commonly used evaluation strategies for cross-view action recognition. Each circle denotes a video, and solid circles represent labeled examples. The data from two different views is arranged in Source and Target columns. The boxes represent different action types. a) In correspondence mode, a fixed fraction of examples is known to be the same action seen from two views. The lines connecting the circles denote these correspondences. b) In semi-supervised mode, a small percentage of test (or target view) examples is annotated with a class label. c) The unsupervised mode is the most challenging case, where no annotations connect the source and the target examples. In this work, we demonstrate our results on the unsupervised modality only.
Figure 3.1  *Mocap dense trajectory* (Mocap Dense Trajectories (MDT))
generation pipeline. (a) Mocap sequences have 3d body joint
locations over time. (b) We approximate human body shape
using tapered cylinders to obtain a “tin-man” model. (c) Sam-
pled points on the surface of cylinders are projected under or-
thography for a fixed number of views and (d) cleaned up using
hidden point removal. (e) Connecting these points over a fixed
time horizon gives us the synthetic version of *dense trajectory*
(Dense Trajectories (DT)) features [WKSL11].

Figure 3.2  (a) A visual comparison between synthesized MDT (left) and
optical flow based DTs [WKSL11] (right). We exploit the sim-
ilarity in their shape to learn a feature mapping between differ-
tent views using MDT, and use this mapping to transform the
DT based features commonly used for action recognition. (b)
To generate view correspondence at the local feature level, the
human body is represented using cylindrical primitives driven
by mocap data. We project the 3D path of the points on the
surface to multiple views and learn how the features based on
these idealized trajectories transform under viewpoint changes.

Figure 3.3  All camera views from the IXMAS dataset. We show the kick-
ing action synchronously captured from the 5 camera angles.
Note that the appearance changes significantly across views.
Camera 4 is especially challenging because of its extreme ele-
vation angle.

Figure 3.4  Accuracy for each camera pair. We highlight the best results
in boldface and underline the second best value. Our mocap
feature synthesis approach performs the best on most train-test
view pairs.
Figure 3.5 Per-class classification accuracy of our method on the IXMAS benchmark. These are the same results as Figure 3.4 rearranged to show the per-class accuracy averaged over all train-test camera pairs. The mocap feature synthesis improves accuracy over the no augmentation baseline on every category except kick.

Figure 3.6 Confusion matrices before and after applying our mocap feature synthesis and retrieval-based augmentation approaches. We show average accuracy over all cameras as train-test pairs. We note that our data augmentation helps resolve confusion for many action categories.

Figure 3.7 Variation of the average classification accuracy (on IXMAS) with the number of divisions of azimuthal and elevation angles for our mocap feature synthesis approach. We note that a small number of divisions in angles does not provide the coverage needed for all the test viewpoints. However, the gain in accuracy saturates as we keep increasing the number of synthesized views.

Figure 4.1 Representing image and mocap sequences for retrieval. The mocap data and videos featuring human motion have related but complementary information. We bridge the gap between the two modalities by estimating the 2d joint locations from each video frame, and projecting 3d mocap joints for a viewpoint (right). We also generate optical flow based DT [WKSL11] from video and synthetic trajectories from mocap (see Section 3.2) to describe motion patterns in both the sequences (left). Hence, we have a pose as well as a motion-based representation for both the modalities that are comparable to each other.
Figure 4.2  Corresponding joints between the 2d pose estimate in an image and the CMU-mocap skeleton. We only use a subset of mocap joints. This mapping is similar to the one used by Zhou and De la Torre [ZDlT14] for matching DT to a motion model learned from mocap.

Figure 4.3  A toy example to illustrate various Dynamic Time Warping (DTW)-based flexible alignment algorithms. The grey boxes in the cumulative cost matrices $C$ represent the chosen warp path for the same distance matrix $D$ shown on the left. Each algorithm initializes some cells of $C$, fills the rest according to a recursion formula, and then chooses a final score $\text{dist}_r(v, z)$ for the alignment. The set of indices $\{(i - 1, j), (i - 1, j - 1), (i, j - 1)\}$ is abbreviated as $(i, j)^*$. In the case of SLN-DTW, $P(i, j)$ is the length of the chosen normalized warp path for the sub-problem up to $D(i, j)$.

Figure 4.4  A comparison between exact normalization (as defined in Equation 4.9), using dynamic programming over a 3d cumulative cost array, and approximate normalization used in SLN-DTW [MGB09]. The corresponding distance matrix ($D$) is shown behind each warp path, with darker shades representing smaller distances. We show results using (a-b) 2 different video queries matched against a mocap sequence and (c) two sequences of length 100 containing random, $L_2$-normalized 256-dimensional vectors. We notice that the approximate local normalization works fairly well in all three cases, while being several times faster. Note that we have kept the end-point constraint here for simplicity.

Figure 4.5  Example queries from YouTube videos provided in the v3DR dataset. Notice the realistic clothing and backgrounds that are not typical in videos collected in a laboratory setting.
Figure 4.6 An example of mocap annotations provided in the V3DR benchmark. At the top, we show a few frames of a mocap sequence. The corresponding action labels are shown below. Note that the annotations are not temporally exclusive. As shown above, one frame can have multiple labels. We use these annotations to evaluate video to mocap alignment.

Figure 4.7 Recall for different feature types. For the same descriptor length, relational pose features significantly improve recall over trajectory-based motion features. And, a concatenation of pose and motion features performs comparably or better than the individual features alone. The improvements in recall using Motion + Pose in the case of realistic data (youtube) indicates that when the 2d pose estimation is not reliable the motion information can be more useful.

Figure 4.8 We compare the performance of the cross-correlation (CC)-based retrieval with Circulant Temporal Encoding (CTE) for different values of regularization parameter ($\lambda$). We plot recall, averaged over all queries, for different values of $N$ (number of retrieved examples). Note that the recall improves for increasing $\lambda$, but CC performs better on most $N$ for both the query sets. This observation indicates that the regularization provided by CTE does not help the retrieval performance in our case.

Figure 4.9 Recall on the localization modality of the video-based mocap retrieval benchmark averaged over all YouTube and IX-MAS queries. The black dotted line depicts the ideal recall curve, and the magenta dotted line shows recall for randomly retrieved examples. These two curves act as the upper and the lower bound on the performance for each class. All matching techniques use motion + pose features except Gupta et al. [GMLW14]. Note that, flexible matching techniques (SS-DTW and SLN-DTW) perform better on most classes.
Figure 4.10 The confusion matrix for average recall (at $N = 100$) over all queries using $SLN-DTW + smooth$ for retrieval with $pose + motion$ features. Each row shows the recall@N for the queries from the action category, and columns depict the target category used to calculate recall. Therefore, the diagonal corresponds to the curves shown in Figure 4.9. Note the confusion of $sit down$ and $get up$ with $pick up$. This is due to the visual similarity of these actions. We also note that the categories with isolated body movement, e.g., $kick$, and $throw overhead$, are much harder to retrieve reliably. Also, the category $turn$ is challenging to distinguish, possibly because of the subtle change in the pose during the action compared to a significant change in case of $sit down$, $get up$, and $pick up$.

Figure 4.11 The confusion matrix for average recall (at $N = 100$) over all queries using $SS-DTW$ for retrieval with $pose + motion$ features. We observe a trend very similar to that shown in Figure 4.10.

Figure 4.12 A few representative alignments for YouTube videos (best viewed in color). The query frames and corresponding frames of the retrieved mocap sequences are shown (right limbs are marked in red). For each retrieval algorithm, we display the top ranked true-positive. Top: Walking sequences are relatively easy to match. All the algorithms perform well on this example. Middle: In this $pick up$ sequence, the flexible matching algorithms can capture the bend down and get up movements. However, $CC$ only aligns with the final get up movement. Bottom: Again, in this $kick$ sequence, we get a better alignment using the flexible matching techniques.
Figure 4.13 Some of the typical error cases for video-based mocap retrieval. We use SLN-DTW with Pose + Motion features for both the examples. We show the top three aligned matches along with the top-ranked true positive inside the green box. **Top:** In this case the *throw overhead* action is best matched to a dance move where the person has their arm lifted, similar to the query video. **Bottom:** The query comes from a *turn* sequence. Here the top ranked sequences are again dancing and walking. In both cases, we do find the appropriate matches, but they are poorly ranked.

Figure 5.1 Localized Motion Trellis (Localized Motion Trellis (LMT)). (a) The input is a monocular video sequence. (b) We use each overlapping subsequence to (c) search a large collection of mocap files for similar motion sequences. (d) The retrieved 3d snippets are connected in time to form a trellis graph. (e) The minimization of energy over this graph produces a smooth 3d output that best explains the image evidence. Being a model-free method, the LMT can estimate 3d motion in sequences with multiple activities, which overcomes one of the major limitations of current approaches.

Figure 5.2 Some of the common challenges with broadcast team sports videos. a) Even in case of a high-definition (HD) video, the player height in pixels is often less than 150 pixels in a wide-angle shot. b) There are motion blur artifacts due to the camera motion required to follow the game. c) Also, severe occlusions are common in team sports.
Figure 5.3  The LMT implementation details. (a) We assume that player bounding box is given at each frame. The red dot is our estimate of the player location based on the current bounding box. The blue line shows the estimated path connecting these locations over time. We transform this path to world coordinates using the homography. (b) We also use the homography at each frame to estimate the camera viewpoint. The left figure shows a square around a player projected to the image using the given homography (solid-magenta) and a local affine approximation to homography (cyan-dotted). Since the camera is located far from the court, the approximation is reasonably accurate in this case. We use the approximate affine transformation to obtain the elevation and the azimuthal angle of the camera under the orthographic projection (right).

Figure 5.4 Percentage of Correct Parts (left-right sensitive) for different variations of the LMT compared against Oracle-pose. Top match only uses the matching error score. Path only minimizes only the path error to find the best 3d sequence for each video snippet. Unary adds 2d pose error along with path and matching error. Full model uses a weighted sum of both unary and binary terms to evaluate the final LMT path. Oracle-pose has access to the ground truth 2d pose in addition to using the same parameters as the full model. We note that adding binary terms (in the full model) leads to a consistent gain in Percentage of Correct Parts (PCP). It demonstrates the importance of pairwise relations in resolving ambiguities.

Figure 5.5 Percentage of Correct Parts (PCP) for the LMT output compared to 2d pose estimation methods Flexible Mixture-of-Parts (FMP) [YR13] and nFMP [PR11]. Since FMP and nFMP do not distinguish between left and the right body parts, we do the same for the LMT projections to make the comparison fair. Again, the oracle-pose uses the same parameters as Ours but has access to ground truth 2d pose.
Figure 5.6 A typical 3d pose sequence generated using Localized Motion Trellis. The top row shows a cropped image sequence from the NBA data. The subsequent rows display the output of different methods, including our full mode (using both unary and binary terms). The 3d output corresponding to the full model is presented in the last row. Note that we have rotated the axis in the final row to emphasize the 3d nature of the output. The arrow on the middle frame shows the viewing direction of the camera. Also, we mark the right limbs in red. In this sequence, the player walks, then turns to his right and starts running. Our full model smoothly handles the transition from one activity to the next and accurately captures the walk cycles i.e., left and right legs are correctly aligned even though this information is not available from FMP. The unary output is affected by the errors in the nFMP pose estimate (see the fourth frame from the left and the last frame), while the full model can correct for these errors.

Figure 5.7 Another example of the 3d pose sequence generated using Localized Motion Trellis. In this sequence, the player runs towards the camera, then turns left. After waiting for a few seconds, he turns around and runs again. Note that the full model is correctly able to capture the walk cycle (see the last three frames), while the players direction is inconsistent in the unary output (see the sixth frame from the left, and third to the last frame).

Figure 5.8 An error case for our full model. In this example, the player runs, turns around, and keeps going in the same direction. He then stops for a few frames, and the turns left. Our 3d output from the full model faces in the wrong direction when the player turns and keeps facing in the wrong direction when the player is standing. This example is one of the cases when the full model may be failing due to over-smoothing. In contrast, the unary output is able to capture the change in direction.
Figure 5.9  Another error case emphasizing a common limitation of all the LMT variants. In this example, the player is performing fast and complicated movements. Even though the LMT can capture the overall jump motion (frames 2-6), the output does not capture any other movements. There can be multiple reasons for this failure such as a) the absence of an appropriate exemplar in the database that can reasonably approximate this motion; or, b) a failure in retrieval.

Figure 5.10  The effect of the LMT parameters on the average PCP (over the wrist, elbow, knee, and foot joints) for different thresholds. a) We plot accuracy as the function of query length. We reconstruct the 3d pose using 50 matches for each video subsequence. Since we are interested in measuring the effect of query length on the quality of retrieved mocaps sequences, we only use 2d pose error to find the best path in the LMT, and no interpolation is done to generate the final 3d output. Based on this result, we choose $k = 35$. b) Average PCP as a function of overlap between consecutive queries. In this case, we fix the query length to 35 frames and calculate PCP using the top 500 matches for each video subsequence. To avoid the effect of other terms, we use only transition error to find the best path.
## Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTE</td>
<td>Circulant Temporal Encoding</td>
</tr>
<tr>
<td>DT</td>
<td>Dense Trajectories</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>FMP</td>
<td>The Flexible Mixture-of-Parts model for 2d pose estimation [YRT13]</td>
</tr>
<tr>
<td>LMT</td>
<td>The Localized Motion Trellis proposed in this thesis</td>
</tr>
<tr>
<td>MDT</td>
<td>Mocap Dense Trajectories</td>
</tr>
<tr>
<td>MG</td>
<td>Motion Graph</td>
</tr>
<tr>
<td>PCP</td>
<td>Percentage of Correct Parts — a measure to evaluate 2d pose estimation</td>
</tr>
<tr>
<td>V3DR</td>
<td>Video-based 3d motion Retrieval benchmark</td>
</tr>
</tbody>
</table>
Acknowledgments

A Ph.D. is often considered a lonely pursuit. However, my graduate school experience couldn’t be any further from this stereotype. This thesis would not have been possible without the help and active involvement of many individuals, to whom I am greatly indebted.

I am grateful to my supervisors Prof. Jim Little and Prof. Bob Woodham for their constant support and encouragement. Bob inspired us to ask ourselves challenging questions. Jim often provided new and insightful perspectives on the problems we discussed. Our meetings were intellectually stimulating and fun. My advisory committee member Prof. Michiel van de Panne provided invaluable guidance throughout the process. Many thanks to Prof. Rick Wildes, the external examiner of the thesis, for his excellent feedback during the exam and for writing a detailed report that helped me improve the final manuscript. I also thank Prof. Jane Wang and Prof. Ian Mitchell for agreeing to be university examiners and carefully going through the manuscript.

Research can be rewarding but often it is a gruelling process. Thankfully, it gets easier working alongside talented individuals who are also passionate about the subject. The contributions of my collaborators are instrumental to the research presented in the following pages. Julieta Martinez and Alireza Shafaei worked hard on these problems with me and provided the much-needed feedback at various stages of the project. I am also grateful to summer research interns Umang Gupta and John He for their help. A special thanks to David Matheson who was the first to suggest the idea of using synthesized features to describe human motion.

I believe that a culture of curiosity and continuous learning is essential for research. I thank Jim for fostering this culture in the lab. Thanks to my labmates
for creating a great learning environment and making the lab a fun place to work. My roommates during this period, Nishant Chandgotia and Krishna Teja, ensured a similar stimulating environment at home. They taught me a large chunk of whatever little I know about topics outside computer vision — from mathematics to classical music to philosophy and religion. The circle of friends that I had at UBC is as priceless as the education and the training.

My gratitude for my family is difficult to express in words. I thank my parents and my brother for their unconditional love, patience, and encouragement. My wife Rashmi has been a great support throughout my Ph.D. journey, which can often be very demanding for a spouse; I thank her for navigating these challenges with grace and humor.
Chapter 1

Introduction

The motion of the body accounts for a large part of human expression. Our movements help communication and reflect intentions. To function safely and efficiently alongside a person, robots (or AI systems) must have the ability to understand the user’s activity. Therefore, building algorithms that can analyze human motion is crucial. Also, a notable percentage of video data “in the wild” — on YouTube, TV, and movies — features humans. Indexing and searching through this massive data would require paying particular attention to human movements. Due to its numerous applications, automated analysis of human motion is an important problem in robotics as well as computer vision.

Obtaining appropriate training examples is a key practical challenge in solving a majority of computer vision tasks. Using supervised machine learning on large human-annotated datasets has led to impressive performance on many fundamental vision problems such as object recognition and detection [KSH12, GDDM14]. However, adapting these methods to more complicated problems such as complex activity recognition in videos is challenging, partly due to lack of annotated data. It becomes more tedious and expensive to obtain these annotations as the task gets more complicated. Also, the labels can be subjective or ambiguous. In response, instead of relying on labeled examples, we propose to use a large set of unlabeled 3d human motion examples, in the form of motion capture (mocap) data. We focus on three related problems in human activity understanding — a) cross-view action recognition, b) video-based retrieval of human motion, and c) 3d pose estimation.
In the computer vision literature, *action recognition* refers to the task of classifying a short video clip into one of the predefined classes. These class labels denote specific activities in the video such as getting-out-of-the-car, playing-football, sitting-down, or getting-up. *Cross-view action recognition* adds viewpoint invariance to action recognition, i.e., it allows actions to be recognized under a diverse set of views. Human activities often have a hierarchical spatiotemporal structure. Hence, assigning a single label to the whole video is not very descriptive. Many techniques decompose activities into smaller primitives [PR14, LZRZS15]. We refer to this simultaneous decomposition and recognition of action as *complex activity recognition*. *Video-based mocap retrieval* is a new task that we introduce in this thesis. We define it as retrieving mocap snippets given a short video as a query. Mocap retrieval can be a handy tool for animators to search through a large database of mocap files. We also show its applications to human activity understanding. *3d pose estimation* is the problem of localizing the 3d joint locations of a person given a video. 3d pose is a detailed, well-localized description of human motion in space and time. Apart from being useful on its own (see Section 1.1.2), 3d pose can also serve as an intermediate feature for complex activity recognition [RF03]. Also note that we use *activity understanding* as an umbrella term that can include all the above tasks.

**What do we set out to achieve?**

We are interested in the detailed description of the human activity in monocular videos captured offline, such as YouTube clips or broadcast sports videos. The theme uniting all our contributions is that our approach only uses a large set of human motion examples as training data and does not require human labeling effort. Here we summarize the main goals of this thesis:

- We are interested in building a view-invariant model of human actions without using labeled or synchronized video examples. For quantitative evaluations, we use a standard benchmark for cross-view action recognition called the INRIA XMAS dataset [WRB06], where we compare our method with recently proposed unsupervised approaches such as [RM15].
• Our second goal is to search through a large number of mocap files with short video queries. We also wish to align the retrieved mocap snippets with the video to establish a one-to-one correspondence between 2d and 3d motion (see Figure 1.3). Since there is no standardized way of measuring success on this task, we assess video-based mocap retrieval on our own benchmark Video-based 3d motion Retrieval benchmark ($V^3DR$).

• Finally, we wish to obtain the 3d pose of basketball players using broadcast video as input (see Figure 1.4). We would like to solve this problem in a data-driven way, allowing us the flexibility of being activity independent (more details in Section 1.2.3). We evaluate our approach on a professional basketball sequence used previously to test player tracking and identity recognition [LTLM13].

1.1 Motivation

1.1.1 Tackling dataset bias in action recognition

Changes in imaging conditions lead to visual domain shifts in videos and images (i.e., shift in the distribution of features) [SKFD10]. One of the factors affecting action recognition performance is the viewpoint of the camera used to capture the video. Recent methods using unconstrained videos for action recognition rely on the internet or movies to obtain training data [KJG+11], but these sources may suffer from a bias in viewpoint. For instance, most YouTube videos are shot with people holding the camera at a similar angle and height (see Figure 4.5 for a few examples). However, many test scenarios in action recognition may involve cameras mounted on a wall or a robot, with a different viewing angles than the typical internet video.

Therefore, we need techniques that can generalize to a wider range of viewing angles not featured in the training data. Cross-view action recognition is crucial to deal with dataset bias in action recognition.
1.1.2 Identifying and localizing human activities

Action recognition only provides a single label to describe a video. Also, it is often tested on video clips rather than unconstrained videos. Recent efforts deal with this limitation by using untrimmed videos [GIJ+15] or by predicting a bounding box around the activity (i.e., action spotting [DSCW10]). In this thesis, we focus on 3d pose retrieval and estimation because it localizes human activity in space and time, which can potentially provide us the information needed to support other applications such as:

NUIs and healthcare. Natural User Interfaces or NUIs — systems using human motion and gestures to control devices — are already making headway in gaming and entertainment [SGF+13]. Effective human activity understanding can also have a positive impact on other aspects of life, including the general safety and well-being of people. For instance, a fall-detection system [NTTS06] for the elderly or a patient monitoring system can assist human caregivers, as well as help reduce the health care costs.

Sports and fitness. Similarly, personal training systems with an ability to provide feedback on posture can be very beneficial. Again, such a system would need to discriminate between subtle differences in human pose and motion. Analyzing sports videos for assessing player performance, refereeing, or commentating is another interesting application [SZI4b].

Robotics. Self-driving cars serve as an excellent use case for human activity understanding. A fully autonomous vehicle not only needs to observe the motion of pedestrians but also to anticipate their future actions, e.g., a child playing near the road may run after the ball onto the road. Similarly, the use of robots in a human environment, such as homes and offices, will not be practical unless the machines can take human motivations and safety into account. Another interesting application is training robots via learning by demonstration [AN04]. We would like robots to acquire skills by watching a person demonstrating the task live or in a video.
Figure 1.1: A comparison of different tasks for localizing human activity in videos vis-à-vis the requirement of training data. We roughly measure the annotation cost as the number of clicks required per frame. Detection and tracking enable locating a person in the 2d space and has the least cost. A more complicated task of 2d pose estimation can be used for action recognition [JGZ+13] and video retrieval [EMJZF12], but the annotations are expensive. The 3d pose estimation task provides an even more detailed description of the activity, but we cannot obtain the annotations using a single view.

1.1.3 Utilizing unstructured data

State-of-the-art activity understanding methods rely on human-annotated data and often use one label per sequence (e.g., [KTS+14]). If we wish to localize actions in time, we need per-frame annotations of actions, which are much harder to acquire. Similarly, localizing activities in space is challenging because it requires detailed labels in each frame of the video [ZZD13]. Human time and effort required for obtaining labeled training data become heavier as we learn more details (see Figure 1.1). Also, annotating training data with suitable labels is often ambiguous and hard to scale to new labels.

Motivated by these challenges, we explore the use of unstructured motion examples (in the form of mocap files with no labels) to address some of the challenging problems in human activity understanding.
1.2 Contributions

1.2.1 Cross-view action recognition via feature synthesis

Our first contribution is a method for adding viewpoint invariance to action recognition without using any additional labeled examples or multi-view videos. Shape [DT05, GBS+07, YS05] and optical flow-based [DTS06, WKSL11] features that are commonly used to describe actions in videos are not viewpoint invariant by design. Therefore, the effectiveness of a method depends heavily on the availability of training data from diverse views. When the training view (or source view) is different from the test view (or target view), we need a strategy to either transfer knowledge across views or devise a view-invariant description. These methods to achieve view-invariance are often referred to as cross-view action recognition techniques (see Figure 1.2).

Many cross-view action recognition approaches transform action descriptors to a view-invariant space where observations from the source and the target view are comparable [LZ12, LCS12, MT13, ZJ13, HW13]. However, learning an invariant space requires supervision in the form of correspondence or partially labeled examples in another view (see Figure 2.1(a-b)). A few methods deal with the unsupervised case [LZ12, ZWX+13, LCS12], but all of them assume the availability
of multi-view videos that can be hard to acquire in a general case.

We present a scheme to deal with the completely unsupervised case, i.e., we have no access to the target view examples at the training time. This is a likely scenario when it is not possible to predict the test view in advance. Instead of looking for a view-invariant space, we learn a function to transform descriptors from one view to another using unlabeled mocap examples. For action recognition, we describe each video with a bag of words (BoW) of Dense Trajectories (DT) features [WKSL11]. DT are optical flow-based features commonly used for action recognition in unconstrained videos. Given multiple feature mappings (learned transformations) and training videos from a single view, we can “hallucinate” action descriptors from different viewpoints and use them as additional training examples for classification. We augment the training data with synthesized descriptors to make our model resilient to viewpoint changes.

In summary:

- To learn correspondence between features across different views, we synthetically generate motion features using mocap examples as seen from multiple viewpoints.
- We show, due to the similarity between synthesized features and real trajectories [WKSL11], we can learn the view transformations on mocap and apply it to videos (Figure 3.2(a)).
- Our relatively simple feature transformation and data augmentation technique improves the action classification accuracy over the baseline with no augmentation. Our approach is also competitive with a non-linear feature mapping method [RM15] proposed recently.

1.2.2 Video-based mocap retrieval

Next, we focus on finding an efficient method for alignment and distance computation between human motion sequences across two modalities — video and motion capture. This problem typically occurs when searching a large database of mocap files. Often mocap examples contain an actor performing multiple activities (e.g., the actor may get up, walk, run and kick in a martial arts sequence). However, we
Figure 1.3: Video-based mocap retrieval and alignment. We illustrate a flexible alignment between a short query video and a longer motion capture (mocap) sequence. The video matches only a part of the mocap, so the end-points of the two sequences do not align. The connections show the frame level correspondences. Note that the flexible alignment (many-to-one or one-to-many matches) ensures that motion at different speeds can be matched correctly.

want to search for a single action as depicted in a short video query. In a nutshell, we wish to a) retrieve mocap files with similar actions and b) align the video query to the relevant portion of the retrieved mocap sequence.

A search through existing mocap files can save the effort of collecting the data again. Also, we can use similar, aligned mocap sequences and blend them to generate new animations satisfying higher-level goals and constraints [KG04]. This retrieval-with-alignment task can also be a crucial step in various vision applications such as 3d human pose estimation [RSH+05] and cross-view action recognition [GMLW14].

We can rely on rigid matching (i.e., one-to-one frame correspondence) to align sequences efficiently [RDC+13], but we hypothesize that due to the style and the speed variations in human motion, flexible temporal alignment is a better approach, and that can improve retrieval quality. Therefore, for aligning video and mocap, we use flexible alignment methods (based on Dynamic Time Warping (DTW)) and show their effectiveness as compared to rigid matching.

Another challenge lies in generating a video and mocap representation that is comparable across the two modalities. Traditional methods based on silhouette are not practical for realistic videos. We also propose a new similarity measure relying
on the 2d pose and motion information (see Figure 4.1 for a summary).

Finally, we notice that there is no standard benchmark for video-based mocap retrieval. Therefore, we propose V3DR. V3DR allows us to evaluate the task quantitatively. The benchmark uses action labels as a way to assess correct alignment between video and mocap. As a part of the benchmark, we provide per-frame action labels for 4.5 hours of mocap data. We also provide a set of realistic video queries taken from YouTube.

In summary:

- We introduce the task of video-based mocap retrieval with a standard benchmark and a method for quantitative evaluation. The benchmark is made publicly available\footnote{http://www.cs.ubc.ca/labs/lci/v3dr/}.
- We propose a new motion and 2d pose based feature descriptor that make the two modalities comparable for retrieval. Our features are effective on realistic videos as demonstrated by our experiments.
- We also benchmark different alignment techniques such as Circulant Temporal Encoding (CTE) \cite{RDC13} and DTW-based methods. These experiments show that flexible matching helps improve the retrieval quality.

1.2.3 3d pose estimation in sports videos

In the final part of the thesis, we address the problem of 3d pose estimation in unconstrained videos. Given a video sequence, the task is to estimate 3d body articulations of all the subjects in each frame. The task has many applications in complex activity recognition, sports video analysis, scene reconstruction, computer games, and natural user interfaces. We focus on estimating 3d pose from professional basketball videos (see Figure 1.4).

Given the complexity of articulated human motion in 3d and the natural depth ambiguity in monocular videos, pose tracking algorithms often rely on human motion or pose models learned from mocap data \cite{ARS10}. A large body of previous research has focused on creating models that can generalize well with small amounts of data \cite{Fle11}. However, these models are often specific to an action.
Figure 1.4: 3d pose estimation from a professional basketball sequence taken from a broadcast video. This is a challenging problem due to depth ambiguity, artifacts such as motion blur, and the complexity of player movements. We show two frames of a sequence with the corresponding 3d pose obtained by our system.

Also, learning these action-specific models requires manually labeling mocap sequences. Therefore, we design a flexible method that can exploit a large set of unlabeled 3d motion examples. We do not need to know the action featured in the video in advance and our method scales well with respect to the size of the data. We use approximately 4.5 hours of CMU mocap dataset [cmu] for our method, and we do not require any labels associated with these mocap files.

Motion synthesis — a related problem often studied in computer graphics — aims to generate human motion sequences that satisfy some constraints, while relieving animators from the time-consuming task of manually editing joint locations. Specifically, the objective is to generate a smooth 3d pose sequence without specifying the joints per frame. This approach is especially useful in interactive animation settings where requirements for a particular motion cannot be anticipated in advance. In this work, we let a video sequence replace the animator, and exploit a motion synthesis algorithm to generate a 3d pose sequence that best explains the image evidence. A canonical motion synthesis approach is the Motion Graph (MG) [KGP02]. The MG models the space of 3d human motion as a graph, where each walk on the graph is a possible 3d pose sequence. MGs have been used
for 3d pose estimation using videos [AF02, RSH+05]. However, there are two main challenges in making this approach practical and scalable, which we address in this thesis.

The first problem lies in constructing the graph. We can build the graph offline with short mocap sequences as nodes and transitions that allow for smooth motion between the nodes as edges. However, this method does not scale well, and the graph becomes unwieldy as we add more mocap examples. We address this challenge by constructing the graph on-the-fly for each video sequence. The second challenge is using noisy evidence from the video to search the graph. Instead of using a general directed graph, we construct a trellis graph that can be searched efficiently. As a trade-off, we need a shortlist of mocap sequences suitable for the input video that we obtain using video-based mocap retrieval described in the last section. Since we also align the nodes in our trellis graph to the video sequence in time, we call our approach the Localized Motion Trellis (LMT).

In summary:

- We present a novel 3d pose estimation method based on motion synthesis, called the LMT. By using video-based mocap retrieval and employing a simpler graph structure, we make the LMT scalable (in data) and easier to search.

- Most 3d pose estimation methods have been demonstrated only in very constrained scenarios with a single activity. The LMT allows us to estimate pose without knowing or estimating an action label. Also, transitions between activities are naturally handled in this model. Therefore, we can demonstrate our method on a challenging sports video sequence.

1.3 Outline

The thesis is organized into 6 chapters. Chapter 2 describes the relevant related work. We discuss our novel cross-view action recognition approach in Chapter 3. We also introduce the Mocap Dense Trajectories (MDT) feature to learn from synthetic examples. Chapter 4 introduces and formalizes the new task of video-based mocap retrieval. We also present a challenging benchmark, V3DR, to test the quality of the retrieved mocap examples. Subsequently, we employ ideas from the
previous chapters to propose the LMT — a non-parametric approach to 3d pose estimation — in Chapter 5. We evaluate the effectiveness of the LMT on a professional basketball sequence. Finally, we conclude and discuss future work in Chapter 6.
Chapter 2

Background and Related Work

Recognizing human activity and estimating articulated pose from monocular videos are two crucial challenges in computer vision because of their wide-ranging applications. The task, in the case of action recognition [VNK15], is to classify a video into one of the predefined action classes, e.g., sports actions such as discus throw, pole vault or typical activities such as pick-up, sit-down, get-out-of-the-car [WKSL11]. Action recognition has numerous applications including video indexing and vision-based security systems. In this thesis, we target a very specific challenge of adding view-invariance to the action recognition task. Pose estimation and tracking deals with localizing body joints of a person in 2d or 3d space. A human pose sequence provides a detailed description of movements that is useful for natural user interfaces (for video games and computing) [SGF+13], human-robot interaction [KS16] as well recognizing complex activities [RF03]. We use video-based mocap retrieval as a subproblem for 3d pose estimation, but it is also useful for other applications. For computer games and character animation, mocap retrieval can be used to pick out the relevant sequence from a large collection of mocap files. Thus, we can save the cost of collecting the data again. Since there is an extensive literature on all of these problems, we limit ourselves to the most relevant methods. For further details see the review by Moeslund et al. [MHKS11].
2.1 Cross-view action recognition

Due to the highly articulated nature of the human body, the appearance of pose and motion changes considerably when seen from a different viewpoint (see Figure 1.2). Cross-view action recognition is defined as the task of recognizing action where training and test videos come from cameras with different points of view, called the source and the target view respectively.

As we mentioned in the Chapter 1, features commonly used for action recognition are not viewpoint invariant. Also, when classifying actions, the performance drops significantly when no supervision is available in the target view (see Figure 2 in [LZ12]). Therefore, to make action recognition robust to changes in viewpoint, we need a strategy to either devise a feature descriptor that is invariant to viewpoint changes or transfer knowledge across views by establishing a relationship between viewpoints and action descriptors.

2.1.1 Reasoning in 3d space

For action recognition, one of the obvious ways to achieve view-invariance is 3d modeling of the human motion. However, the challenge lies in solving some of the intermediate problems such as 2d or 3d pose estimation. Ramanan and Forsyth [RF03] track 2d body parts and match them to mocap data annotated with action labels. They also recover 3d pose and camera viewpoint in the process. Another set of methods build a voxel-based 3d representation of human motion using multi-view video data [WBR07, YKS08]. Although these methods do not require explicit 2d part detectors, they still need synchronized and calibrated multi-view videos to construct the model. Such data is challenging to acquire in an unstructured environment.

2.1.2 Statistical approaches

Many recent approaches to cross-view action recognition begin with the view-dependent feature description for activity. Rather than reasoning about geometry, these methods directly transform the view-dependent descriptor spaces to align them. Farhadi et al. [FT08] use the frame-level correspondence between descriptors from synchronized source and target views to learn the transformation between
features across views. The training requires synchronized multi-view videos. Liu et al. [LS11] build separate dictionaries for features in source and target views, and look for correspondence between words using bipartite matching to form bilingual words. These bilingual words act as a mapping from individual dictionaries to a common one. This approach is more flexible as it requires video-level correspondences as opposed to frame-level correspondences between different views, which relaxes the constraint of videos being precisely synchronized. Zheng et al. [ZJ13] simultaneously learn a common dictionary between views and individual view-specific dictionaries. They describe each video as a combination of these two factors. All the above methods use supervision in the form of correspondence or partial labels in the target view (see Figure 2.1(a-b)).

Inspired by unsupervised domain adaptation [GSSG12], some of the recent approaches [LZ12, ZWX+13] can work only given unlabeled examples in the target view. In Chapter 3 we further relax the assumption by using unlabeled mocap data instead. Our method does not require the matching step [LS11] or any target view examples (labeled or unlabeled).

2.2 Retrieval of human motion

Mocap data serves as a compact description of human motion. It is commonly used in video games, computer animation, and special effects. Efficient mocap retrieval can help animators find relevant clips to reuse in an animation. Various input types such as hand-drawing [CLAL12, CYI+12], movements of a wooden puppet [FGDJ08, NNSH11], Kinect depth map [KCT+13], and mocap [MRC05] itself have been used to query mocap datasets. Retrieving similar motion capture sequences given a short video as a query can also be useful 3D pose estimation (Chapter 5) and cross-view action recognition (Chapter 3). Although it is widely applicable, video remains a largely unexplored and challenging input modality for mocap retrieval.

2.2.1 Descriptors for pose and motion retrieval

The first challenge in video-based mocap retrieval is to establish similarity between a mocap and a video frame. The question arises — what is a good representation
Figure 2.1: Commonly used evaluation strategies for cross-view action recognition. Each circle denotes a video, and solid circles represent labeled examples. The data from two different views is arranged in Source and Target columns. The boxes represent different action types. 

(a) In correspondence mode, a fixed fraction of examples is known to be the same action seen from two views. The lines connecting the circles denote these correspondences.
(b) In semi-supervised mode, a small percentage of test (or target view) examples is annotated with a class label.
(c) The unsupervised mode is the most challenging case, where no annotations connect the source and the target examples. In this work, we demonstrate our results on the unsupervised modality only.

Efficient retrieval of similar 3d motion examples given a short mocap query (mocap-to-mocap matching) is a closely related problem. Müller et al. [MRC05] describe each mocap sequence using binary geometrical features based on relative 3d arrangements of body parts. Their features are designed to capture the similarity in the activity space rather than the exact numerical similarity in pose. Kovar and Gleicher [KG04] use a numerical similarity measure ($L_2$ distance between the aligned point cloud of 3d joints), but add query expansion to extend the search to logically similar motions. These techniques are effective at 3d-to-3d retrieval.

2d pose can be used for video retrieval. Given an image featuring a person as a query, the task is to find the frames with a similar pose from a video database. To describe pose in each video frame Eichner et al. [EMJZF12] first run a 2d pose de-
A set of statistics computed on these heatmaps acts as the descriptor for the pose to calculate similarity between the image and the video frame. Jammalamadaka et al. [JZJ15] propose an extension based on deep-poselets [BYF14]. Their method can also be extended to other query types such as Kinect depth maps [JZE+12].

Silhouettes are commonly used to match human pose in images to 3d examples. For an image, we can use background subtraction to obtain a silhouette. In case of mocap, silhouette can be easily obtained using a 3d shape model driven by the mocap sequence [SVD03, RSH+05]. For instance, Ren et al. [RSH+05] search for mocap examples given an image using Haar-like features based on silhouettes from multiple synchronized views. However, in the case of realistic images, it is difficult to generate a clean silhouette of the person. To deal with this limitation, we use an off-the-shelf 2d pose estimation [YR13] technique to establish a correspondence between a video and a mocap frame (Section 4.2). We also use MDT (introduced in Chapter 3) to describe human motion in a mocap sequence. The features based on these trajectories are comparable to DT features [WKSL11] in videos. We demonstrate that MDT are complementary to the pose-based features on this task.

2.2.2 Exemplar-based 3d pose estimation

Video-based mocap retrieval can also be used for 3d pose estimation (more details in Section 2.3). The example-based approaches for 3d pose estimation in videos often retrieve 3d pose for each frame from a database and then smooth the output over time [BMB+11, YKW14]. Another possible way to add temporal consistency and restrict search is using a Motion Graph (MG) [RSH+05]. We hypothesize that matching short videos instead of single frames can help add higher-order temporal constraints to matching. Also, among the methods mentioned above, [BMB+11] uses a depth image and [RSH+05] works with multi-camera input. Since we are only using monocular RGB videos, a sequence has much richer information than a single image to allow for a more robust match in our case. Therefore, we focus on matching short video sequences to mocap examples.
2.2.3 Comparing and aligning temporal sequences

To evaluate a match between a mocap and a video, we require alignment of these two temporal sequences. This alignment can be flexible (elastic matching) or inflexible in time. The rigid alignment works well for tasks such as copy detection (in videos) or event recognition [RDC+13]. However, due to the style and speed variations in human motion, flexibility in time is crucial to improving matching quality (Chapter 4). We also wish to align the video frames to their corresponding mocap frames based on pose and motion similarity, which requires temporal flexibility in the general case.

Dynamic Time Warping (DTW) is a popular algorithm used to align temporal sequences, and to cluster time-series data. It uses dynamic programming to find the minimum-cost alignment of two sequences subject to constraints suitable for time-series data, i.e., monotonicity and local continuity (described in Section 4.3.2). FastDTW provides an efficient approximation to DTW by solving the problem iteratively at multiple scales, achieving a complexity of $O(n)$ [SC04] in the length of the sequence, as opposed to $O(n^2)$ for the original DTW. One of the limiting assumptions in DTW and its variants is that the first and the last frames of the two sequences must be aligned (the end-point constraint). The constraint is not suitable for comparing sequences of different length. Some recent methods have attempted to relax the end-point constraint partially by fixing one of the ends but letting the other float [SVBC08, SVBC09]. DTW-S [YCN+11] matches sequences of different sizes and allows flexibility at both ends by assuming balanced alignment, i.e., the warping uses an equal number of frames from both sequences. This assumption is violated when we match actions performed at different speeds. Subsequence DTW (SS-DTW) [Mül07] is an efficient method for relaxing the end-point constraint; however, it introduces a bias for choosing a shorter database subsequence for a given query (more details in Section 4.3.2). Normalizing the score with a measure of path length [AF13, MGB09] can remove the bias. We experiment with different versions of these normalizations in Chapter 4.

An alternate approach to sequence alignment is to treat the warping as a discrete version of a monotonic function. For instance, GCTW [ZdIT15] poses alignment as an optimization over a continuous space. GCTW can also incorporate
floating end-points; however, due to a non-convex objective function, GCTW requires effective initialization to avoid local minima.

2.3 3d human pose estimation in videos

The final part of this thesis deals with estimating articulated 3d pose of the players in broadcast team sports videos, where only one view is available at a time. Estimating pose in a monocular video is particularly challenging due to foreshortening and occlusion that make the solution inherently ambiguous.

2.3.1 Statistical priors to human motion

To resolve the ambiguity, a prior model of human pose or motion can be used to help the problem. Under the a Bayesian formulation, we can write the problem of tracking 3d pose in a video as

$$\arg\max_{e} p(e|D) = \arg\max_{e} \prod_{i} p(I_i|e_i) \times p(e_1, e_2, ..., e_N)$$

(2.1)

where $e = (e_1, e_2, ..., e_N)$ is the pose at all the frames in the range $[1, N]$. $D$ is the set of observations $(I_1, I_2, ..., I_N)$. The first term $p(I_i|e_i)$ is the likelihood of an observation $I_i$ given pose $e_i$, and the second term is the prior probability of a pose sequence $p(e)$. Under the first order Markov assumption the expression can be simplified as

$$p(e_1, e_2, ..., e_N) = p(e_1) \prod_{i=2}^{N} p(e_i|e_{i-1})$$

(2.2)

where $p(e_i|e_{i-1})$ can be modelled as a normal distribution to encourage a smooth output sequence [SHG+11].

In general, it is challenging to build an empirical probability distribution $p(e)$ because of the high-dimensionality of the pose data. Therefore, the prior term is often modeled as a latent low dimensional space of the activity-specific human poses [Fle11]; particularly, non-linear regression with Gaussian process has proven successful using small amounts of training data [LM07, UFHF05, WFH08]. Recently, bootstrapping 3d pose with 2d pose detectors has become a popular approach [ARS10, EAJ+15]. Andriluka et al. [ARS10] associate 2d body parts de-
tections over time, using tracking-by-detection, and later used an hGPLVM \cite{LM07} (hierarchical Gaussian process latent variable model) to generate stable 3d output. Similarly, Simo-Serra et al. \cite{SSQTMN13} applied kinematic constraints to obtain 3d pose from noisy 2d estimates. The main drawback of these statistical priors is that they are specific to an action and hence are not suitable for complex activities involving multiple actions.

In contrast, our approach to 3d pose estimation (Chapter 5) is designed to benefit from large collections of unlabeled mocap data. Although action recognition can be used as a means to pick the appropriate 3d motion model \cite{YGG12,YKC13}, the common assumptions are: (a) a set of actions is known \textit{a priori} and (b) there is a pre-trained model for each action.

### 2.3.2 Motion synthesis

In addition to generating the pose, we are also interested in reconstructing the realistic human motion. Thus, we can alternatively view this problem as 3d motion synthesis driven by the video input. In many graphics applications, motion synthesis aims to generate a sequence of 3d poses that satisfy user-specified constraints while relieving animators from the time-consuming task of manually editing individual joint locations. The objective is to generate the sequences by specifying only high-level goals (i.e., get the character from a location $a$ to $b$). This method is especially useful in interactive animations, where the requirements for a particular motion cannot be fully anticipated in advance.

Approaches to motion synthesis can be divided into three main categories — physics-based, statistical and example-based. Physics-based methods simulate the dynamics of the body and the physical world with the goal of making virtual characters learn control strategies to perform a variety of tasks \cite{GvdPvdS13}. These methods have been previously used for 3d pose estimation in 2d videos by adding visual evidence in the control loop \cite{VSHJ12,WC10,BF08}. However, they suffer from a high computational cost and are not able to produce motion that looks natural to the human eye. Statistical methods search for a low-dimensional representation of human motion to build a generative model. These methods can be used to produce character animations, given user-defined constraints \cite{LWS02}, and
have also been used for 3D pose estimation in monocular video [ARS10, LTSY09]. While these models can be learned from a small number of 3D examples, they tend to be action-specific, and are unable to capture complex variations in motion. Example-based methods are widely popular in interactive graphics [PPI10] due to their simplicity and the ability to generate natural motion. In these methods, motion examples are spliced, interpolated and concatenated to synthesize new character animations. Our approach to 3D pose synthesis is inspired by Motion Graphs [KGP02] — a popular approach to 3D exemplar-based synthesis.

Motion Graphs
Motion Graphs (MGs) exploit large collections of mocap data by discovering good transitions between different sequences. For any pair of motion sequences, a similarity matrix is computed, where high scores correspond to points of smooth transitions. A graph is then built, such that nodes represent mocap frames and edges denote suitable transitions. Motion can be generated by simply walking the MG, interpolating the joint angles to generate natural-looking 3D sequences.

Generating motion using a MG is very simple: one only has to walk the graph. However, a random walk of the graph is likely to be of little use. Rather, the challenge is to find walks in the graph that satisfy some constraints. The expressiveness of an MG is determined by the number and variety of sequences used in its construction: a large number of examples can create a huge MG that is able to generate a variety of outputs; however, searching for a path that satisfies desired constraints in a large MG quickly becomes infeasible.

2.3.3 Direct regression to 3D pose
In contrast to tracking assisted with learned priors or motion synthesis, regression based techniques such as [AT06, UD08] learn a regression function from 2D observations to 3D pose. Recently, Tekin et al. [TSW+15] use HOG3d features to regress directly from image sequences to 3D pose using kernel ridge regression. The motion as well as the pose is encoded in the space-time volume and using data from multiple action categories helps the accuracy. This observation suggests that the regression is not activity-specific. However, regression can only be applied to
short video sequences, as the dimensionality of regression input and output grows linearly with the number of frames. Another limitation of such methods is the requirement of synchronized videos with the 3d pose for training. Such data is expensive to acquire in realistic scenarios such as sports, and outdoor environments.

2.4 Summary

We have reviewed the most relevant approaches to human-motion analysis and synthesis. For cross-view action recognition, the present state-of-the-art methods are based on domain adaptation (i.e., learning a transformation for action descriptors from one-view to the next). Thus, these methods do not depend on the underlying feature representation. However, the requirement of multi-view video examples in realistic settings is challenging. Therefore, we look for a completely unsupervised technique, requiring no-synchronized views or specialized data collection. Similarly, in the case of pose estimation, the most successful techniques using monocular videos (tracking-by-detection approaches) learn their pose or motion prior with carefully curated data while the direct regression based methods need synchronized video and mocap. Since we are interested in avoiding any labeling effort and using unstructured examples, our approach to 3d pose estimation is based on retrieval of mocap subsequence from a large dataset using a short video as query and effectively combining the output using exemplar-based motion synthesis.
Chapter 3

Unsupervised Cross-View Action Recognition

We present a novel approach to recognize human actions in videos from a new viewpoint — a camera angle not seen in training examples. To build a view-invariant model of actions, we establish a link between the video descriptor and the point of view of the camera. Many successful approaches to the problem use labeled examples to link descriptors across views [FT08, ZJ13, LS11]. Instead of relying on labeled examples, we learn the mapping between views via synthetic feature generation using a large corpus of unlabeled motion capture (mocap) sequences.

We also present a method to generate video-like motion features from mocap without any photo-realistic rendering. These synthetic features are analogous to Dense Trajectories (DT) [WKSL11] features in videos, often used for action recognition. We refer to our synthesized version of DT as Mocap Dense Trajectories (MDT). Once we generate synthetic features for a variety of mocap sequences, a linear function can be learned to map these features from one view to the next. Because of the similarity between the synthesized and the real trajectories (see Figure 3.2(a)), the mapping learned on synthetic features can be applied to the real training data to generate additional multi-view training examples. These synthetic descriptors along with the original training data are then used to train the action classifier. As shown by our experiments, this simple scheme of generating syn-
thetic training examples leads to significant improvements in the cross-view action recognition accuracy. Since we do not require any multi-view labels, we refer to the approach as unsupervised.

3.1 Overview

A view-invariant representation of human motion is crucial for effective action recognition. However, as noted in Chapter 1, widely popular shape and optical flow-based features [DT05, DTS06, GBS+07, YS05], which are used to describe actions in videos, are not specifically designed to be viewpoint invariant. Consequently, the effectiveness of a method depends on the availability of training data from diverse views. However, this may not be the case for many data sources, e.g., internet videos often have a very limited set of viewpoints (see Figure 4.5 for a few samples). If our test videos do not follow the same limited distribution of viewpoints, we need to come up with strategies to transfer knowledge across views.

Our approach uses mocap examples for knowledge transfer. To utilize mocap sequences, we define MDT as the orthographic projection of 3d trajectories. We obtain 3d trajectories by following points on a human model driven by a mocap sequence (details in the next Section). In this chapter, first, we describe the process to generate MDT (Section 3.2). Second, we use corresponding trajectories synthesized from multiple views to learn the transformation of motion features due to change in viewpoint (Section 3.3). We further utilize these transformation functions to add view-invariance to action recognition. Finally, we present evaluations on the INRIA XMAS (IXMAS) [WRB06] dataset, a standard benchmark for the problem (Section 3.5).

3.2 Dense Trajectories from Mocap (MDT)

Our goal is to generate mocap features equivalent to the DT [WKSL11] features. However, mocap sequences only provide a series of 3d joints locations over time. To generate a surface representing the body, we approximate body parts by tapered cylinders with bones as axes, and put a dense grid of points on the surface of each cylinder (see Figure 3.1 (b)). We uniformly sample points to get approximately 1500 points for the whole body model.
Figure 3.1: Mocap dense trajectory (MDT) generation pipeline. (a) Mocap sequences have 3d body joint locations over time. (b) We approximate human body shape using tapered cylinders to obtain a “tin-man” model. (c) Sampled points on the surface of cylinders are projected under orthography for a fixed number of views and (d) cleaned up using hidden point removal. (e) Connecting these points over a fixed time horizon gives us the synthetic version of dense trajectory (DT) features [WKSL11].

3.2.1 Generating multiple projections

We project the points of the model surface under orthography for a fixed number of views. For observing human motion at a distance, orthographic projection offers a reasonable approximation to perspective projection. With orthographic projection, there are only two parameters to vary — the azimuthal angle and the elevation angle. We choose the azimuthal angle $\phi \in \Phi = \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3\}$, and the elevation angle $\theta \in \Theta = \{\pi/6, \pi/3, \pi/2\}$ measured from the vertical pointing upwards. By discretizing the angle space, we get 18 different projections per frame. Although the equal division along azimuthal and elevation angles does not uniformly sample the viewing sphere, it is a simple and adequate mechanism for choosing camera viewpoints. Since we assume that a camera looking up is unlikely, we do not include elevation angles greater than $\pi/2$. We also assume that there is no camera roll (see Figure 3.1 (c)).
3.2.2 Hidden point removal

MDT account for self-occlusions by removing points that should not be visible from a given viewpoint. We use a freely available off-the-shelf implementation of the method by Katz et al. [KTB07]. Hidden point removal gives us a set of filtered points for each projection (see Figure 3.1 (d)).

3.2.3 Trajectory generation and postprocessing

For a given viewpoint, we connect the filtered 2d points over a fixed time horizon $\tau$ to obtain synthetic trajectories. Therefore, only points that are visible within the $\tau$ frame window are included in trajectories. To make synthetic trajectories comparable to video dense trajectories, we make sure that the frame rate for mocap is the same as for the videos used in the experiments. We use $\tau = 15$ (covering half a second in a 30 fps sequence), which has been found to work well in the past [WKSL11]. Again following Wang et al. [WKSL11], we remove trajectories smaller than a threshold as described in their paper. Figure 3.1 provides an overview of the MDT generation pipeline.

A trajectory descriptor can be generated for a physical trajectory by concatenating the velocities at each frame and normalizing it by the total length of the trajectory [WKSL11]. Thus, each $\tau$ frame long trajectory can be described using a $2\tau$ dimensional vector.

3.3 Learning from synthetic data

Mocap data allows us to observe how the appearance of the same 3d trajectory, generated using human movements, transforms from one view to the next by looking at the descriptors from two corresponding trajectories. We begin by generating mocap trajectories as seen from a pair of viewpoints (as shown in Figure 3.2 (b)) for a large number of mocap sequences. The mocap sequences are taken from the CMU-mocap dataset [cmu]. We use a visual vocabulary for trajectory descriptors and quantize descriptors to their closest codewords. This hard vector encoding simplifies learning the dependency between the codewords in the two views.

---

model the transformation of features due to viewpoint change as a linear function of codewords. We make the following two assumptions while learning the transformation function:

- We assume that the feature transformation is independent of the activity, and generate only one function per view change. Although this assumption simplifies the learning process, it may not be a good approximation in all the scenarios. For instance, an individual trajectory may have a very similar shape for a walk and a turn sequence from one view, but it may look very different from another viewpoint.

- We also assume that each trajectory transforms independently, i.e., we do not model the effect of transformation of a trajectory on the neighboring trajectories. The independence assumption lets us transform each codeword separately. Although it works well in practice (as shown by our results), the assumption clearly does not hold true for human motion.

3.3.1 Generating correspondences

We assign a unique ID to each point on the 3d surface of the human model (described in Section 3.2). Then, given two viewpoints, we can get a feature pair that originates from the same point on the surface (see Figure 3.2(b)). When the exact match is not found due to self-occlusion, a small neighborhood on the model surface is considered equivalent, i.e., points in the neighboring region are assumed to have the same ID.

3.3.2 Learning codeword transformations

For each view pair, we refer to the initial viewpoint as the source view, and the changed viewpoint as the target view. We quantize the MDT features using a fixed codebook $\mathcal{C}$ of size $n = 2000$. Given a source elevation angle $\theta$ and a relative change in viewpoint given by $\Delta = (\delta \theta, \delta \phi)$, we define the training set $D_\Delta = \{(f_i, g_i)\}_{i=1}^m$ to be the set of $m$ pairs $(f, g) \in \mathcal{C} \times \mathcal{C}$, where $f_i$ and $g_i$ are the codewords for two corresponding MDT features.
Figure 3.2: (a) A visual comparison between synthesized MDT (left) and optical flow based DTs [WKSL11] (right). We exploit the similarity in their shape to learn a feature mapping between different views using MDT, and use this mapping to transform the DT based features commonly used for action recognition. (b) To generate view correspondence at the local feature level, the human body is represented using cylindrical primitives driven by mocap data. We project the 3D path of the points on the surface to multiple views and learn how the features based on these idealized trajectories transform under viewpoint changes.

Given the training data $\mathcal{D}_\theta^\Delta$, the relationship between codewords $f_i$ and $g_i$ can be modeled within a probabilistic framework. Since we assume each feature transforms independently, we can learn a joint probability mass function $p(F,G)$ which captures the probability of the codeword pairs $(f_i, g_i)$. We train the model using maximum likelihood estimation and calculate the empirical probability by counting the co-occurrences of $(f_i, g_i)$ in $\mathcal{D}_\theta^\Delta$ followed by normalization. The conditional probability distribution of $G$, given an observation of codeword $f_i$ in the source domain can be written as

$$p(G|F = f_i) = \frac{p(F = f_i, G)}{p(F = f_i)} = \frac{p(F = f_i, G)}{\sum_{c \in C} p(F = f_i, G = c)} \quad (3.1)$$

After observing an instance of codeword $f_i$ in the source view, $p(G|F = f_i)$ allows us to infer the possible outcomes in the target view. In the next section, we
use this probability distribution to map source Bag of Words (BoW) descriptors to the target view.

3.4 Synthesizing cross-view action descriptors

Given a training descriptor, we use the mapping between codewords to “hallucinate” action descriptors as seen from different viewpoint changes from the initial viewpoint and use them as additional examples for training, thus adding view-invariance to our model. We assume that, due to the similarity between mocap and dense trajectories, we can learn the view transformations on one and apply it to the other. Figure 3.2 (a) shows the visual similarity between trajectories generated for a video and synchronized mocap trajectories.

Given a BoW descriptor of an action, we wish to synthesize a corresponding new descriptor as seen from the viewpoint \( \Delta = (\delta \theta, \delta \phi) \) away from the original view. Let \( x = [x_1, \ldots, x_n]^T \) be the BoW descriptor in the source view, and \( y = [y_1, \ldots, y_n]^T \) be the descriptor we want to estimate. As seen in the last section, we have a probabilistic mapping between codewords across views. Using this mapping, we return an average descriptor by taking the expectation.

\[
\bar{y} = [E[y_1], \ldots, E[y_n]]^T 
\]  
(3.2)

\[
E[y_j] = \sum_{i=1}^{n} x_i \cdot p(G = f_j | F = f_i) 
\]  
(3.3)

By organizing \( p(G|F) \) in the form of a matrix (say \( N \)) where the \( i \)-th row is the categorical distribution \( p(G|F = f_i) \), we can rewrite the above formulation as a linear transformation

\[
\bar{y} = N^T x 
\]  
(3.4)

where \( N \) is the transition matrix corresponding to a transition \( \{ \theta, \Delta \} \).

We generate these additional multi-view examples for each training sequence and append them to our training data for cross-view action recognition.
Figure 3.3: All camera views from the IXMAS dataset. We show the kicking action synchronously captured from the 5 camera angles. Note that the appearance changes significantly across views. Camera 4 is especially challenging because of its extreme elevation angle.

3.5 Experiments

The goal of our experiments is to evaluate the effectiveness of the synthetic multi-view descriptors on unsupervised cross-view action recognition. We hypothesize that including these descriptors in the training data will lead to view-invariance in action classification.

3.5.1 Dataset and evaluation

We choose the INRIA IXMAS dataset [WRB06] for our experiments. The dataset has 11 actions categories, check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, and pick up, performed 3 times by 10 subjects. The dataset contains videos of subjects performing these activities — synchronously captured using 5 cameras (Figure 3.3).

Previously, the IXMAS dataset has been used for cross-view action recognition in different evaluation modes [ZWX13] (described in Section 2.1). Our approach is well-suited for the unsupervised mode, where no labeled examples in the test view are available. The unsupervised mode tests the scenario in which the entire training data comes from an unknown viewpoint (called the source view), and the test view (or the target view) is not known in advance. This mode is the most challenging in terms of classification accuracy reported in the literature so far, since the target view is novel. To evaluate action recognition, we pick videos from one camera view as the training set and use another camera view as the test set. Five cameras give us $2 \times C_2^5 = 20$ train-test pairs. We report classification accuracy for
each camera pair separately in Figure 3.4.

As mentioned earlier, IXMAS data is synchronously captured. To avoid contaminating our training data with test examples, we use a leave-one-subject-out strategy for evaluation, i.e., we exclude the videos featuring the test subject from the training data. This strategy prevents including the test example, seen from the training view, in the training set. However, a more prevalent mode of evaluation is where all the examples from a source view are used for training. We use this mode to compare with the state of the art (Table 3.1).

3.5.2 Dense trajectories for action classification

We describe each video in our dataset as a BoW of \( DT \). We compute \( DT \) using the code provided by Wang et al.\(^2\), sampling every other pixel, and let each trajectory be 15 frames long as in [WKSL11]. We cluster the trajectories from the training view into 2000 k-means clusters to obtain a codebook \( C \), and generate a BoW descriptor for each video. For classification, we train a non-linear SVM with a \( \chi^2 \) kernel. The classification only based on the BoW descriptors from the source view videos without any knowledge transfer across views gives us the no augmentation baseline.

3.5.3 Mocap feature synthesis

For learning feature correspondences using the mocap data, we use the CMU Motion Capture Database [cmu]. This dataset includes over 2600 mocap sequences of human subjects performing a variety of actions. Though the CMU dataset contains some action labels for each file, we do not use these annotations.

To learn the mapping between codewords, we take a random subsample of the mocap data, keeping 10% of the frames. We generate \( MDT \) from multiple viewpoints and quantize them using the same codebook \( C \). For this experiment, we quantized the elevation angle \( \theta \) to \( \{ \pi/6, \pi/3, \pi/2 \} \) degrees and the azimuthal angle \( \phi \) to 6 equally spaced angles in \( [0, 2\pi) \). Thus, for each source elevation \( \theta \), we have 17 possible viewpoint transitions (excluding transition to itself). Given a training example from IXMAS we generate one synthesized descriptor per transi-

\(^2\)https://lear.inrialpes.fr/people/wang/dense_trajectories
tion as described in Section 3.4. Since we do not know the source elevation $\theta$ for the training set, we consider transitions from all the possible elevation angles. This gives us 51 synthesized descriptors per training example.

Following [CG13], we augment our original training data using these new descriptors. In such augmentation schemes, the synthesized data is often weighted less compared to the original data. We empirically set the weight of the augmented data to 0.01, while the original examples have equal weight 1. In our case, this importance is controlled by the slack penalty of the SVM. This way, we account for a) the imbalance in the number of examples in original and augmented data, and b) the fact that the augmented data might contain errors. Again, we train a non-linear SVM with a $\chi^2$ kernel on the real and augmented data.

### 3.5.4 Mocap retrieval-based augmentation

We add another baseline using video-based mocap retrieval (Chapter 4). Instead of generating synthetic descriptors for each view change, we directly search for the best-matching mocap sequence in a mocap database. We generate descriptors corresponding to different projections (18 different projections described above) of the retrieved mocap example, and add them to our training set. This is an alternative method for adding view-invariance to our model.

For retrieval we use the shortest 2000 sequences from the CMU mocap. We also use a concatenation of pose and trajectory features (P+T) to describe each video and mocap frame, along with SLNDTW (+smooth) for matching the sequences. Each of these steps is described in detail in Section 4.2 and Section 4.3. Our retrieval method also returns the aligned mocap frames for each video query. We only use MDT corresponding to these aligned frames for data augmentation. The synthetic examples are weighted the same as above for SVM training.

### 3.5.5 Results

Figure 3.4 shows the classification accuracy per camera pair. We use the leave-one-subject-out scheme for evaluation. Our mocap feature synthesis approach performs the best on most of the camera pairs. We also note that our method consistently outperforms the baselines with camera 4 as the target view. Camera 4 is an espe-
Figure 3.4: Accuracy for each camera pair. We highlight the best results in boldface and underline the second best value. Our mocap feature synthesis approach performs the best on most train-test view pairs.

Figure 3.5: Per-class classification accuracy of our method on the IXMAS benchmark. These are the same results as Figure 3.4 rearranged to show the per-class accuracy averaged over all train-test camera pairs. The mocap feature synthesis improves accuracy over the no augmentation baseline on every category except kick.
Figure 3.6: Confusion matrices before and after applying our mocap feature synthesis and retrieval-based augmentation approaches. We show average accuracy over all cameras as train-test pairs. We note that our data augmentation helps resolve confusion for many action categories.

Especially difficult test set because its elevation is significantly different than the other camera views (see Figure 3.3). We show per-class classification accuracy in Figure 3.5. Again, mocap feature synthesis gives the best classification accuracy on most classes. Also, it outperforms the no augmentation baseline on all categories except kick. We also compare the confusion matrices for no augmentation and feature synthesis approaches in Figure 3.6.

Finally, we test the sensitivity of our method to the number of viewpoints.
Figure 3.7: Variation of the average classification accuracy (on IXMAS) with the number of divisions of azimuthal and elevation angles for our mocap feature synthesis approach. We note that a small number of divisions in angles does not provide the coverage needed for all the test viewpoints. However, the gain in accuracy saturates as we keep increasing the number of synthesized views.

As described in Section 3.5.3, we quantize the viewpoint space into 3 divisions in elevation, and 6 along azimuthal angle. We vary the number of divisions and observe the average accuracy (in Figure 3.7). We observe that for the IXMAS benchmark, 2 divisions in elevation and 4 along the azimuthal are enough to cover the viewpoints. Note that the rest of the parameters are kept the same for this experiment including the weights of synthetic examples for learning the classifier. Since the number of augmented descriptors changes with the chosen number of divisions, we also experimented with fixing the weight for 3 divisions in elevation, and 6 in azimuthal, while adjusting other weights inversely proportional to the total number of views. However, this gave a very similar trend to Figure 3.7.

Comparison with the state of the art

We also compare with the previous work of Li et al. [LCS12] and the recent work by Rahmani and Mian [RM15] (Table 3.1). We use all the examples from the
Table 3.1: Comparison of the overall performance of our approach and the state of the art on the IXMAS dataset. We show the accuracy averaged over all the camera pairs. We highlight the best value with boldface and underline the second best value. The results for the leave-one-subject-out evaluation mode, as well as the usual mode (common in the literature) are shown. Rahmani and Mian [RM15] build on the initial version of our method [GSLW14], but learn a non-linear transformation between mocap trajectory features from different viewpoints to a canonical view. source view (instead of the leave-one-subject-out scheme) for a fair comparison with these methods. The compared results are taken directly from the cited papers.

We note that our no augmentation baseline outperforms [LCS12]. Also, our mocap feature synthesis approach is competitive with the state of the art [RM15].

### 3.6 Discussion

We have demonstrated a novel method for using unlabeled motion capture sequences as prior knowledge of human activities and their view dependent descriptions. To this end, we have introduced the MDT, a synthetic, idealized, and viewpoint-aware motion feature, generated using motion capture data. The MDT can also be seen as a bridge between 2d motion in videos and human movements in mocap examples.

We have used MDT to add view-invariance to action recognition without using multi-view video data or any additional data labeling effort. Since mocap data lives in 3d, we were able to use the MDT generated from different views of the same mocap sequence to learn the transformation of features from one view to the next. The learned transformation was then used to synthesise new training examples. Finally, we have also shown that synthetic feature generation is an effective technique for
unsupervised cross view action recognition.

### 3.6.1 Limitations and the future work

Although our method is competitive with the state of the art, it has multiple assumptions in the formulation that can be relaxed for further improvements.

As mentioned in Section 3.3, we assume that transformation of features with the change in viewpoint is independent of the activity. Though the simplification allows us to learn one transformation matrix per view change, it may not be a good approximation in some cases. Intuitively, the feature transformation depends on the activity of the person. This assumption may be one of the reasons for our method’s poor performance on the action category *kick*. Similarly, features from the different part of the body may not follow the same geometrical transformation. One of the ways we can deal with these limitations is to have a bank of transformations per change in viewpoint. To generate synthetic examples, we should be able to choose one transformation on the fly based on the overall action descriptor.

We note that the improvement due to mocap feature synthesis is not consistent across all action categories (Figure 3.5), as well as its performance is not symmetric across view pairs (e.g., 1-0 vs. 0-1 in Figure 3.4). This discrepancy may be due to occlusion of the critical body parts (involved in the action) from a particular camera angle and relatively large overall occlusion of the body at extreme camera angles. A more rigorous understanding of the dependency of our results on action categories and viewpoints requires further exploration.

Additionally, we decided to use MDT because they can be generated efficiently from mocap. However, the underlying model for generating motion features in MDT is very simplistic. A more realistic human model and rendering may give better features for learning view-invariance.
Chapter 4

Video-Based Mocap Retrieval and Alignment

In this chapter, we present a technique to retrieve motion capture (mocap) files efficiently, using a short video query. Our retrieval approach generates a list of aligned mocap snippets, ranked by their similarity to the video. We define the similarity based on the human motion depicted in these sequences.

The above task requires us to establish a frame-level similarity metric between video and mocap. To this end, we explore a set of features that are comparable across these two modalities. Given the similarity measure, we can use different temporal alignment and retrieval techniques. The first method we explore is based on cross-correlation that can be computed efficiently in the Fourier domain, thus, it is well-suited for our application. However, matching similar but stylistically different actions in mocap and video may require temporal flexibility to align them. Therefore, we also experiment with different flexible alignment methods.

We thoroughly evaluate the effect of the two stages — feature extraction and alignment — on the retrieval accuracy. Since there is no publicly available dataset for such an evaluation, we propose a new benchmark: Video-based 3d motion Retrieval or V3DR. Our benchmark consists of realistic video queries as well as frame-level annotations for a large mocap database to measure the performance of video-based mocap retrieval and alignment.
4.1 Overview

Retrieving similar mocap sequences from a large database can be useful for character animation, 3d pose tracking [RSH+05], and cross-view action recognition [GMLW14]. In this thesis, we use retrieval for 3d human pose estimation in complex activities such as team sports, with only monocular video as input. We discuss this application in Chapter 5.

To connect human motion in video and mocap, we look for a common representation between the two. Many previous methods have used features based on silhouettes or edge-maps [RSH+05]. A silhouette can be easily extracted from a video with a static camera and a known background, while generating the same from a mocap sequence requires creating a mesh for the body, rigging it to the skeleton, and rendering each frame using a virtual camera. Although these simple features are useful when dealing with videos in laboratory settings, they are hard to obtain reliably for realistic videos with complex, unknown backgrounds. To address this problem, the likelihood of a 3d pose given an image can also be obtained using discriminatively-trained 2d pose detectors (as shown in [SSQTMN13, ARS10]). Since these detectors (e.g., Flexible Mixture-of-Parts (FMP) [YR13]) are trained on realistic images, they tend to be more robust. We too partly base our similarity measure on the 2d pose estimate for each frame, as described in the next section.

The next stage of our retrieval and alignment algorithm involves temporal matching between the two sequences. The efficiency of the matching algorithm is crucial for scaling it to large datasets. Therefore, all the methods described in this chapter are linear-time (in the length of the sequence) algorithms. The final consideration is the viewpoint. We match the same mocap sequence as seen from multiple points of view. Thus, viewpoint estimation becomes a part of the retrieval process. We describe each of these ideas in detail in Section 4.3.

4.2 Video and mocap representation

A video featuring a person and a mocap sequence are complementary representations for human motion analysis. While videos have rich appearance information with a large variation in shape, clothing, and background, mocap sequences have
Figure 4.1: Representing image and mocap sequences for retrieval. The mocap data and videos featuring human motion have related but complementary information. We bridge the gap between the two modalities by estimating the 2d joint locations from each video frame, and projecting 3d mocap joints for a viewpoint (right). We also generate optical flow based DT [WKSL11] from video and synthetic trajectories from mocap (see Section 3.2) to describe motion patterns in both the sequences (left). Hence, we have a pose as well as a motion-based representation for both the modalities that are comparable to each other.

accurate 3d pose. Hence, it is a challenge to compare them. We need features that can be computed efficiently for both video and mocap. Additionally, they should be able to discriminate among various pose and motion patterns arising from human activities. In this work, we rely on features that have been shown to be effective for recognizing actions in videos (see Figure 4.1).

4.2.1 Trajectory-based motion feature

In the last chapter, we used Mocap Dense Trajectories (MDT) features as an idealized version of Dense Trajectories (DT) [WKSL11] features for cross-view action recognition. Here we use them to compare motion patterns between mocap and videos for retrieval. We synthesize trajectories for each file in the mocap database, as seen from 18 different viewpoints — elevation angle $\theta = \{\pi/4, 3\pi/8, \pi/2\}$ and azimuthal angle $\phi = \{0, \pi/3, 2\pi/3, \pi, 4\pi/3, 5\pi/3\}$. Each frame (per view) contains a different number of trajectories ($\tau$ frames long). Each trajectory can be described by a $2\tau$ long feature vector by only keeping horizontal and vertical
Figure 4.2: Corresponding joints between the 2d pose estimate in an image and the CMU-mocap skeleton. We only use a subset of mocap joints. This mapping is similar to the one used by Zhou and De la Torre [ZDIT14] for matching DT to a motion model learned from mocap.

Displacements over consecutive frames, normalized by the total length of the trajectory (as described in [WKSL11]). For each frame, we aggregate trajectories terminating at the frame using Fisher Vector (FV) encoding [PSM10], because it has been shown to better describe human actions as compared to Bag of Words (BoW) [OVS13]. Finally, following good practice [PSM10], we take signed square-root and $L_2$ normalize the Fisher Vector to obtain one motion descriptor per mocap frame (in this case containing the information from the last $\tau$ frames). We similarly obtain a corresponding descriptor for each video frame using DT features. Finally, we run PCA to reduce the dimensionality of both video and mocap frame descriptors to 128 dimensions.

One of the limitations of aggregating features is that we lose the information about the relative location of each feature. Also, trajectories do not have any semantic labels associated with the body parts. Therefore, we add further details using pose-based features.
4.2.2 Relational pose feature

We also use 2d relational pose features \[\text{MRC05}\] to describe each video and mocap frame. Relational features capture relative distances and orientations between all the body joints. These features are shift-invariant and robust to noise in the pose estimation \[\text{MRC05, YGG12}\]. Following \[\text{JGZ+13}\], we extract the feature for a frame, and compute all pairwise distances and all pairwise orientations between joints giving rise to a \(C_2^{15} = 105\) dimensional vector each. All inner angles for all combinations of 3 joint angles are also concatenated \((3 \times C_3^{15} = 1365)\) to obtain a 1575 dimensional vector. As suggested by Jhuang et al. \[\text{JGZ+13}\], we add motion information by appending the temporal differences of these features over the preceding few frames. We whiten and \(L_2\) normalize this feature vector along each dimension. Finally, we run PCA on each feature type separately and keep the same number of dimensions for each type to obtain a 128-dimensional vector.

To estimate 2d pose in each video frame, we use the Flexible Mixture-of-Parts (FMP) \[\text{YR13}\]. For mocap sequences, a comparable representation can be obtained by projecting the corresponding 3d joint locations along multiple viewpoints. We use an orthographic projection and the same viewing angles as described above in Section 4.2.1. Since mocap usually contains more joints than most 2d pose estimates, we pick a subset of joints from mocap. Figure 4.2 shows the correspondence between the 2d pose in an image and the mocap joints. Note that, in contrast to the 3d model, the 2d pose estimate does not distinguish between the right and the left body parts. However, the projection of the 3d joints can be made consistent by switching left and right labels of body parts when the 3d model faces away from the camera. We do this by checking the relative positions of the projected left and right shoulder joints.

We have made the code to generate relational pose features from the mocap sequences publicly available\(^1\).

4.3 Retrieval and alignment

Given frame level descriptors for video and mocap, our next task is to search a mocap database using a video as query. For retrieval, we rank all the files based

\(^1\)https://github.com/UBC-CVLab/rel-pose-feats
on their similarity with the video. In addition, for each mocap file we also find the best matching portion of the mocap to the query video. We refer to this problem as the alignment task. We explain how to measure the performance on these tasks in Section 4.4.1. First, we describe a few existing retrieval techniques and discuss their applicability to our problem.

**Notation and the measure of frame similarity**

We concatenate the pose and motion features (described in the last section) for each frame into a single vector of $d$ dimensions. Let $v$ be an $n$-frame video descriptor. We can obtain its matrix representation as $v \in \mathbb{R}^{d \times n} = [v_1^\top, \ldots, v_d^\top]^\top$ (in column notation) $= [v_1, \ldots, v_n]$ (in row notation). Similarly, we construct a database of the mocap descriptors using all the mocap sequences available in the dataset. Let $z_i \in \mathbb{R}^d$ be a mocap frame descriptor. We concatenate these for an $m$ frame long mocap file as $z = [z_1^\top, \ldots, z_d^\top]^\top = [z_1, \ldots, z_m] \in \mathbb{R}^{d \times m}$.

We use the dot-product, i.e., $v_i \cdot z_j$ as our measure of similarity between a mocap and a video frame. In case of DTW-based methods (described in Section 4.3.2), we need a distance measure instead of similarity. We use $(1 - v_i \cdot z_j)$ as our measure of distance. Note that both the descriptors $v_i$ and $z_j$ are $L_2$ normalized.

### 4.3.1 Retrieval with inflexible alignment

Assuming that the motions in two sequences are performed at the same speed, and the dot-product is a good measure of similarity, we can write the overall similarity between a video and a mocap sequence as cross-correlation

$$s_\delta = \sum_{i=-\infty}^{\infty} v_{i+\delta} \cdot z_i$$

(4.1)

where $\delta$ represents the shift needed to align the two sequences. This similarity for all possible shifts can be written in the form of a 1-d cross-correlation along each dimension

$$s(v, z) = \sum_{i=1}^{d} v_{\delta} \ast z_{\delta}$$

(4.2)
where $\star$ is the cross-correlation operator. This expression can be very efficiently computed by taking the signals to the Fourier domain. Let $\mathcal{F}^{-1}(.)$ be the inverse Fourier transform function and $V_i, Z_i$ be the discrete Fourier transforms of $v_i$ and $z_i$ respectively.

$$s(v, z) = \mathcal{F}^{-1}\left(\sum_{i=1}^{d} V_i^\ast \odot Z_i\right)$$

(4.3)

where $\odot$ is the element-wise multiplication operator and $\ast$ denotes the complex conjugate. Finally, we take a max over $s(v, z)$ to obtain the matching score (used to rank the mocap sequence) as well as the best alignment between the mocap and the video.

**Circulant Temporal Encoding (CTE)**

CTE [RDC+13] adds a filtering stage to the computation of cross-correlation to ensure that the peaks in $s(v, z)$ are salient and well-localized in case of a good match. CTE has been shown to be effective for video retrieval and copy detection. Similar to cross-correlation, CTE can be computed very efficiently in the Fourier domain as

$$s(v, z) = \mathcal{F}^{-1}\left(\sum_{i=1}^{d} \frac{V_i^\ast \odot Z_i}{V_i^\ast \odot V_i + \lambda}\right)$$

(4.4)

where the parameter $\lambda$ adjusts the regularization controlling the “peakiness” of the response. It also stabilizes the solution by making sure that the denominator in Equation (4.4) does not collapse to zero.

The original implementation of CTE removes high frequencies after DFT to compress the representation. CTE also uses Product Quantization (PQ) [JDS11] to reduce the memory requirement and speed up the similarity computation. In our case, we do not use frequency pruning or PQ to keep the CTE matching comparable to other techniques using a complete, uncompressed representation. For more CTE implementation details see [RDC+13].

### 4.3.2 Flexible alignment

The methods described above assume similar speeds of action in the query and the database. To allow for some temporal flexibility in alignment we also try flexible
matching techniques based on Dynamic Time Warping (DTW).

Again let \( v \) and \( z \) be the query and the database sequence respectively. The function \( \text{dist}(i,j) \) returns the distance between the two frames \( v_i \) and \( z_j \) of the respective sequences. A warp can be described using a pair of functions \( \phi(.) = \{ \phi_v(.), \phi_z(.) \} \) that maps aligned frame indices to the original \( v, z \) frame indices. The distance between the warped sequences can be defined as

\[
d(v,z) = \sum_{i=1}^{T_\phi} \text{dist}(\phi_v(i), \phi_z(i)) \tag{4.5}
\]

where \( T_\phi \) is the warped length after alignment. A warp path is the sequence of indices \( (\phi_v(i), \phi_z(i)) \), for \( i \) from 1 through \( T_\phi \).

**Dynamic Time Warping (DTW)**

In the original DTW formulation, the warp path is constrained such that we do not skip any frames and only move forwards in time during alignment, i.e., \( (\phi_v(i+1), \phi_z(i+1)) - (\phi_v(i), \phi_z(i)) \) equals either \((1,0), (1,1), \) or \((0,1) \) \( \forall i \). This property enforces monotonicity and continuity of DTW’s warp paths, which are needed for a well-behaved alignment of time-series (for details see [RJ93]). Another important consideration in DTW is the end-point constraint that can be written as

\[
\phi_v(1) = 1, \phi_v(T_\phi) = n; \phi_z(1) = 1, \phi_z(T_\phi) = m, \tag{4.6}
\]

and forces the algorithm to use the full length of both sequences.

Given these constraints, DTW solves the minimization problem

\[
\phi^* = \arg\min_{\phi} d(v,z) \tag{4.7}
\]

using dynamic programming to obtain the optimal warp path \( \phi^* \). The distance function \( \text{dist}(.) \) is used to fill a matrix \( D \in \mathbb{R}^{n \times m} \), where each element \( D(i,j) \) is the distance between \( v_i \) and \( z_j \) (see Figure 4.3 for an example \( D \) matrix). DTW aggregates the cost of matching subproblems in a cumulative cost matrix \( C \) such that \( C(n,m) \) returns the score of the optimal alignment between the original sequences. Figure 4.3 also shows the initialization and the update rules for filling
matrix $C$. Note that, unlike CTE, the time and memory requirements of DTW scale quadratically in the sequence length ($O(n^2)$, assuming $n = m$). Even though DTW returns the lowest-cost warp path, because of the aforementioned end-point constraint it cannot localize a query within a larger database file. Therefore, in its original form, DTW cannot be applied to our task of retrieval and alignment.

**Subsequence DTW**

Müller [Mü07] presents a modified version of DTW, called Subsequence DTW (SS-DTW), that removes the end-point constraint. Instead of finding the lowest-cost alignment using both the sequences, we can compute

$$\arg\min_{k,l,\phi} d(v, z_{k:l}),$$

(4.8)

where $z_{k:l} = [z_k, \ldots, z_l]$ is a subsequence of $z$, i.e., $m \geq l \geq k \geq 1$. The solution to this problem can also be obtained using a dynamic program, as shown in Figure 4.3. Note the changes, compared to DTW, in the initialization of the cumulative cost matrix and calculation of the final score. Conceptually, this can be thought of as placing two “special frames”, at each end of the query sequence $v$ that can align with any extra frames in $z$ for a cost of 0.

**Normalization by the warp path length**

Although SS-DTW loosens the end-point constraint, it is biased towards matching shorter database subsequences. The increased warp path length due to choosing a longer subsequence (when multiple query frames match to a one database frame) severely penalizes the SS-DTW objective. Since it is just as likely for a motion occurring in a database sequence to be slower than a motion occurring in a query sequence as it is the other way around, this asymmetry in flexibility is problematic. To get rid of this bias we consider a new objective that normalizes the distance by the warp path length

$$\arg\min_{\phi} \sum_{i=1}^{T_v} \frac{1}{T_{\phi}} \text{dist}(\phi_i(i), \phi_z(i)).$$

(4.9)
Figure 4.3: A toy example to illustrate various DTW-based flexible alignment algorithms. The grey boxes in the cumulative cost matrices $C$ represent the chosen warp path for the same distance matrix $D$ shown on the left. Each algorithm initializes some cells of $C$, fills the rest according to a recursion formula, and then chooses a final score $\text{dist}(v,z)$ for the alignment. The set of indices $\{(i-1,j), (i-1,j-1), (i,j-1)\}$ is abbreviated as $(i,j)^*$. In the case of SLN-DTW, $P(i,j)$ is the length of the chosen normalized warp path for the subproblem up to $D(i,j)$.

Note that $T_\phi$, the length of the warp path, is dependent on $\phi$. This problem is known as the normalized edit distance problem [MV93], which can again be solved using dynamic programming. However, in this case the dynamic program needs another dimension for path length, leading to a run time of $O(n^3)$ in the sequence length, which is not suitable in a retrieval setting.

Since this expensive normalization is not practical, we use a local normalization approach (SLN-DTW) [MGB09]. This $O(n^2)$ approximation ($O(n)$ when built on top of FastDTW [SC04]) works well in practice. Rather than building a three-dimensional cumulative cost array — over query frames, database frames, and the warp path length — SLN-DTW just keeps track of the path length (obtained greedily) so far in a separate matrix $P$. See Figure 4.3 for the update rule. Note that the initialization depicted here differs slightly from the original SLN-DTW initialization. Although this algorithm does not return an optimal path, as defined by Equation 4.9, it often finds a warp path very close to the best normalized path on our data (see Figure 4.4).
Figure 4.4: A comparison between exact normalization (as defined in Equation 4.9), using dynamic programming over a 3d cumulative cost array, and approximate normalization used in SLN-DTW [MGB09]. The corresponding distance matrix \((D)\) is shown behind each warp path, with darker shades representing smaller distances. We show results using (a-b) 2 different video queries matched against a mocap sequence and (c) two sequences of length 100 containing random, \(L_2\)-normalized 256-dimensional vectors. We notice that the approximate local normalization works fairly well in all three cases, while being several times faster. Note that we have kept the end-point constraint here for simplicity.

Smoothing warp paths

Since the original DTW objective is the sum of distances between warped frames of the two sequences, the warp path is regularized. Adding an extra frame in the warp path adds a positive value to the overall cost. But once we normalize the cost by path length, the objective has no preference for shorter path lengths. As a result, normalization leads to “jagged” warp paths. However, a smooth warp path is more desirable because it better describes the distortions due to variations in speed. To achieve smoothness, we use simple slope weighting [RJ93] which applies local constraints by associating small multiplicative costs to horizontal and vertical movements in the warp path. Thus, the final update rule becomes

\[
C(i, j) = \min \left\{ \frac{D(i, j) + w^* C(i, j)^* P(i, j)^*}{1 + P(i, j)^*} \right\}
\]

where \(w^*\) are weights \(\{w_{10}, w_{11}, w_{01}\}\) for indices \((i, j)^* \equiv \{(i - 1, j), (i - 1, j - 1), (i, j - 1)\}\) respectively. We choose \(w_{01} = w_{10} = 1.15\) and \(w_{11} = 1\) for all our
4.4 Experiments

To quantitatively evaluate the retrieval performance of different approaches described above we propose a new benchmark called $V3DR^2$. Since it is hard to obtain ground truth 3d pose for the query videos, and since action category is known to be a good proxy for 3d motion [YGG12, YKC13], we define similarity between a video query and the retrieved sequence using their action labels. In other words, we assume that each action is defined by a series of poses over time.

4.4.1 V3dR: Video-Based 3D Motion Retrieval Benchmark

The benchmark contains a series of video queries, a database of 3d motion sequences (mocap), ground truth annotations and two protocols for evaluation.

---

Figure 4.5: Example queries from YouTube videos provided in the $V3DR$ dataset. Notice the realistic clothing and backgrounds that are not typical in videos collected in a laboratory setting.

experiments. We refer to the resulting technique as $SLN-DTW (+smooth)$. 

2 http://www.cs.ubc.ca/labs/lci/v3dr/
Queries

We select two sets of queries: videos captured in a controlled environment, as well as a more challenging set of videos downloaded from YouTube. To factor out occlusion and camera motion we choose videos where the full body is visible and there is little or no camera motion. The videos are typically short snippets of 1 to 4 seconds in length. Figure 4.5 shows some of the frames from example video sequences.

Another set of video queries come from the IXMAS dataset [WRB06]. IXMAS consists of short video sequences where 10 actors perform simple actions in a laboratory setting. This dataset is typically used to evaluate cross-view action recognition, and contains videos captured from 5 distinct camera views (see Figure 3.3). While most of the YouTube videos have similar camera elevations, these video snippets from IXMAS add variation in viewpoint. The queries are picked randomly from cameras 0−3, excluding camera 4 due to its low elevation angle (measured with respect to the positive z-axis).

Both sets contain the following 8 classes: sit down, get up, turn, walk, punch, kick, pick up and throw overhead. We choose these classes because they are commonly featured in the mocap dataset that we use as our database. Each group of queries has 20 videos per action, with a single person performing the action. This amounts to a total of 160 queries per set. The dataset also provides manually annotated bounding boxes around the person in each video.

Mocap Database

We use the CMU-mocap dataset [cmu] — the largest publicly-available mocap database that we are aware of. From the 2549 sequences, we kept the 2000 shortest, which results in around 4.5 hours of 3d human motion. To make mocap and video data comparable we sub-sample each mocap sequence to 24 fps.

Ground truth

We annotated the mocap sequences using the same action labels as the queries. Since a mocap sequence may contain more than one action performed in sequence (e.g., walk, turn, sit-down), we annotate each file per frame. These annotations
Figure 4.6: An example of mocap annotations provided in the V3DR benchmark. At the top, we show a few frames of a mocap sequence. The corresponding action labels are shown below. Note that the annotations are not temporally exclusive. As shown above, one frame can have multiple labels. We use these annotations to evaluate video to mocap alignment.

are not necessarily temporally exclusive (e.g., a person might walk and turn at the same time, and so some frames may have both of these labels. 976 sequences were annotated with at least one class, leaving 1024 sequence not having any of the above action classes. Figure 4.6 illustrates an example annotation from V3DR.

Evaluation metrics

Given a query, we retrieve the top \( N \) mocap files based on their similarity to the video. We choose two different metrics to evaluate performance on V3DR, which correspond to different use cases of video-based mocap retrieval. The first evaluation protocol is called detection modality. In this modality, a retrieved example is counted as a true positive if it contains the action featured in the video.

In the second protocol, localization modality, retrieval is expected to produce both a ranking and a frame number that localizes the action in each 3d sequence in time. A retrieved 3d sequence is counted as a true positive if it contains the queried action \textit{and} its localization is correct. To evaluate localization, we compare the query label against the ground truth mocap annotation at the frame number returned by the algorithm.

We evaluate our detection modality using mean Average Precision (mAP) which is defined as the mean of the APs over all queries, and serves as a single number to evaluate performance per class. We evaluate our localization modality using recall@\( N \) curves [JDS11]. For each query, we plot the number of true positives in the top \( N \) retrieved sequences over the total number of positive examples in the
Figure 4.7: Recall for different feature types. For the same descriptor length, relational pose features significantly improve recall over trajectory-based motion features. And, a concatenation of pose and motion features performs comparably or better than the individual features alone. The improvements in recall using Motion + Pose in the case of realistic data (youtube) indicates that when the 2d pose estimation is not reliable the motion information can be more useful.

database. This results in a monotonically increasing curve for increasing $N$. We show the average of these curves over all queries.

4.4.2 Results

The goal of the evaluation is to study the effect of features as well as matching techniques on the retrieval performance. We also show qualitative results to illustrate strengths and weaknesses of different retrieval algorithms. Note that, to report alignment in the case of cross-correlation (CC) and CTE we return the mocap frame matching the central video frame of the query; for SS-DTW and SLN-DTW we use the mocap frame at the middle of the warp path.

Effect of features

First, we study the performance of the pose and motion features described in Section 4.2. See Figure 4.7. We use the same matching algorithm (cross-correlation or CC) for all the feature types and keep the total number of dimensions of the descriptor fixed to 256. We use the localization modality and plot recall for different
Figure 4.8: We compare the performance of the cross-correlation (CC)-based retrieval with CTE for different values of regularization parameter (λ). We plot recall, averaged over all queries, for different values of N (number of retrieved examples). Note that the recall improves for increasing λ, but CC performs better on most N for both the query sets. This observation indicates that the regularization provided by CTE does not help the retrieval performance in our case.

values of N (number of retrieved examples). We observe the same trend in the detection modality as shown in Table 4.1 (see columns corresponding to CC). Since the concatenation of motion and pose descriptors performs better than the individual features, we use Motion + Pose descriptors for all following experimentation.

Retrieval and matching performance

We compare the performance of the two inflexible alignment techniques, cross-correlation (CC) and Circulant Temporal Encoding (CTE), in Figure 4.8. For CTE, we try different values of the regularization parameter λ. We observe that though CTE has proved to be effective for copy-detection in videos, CC works better for our retrieval task. Higher values of λ dominate the denominator in Equation 4.4, which corresponds to reducing regularization, eventually approaching CC. Henceforth we use CC as our inflexible retrieval technique and do not include CTE in the results.

Next we present a detailed analysis of our retrieval results for the flexible alignment methods — SS-DTW and SLN-DTW. Figure 4.9 shows the results on localization modality, and Table 4.1 displays mAP values for the detection modality. In
both cases, we average the results over IXMAS and YouTube. We add Gupta et al. [GMLW14] as a baseline to compare our techniques with the first published method on this task. Gupta et al. use only motion features (aggregated using BoW instead of Fisher Vector), and retrieve mocap using CTE. This baseline serves to show the cumulative improvements based on the techniques suggested in this chapter. A significant gap between the ideal performance (the black dotted line) and our best methods points at the difficulty of the mocap retrieval task. We note that SLN-DTW(+smooth) outperforms other matching approaches overall. However, the results are not consistent across different action classes. This result points to the complementary nature of these methods and requires further exploration.

Finally, we present a confusion matrix to show commonly confused classes for the two best performing retrieval approaches SLN-DTW (+smooth) and SS-DTW in Figures 4.10 and 4.11. This matrix is different from the confusion matrix commonly used to evaluate a multi-class classification. Here, instead of classification accuracy or error, we show the recall at \( N = 100 \) with different classes as retrieval targets. Therefore, each row does not sum to 1. We note that it is challenging to distinguish examples of actions such as turn, punch, and throw. Also, get up and sit down are commonly confused as pick up.

**Qualitative results**

We demonstrate the quality of our recovered alignments in Figure 4.12 and 4.13. A few successful top matches for the considered methods (CC, SS-DTW, and SLN-DTW) are shown in Figure 4.12. We note that the flexible methods lead to better alignment in the illustrated examples. The error cases are shown in Figure 4.13. We observe that although actions involving the full body or a significant change in pose are relatively easy to match, activities such as throw remain hard to disambiguate.

4.5 Discussion

In this chapter, we have presented an efficient method for alignment and distance computation between a short video query and a mocap sequence. We also formalize the problem of video-based mocap retrieval by introducing a challenging benchmark, V3DR, and by proposing metrics for quantitative evaluation on the
Recall on the localization modality of the video-based mocap retrieval benchmark averaged over all YouTube and IXMAS queries. The black dotted line depicts the ideal recall curve, and the magenta dotted line shows recall for randomly retrieved examples. These two curves act as the upper and the lower bound on the performance for each class. All matching techniques use motion + pose features except Gupta et al. [GMLW14]. Note that, flexible matching techniques (SS-DTW and SLN-DTW) perform better on most classes.
Figure 4.10: The confusion matrix for average recall (at $N = 100$) over all queries using SLN-DTW + smooth for retrieval with pose + motion features. Each row shows the recall@N for the queries from the action category, and columns depict the target category used to calculate recall. Therefore, the diagonal corresponds to the curves shown in Figure 4.9. Note the confusion of sit down and get up with pick up. This is due to the visual similarity of these actions. We also note that the categories with isolated body movement, e.g., kick, and throw overhead, are much harder to retrieve reliably. Also, the category turn is challenging to distinguish, possibly because of the subtle change in the pose during the action compared to a significant change in case of sit down, get up, and pick up.
1 Sit down 30 0.015 0.038 0.043 0.172 0.162 0.169 0.155
2 Get up 56 0.028 0.076 0.076 0.231 0.225 0.203 0.219
3 Turn 194 0.098 0.148 0.161 0.229 0.222 0.195 0.230
4 Walk 739 0.373 0.598 0.624 0.596 0.666 0.585 0.669
5 Punch 13 0.007 0.032 0.027 0.033 0.054 0.070 0.059
6 Kick 23 0.012 0.017 0.023 0.015 0.022 0.042 0.027
7 Pick up 76 0.038 0.147 0.177 0.335 0.352 0.392 0.438
8 Throw oh. 17 0.009 0.015 0.019 0.027 0.023 0.035 0.027
mAP 0.072 0.134 0.144 0.205 0.216 0.211 0.228

Table 4.1: Per-class and overall mean Average Precision (mAP) on the detection modality of the video-based mocap retrieval benchmark. We show an average performance using both IXMAS and YouTube queries. # ex. is the number of files in the database containing the given action. Chance corresponds to the expected performance of uniformly random retrieval. We highlight the best value in each category with boldface and underline the second best value. Again, we observe that using pose + motion (P+T), significantly improves the retrieval performance over motion-based features (T). Also, the flexible alignment techniques (SS-DTW and SLN-DTW) perform the best on most action classes in comparison to cross-correlation (CC).

task. V3DR provides frame-level action annotations for around 400 000 frames of the CMU-mocap dataset. Video queries with variety in viewpoints, realistic clothing and backgrounds are also provided as a part of the benchmark. We hope that V3DR will encourage further research in video-based mocap retrieval.

We have also shown that DT and relational pose features, previously used for action recognition, are also effective for human motion retrieval. This finding allows us to use DT along with any state-of-the-art 2d pose detector trained on images (rather than videos) to retrieve similar mocap sequences using videos. Our approach does not depend on any additional training data such as synchronized video and mocap examples. This feature is important because acquiring such data in a realistic setting remains a challenge.
As seen in the qualitative results, we can obtain the matched viewpoint for the retrieved mocap snippet. One of the main limitations of V3DR is that it does not yet measure the accuracy of a viewpoint prediction. In the future, we would like to extend V3DR to include viewpoint annotations, to allow evaluation of the matched viewpoint.

Also, the mocap retrieval for action categories such as punch and throw is challenging and often results in poor alignments for the top retrieved examples. Since our features are completely hand-crafted and are not trained to discriminate between these categories, they fail to adequately distinguish the isolated body part movements and subtle changes in pose. Therefore, our approach leads to poor retrieval in some cases. To improve feature representation for retrieval, we
Figure 4.12: A few representative alignments for YouTube videos (best viewed in color). The query frames and corresponding frames of the retrieved mocap sequences are shown (right limbs are marked in red). For each retrieval algorithm, we display the top ranked true-positive. **Top:** Walking sequences are relatively easy to match. All the algorithms perform well on this example. **Middle:** In this pick up sequence, the flexible matching algorithms can capture the bend down and get up movements. However, CC only aligns with the final get up movement. **Bottom:** Again, in this kick sequence, we get a better alignment using the flexible matching techniques.
Figure 4.13: Some of the typical error cases for video-based mocap retrieval. We use SLN-DTW with Pose + Motion features for both the examples. We show the top three aligned matches along with the top-ranked true positive inside the green box. **Top:** In this case the throw overhead action is best matched to a dance move where the person has their arm lifted, similar to the query video. **Bottom:** The query comes from a turn sequence. Here the top ranked sequences are again dancing and walking. In both cases, we do find the appropriate matches, but they are poorly ranked.
can either perform metric learning on our hand-crafted features (similar to Ren et al. [RSH+05]) or reuse deep learning-based descriptors trained on a similar task such as action recognition [KTS+14]. Both of these directions can potentially improve features, and lead to a significant gain in the retrieval performance.

Another related problem is that we discard the uncertainty in the 2d pose estimation to evaluate our pose descriptor. We can potentially use the heat-map of different body joint locations, instead of the final MAP estimate that we currently use. Incorporating uncertainty of observation in a retrieval pipeline is another interesting direction to explore.

Finally, our evaluation relies on frame-level action annotations of mocap sequences. Since actions depend on the context and the interaction with other objects, these labels can be ambiguous, e.g., for a mocap sequence, the activity of cleaning a window and waving can be easily confused and mislabeled. A 3d-to-3d matching score between mocap (such as [MRC05]) sequences can be a more objective measure of similarity than discrete labels. Also, we can use a similarity measure between mocap sequences to mine examples with similar motion, and speed up the labeling process.
Chapter 5

Localized Motion Trellis

In this chapter we introduce the Localized Motion Trellis (LMT) — a novel non-parametric approach to 3d pose estimation from monocular RGB video. Our focus is on generating realistic 3d pose and motion without knowing the human activity label at each time instance in the video. Given a video sequence featuring a person as input, we first retrieve snippets with a similar motion from a large collection of mocap files. We efficiently combine these short motion exemplars into a continuous 3d pose sequence that best explains the image evidence. We demonstrate our approach by estimating articulated 3d motion of players in a challenging sequence from a broadcast sports video. Since we only use real motion examples, the resulting poses are anthropomorphic, and the overall motion is realistic.

The LMT distinguishes itself from the state-of-the-art methods for 3d pose estimation from monocular video in that it does not require action-specific pose or motion priors, and uses exemplar-based motion synthesis as a model to estimate human pose. As a result, video or mocap used for training does not need any action labels. This property is desirable because in the case of complex activities such as sports, annotators need to have deep domain knowledge — making it cumbersome and expensive to get labeled examples. Additionally, since no labels are needed, we can use a much larger set of examples. For instance, we use 2000 files from the CMU mocap dataset (approximately 4.5 hours of mocap) for our 3d pose estimation pipeline.
5.1 Overview

In the last chapter we introduced a method for retrieving mocap snippets given a short monocular video as a query, which can also be thought of as example-based 3d pose estimation from video. However, for longer sequences (more than a few seconds), the same approach cannot be applied. As the length of the video increases, it becomes harder to find a similar sequence in mocap. One way to deal with this problem is to divide the video into overlapping chunks and retrieve examples for each short clip. We can later combine the retrieved mocap sequences using interpolation. Analogous to Tracklets (e.g., as used in [ZLN08]) for object tracking, the top retrieved examples can serve as a mid-level representation of the full motion over the video. However, the best match for each video snippet may not always emerge as the top match due to a) noise in the features used for retrieval, and b) the inherent ambiguity in recovering 3d motion from a 2d image sequence.

We address this problem using context. Apart from the spatial structure of the body (often represented as a tree), there is a high temporal structure to human motion. In this chapter we show that the LMT can be used to model longer term temporal context (e.g., person running does not stop suddenly) and spatial continuity over time (e.g., the body joints move in smooth trajectories) to help us recover a consistent and realistic output sequence.

Given a video sequence, we divide it into overlapping subsequences. For each of these videos as queries, we search for similar examples in a mocap database (as described in the Section 4.3). The LMT allows us to model continuity between neighboring mocaps to build a time-forwards graph. We then construct a minimum energy path through this graph using dynamic programming to generate a 3d pose sequence similar to that seen in the input video (see Figure 5.1).

5.2 Localized Motion Trellis

First, we divide the input video sequence into $k$–frame chunks with $p$ overlapping frames between the neighboring chunk. For each of these subsequences, we calculate a combination of pose and motion features (as described in Section 4.2) for retrieval. We set cross-correlation (CC) as our matching algorithm for retrieval (Section 4.3.1) because of its simplicity and competitive performance with respect
Figure 5.1: Localized Motion Trellis (LMT). (a) The input is a monocular video sequence. (b) We use each overlapping subsequence to (c) search a large collection of mocap files for similar motion sequences. (d) The retrieved 3d snippets are connected in time to form a trellis graph. (e) The minimization of energy over this graph produces a smooth 3d output that best explains the image evidence. Being a model-free method, the LMT can estimate 3d motion in sequences with multiple activities, which overcomes one of the major limitations of current approaches.
to DTW-based methods. In the last chapter, for each query we only matched one 
example per mocap file. This restriction is not required in case of the LMT. The 
matches can come from a single file as long as they are not the same. Therefore, for 
the LMT we keep our retrieval algorithm the same, except that we allow retrieving 
multiple non-overlapping matches per file.

We build a trellis graph with the top-N retrieved 3d snippets as nodes and con-
nect all neighboring nodes with directed edges (forward in time). Since each node 
is temporally aligned to the video input, we call this graph the Localized Motion 
Trellis or LMT. The weights in an LMT consist of both unary and binary terms; 
the former are meant to enforce a high similarity with the image evidence and the 
latter encourage smooth transitions over time. The resulting graph is similar to a 
Motion Graph (MG) [KGP02] typically used in Graphics to synthesize human mo-
tion. Before describing different energy terms in the LMT, we highlight some of its 
key differences to MGs:

- The MG and the LMT both have short mocap snippets as nodes, and the con-
  nection between nodes describes the transition from one node to the next. 
  But, in the case of MGs, the graph is generated before the user input is ob-
  served, and, to limit the number of connections, a threshold is used on the 
  node similarity to decide if an edge should be added, while in the case of the 
  LMT, the graph is generated on the fly based on the input. The retrieval step 
  selects the nodes to include in the graph, and no threshold is required to limit 
  connectivity.

- Another significant difference lies in the search strategy. The search in an 
  MG finds a path in the graph that maximizes the objective defined by the 
  user. However, the search can become cumbersome when the graph is large. 
  Therefore, often an MG is constructed using small carefully chosen exam-
  ples. In contrast, we restrict our method to a time-forwards graph, with di-
  rected edges and no loops, which can be searched efficiently. Also, retrieval 
  acts as a filter to limit the number of nodes, allowing us to use a larger mocap 
  database to construct an LMT.

- Another subtle but crucial point is transitions. For an LMT, the transitions are
constrained because each retrieved snippet corresponds to a user input frame by frame. In the case of an MG, since the graph is generated a priori, the allowed transitions in the graph may mismatch with the user input. To allow for some flexibility Ren et al. [RSH+05] insert a few nodes and extra edges to construct an augmented motion graph. However, any addition of nodes and edges leads to increased complexity of the search. Also, for MGs the generated graph needs to be pruned to avoid a few undesirable cases where the search cannot progress, e.g., nodes with no outward edges.

5.2.1 Unary terms: accounting for image evidence

To evaluate the likelihood of a retrieved mocap snippet given the video subsequence, we investigate three unary terms. We calculate these terms for each node in the graph.

**Matching error**

We define the matching score as the normalized sum of the per-frame similarity of the video and mocap sequences, where an aggregation of features represents each frame. These features are the same as the ones used for retrieval (described in Section 4.3). Formally the matching error can be written as

\[
E_m = K - \frac{1}{p} \sum_{i=1}^{p} \langle v_i, z_i \rangle,
\]

where \(v_i\) and \(z_i\) represent the aggregated features of the aligned video and 3d motion sequences over \(p\) frames. \(K\) is a positive constant to convert the matching score into an error measure while keeping the error positive.

**2d pose error**

To evaluate the quality of the predicted pose, we calculate the distance between the joints of the 3d sequence projected onto the corresponding image and the output of a 2d pose detector (the FMP [YRI3]) on the image. Let \(f_i\) be all the 2d joint locations returned by FMP over \(p\) frames and \(g_i\) be the location of the corresponding
joint $i$ in a 3d sequence. The 2d pose similarity error is given by

$$E_{2d} = \frac{1}{m} \sum_{i=1}^{m} w_i \cdot \| \mathbf{f}_i - p_{\theta,T}(\mathbf{g}_i) \|_2^2,$$

(5.2)

where $m = 14 \times p$ is the number of common joints between the model of FMP and the CMU-mocap dataset, and the weights $w_i$ assign importance to each joint, accounting for the fact that some 2d joint predictions are more reliable than others. The function $p_{\theta,T}(\cdot)$ projects the 3d sequences to the image plane, as seen from viewpoint $\theta$ (note that $\theta$ is known from retrieval) under orthographic projection. $T$ is the translation that best aligns the projection of the mocap frame to the FMP output by aligning the centroid of the hip and the shoulder joints. Since we already use hip and shoulders to align skeletons, we do not use them for error computation i.e., set $w_i = 0$ for corresponding joints. We also exclude head and neck joints as they move very little about the torso. All other joints are weighted equally except the wrist, which has the relative weight 0.5. We make this choice based on the observation (see Figure 5.4) that wrists are often poorly localized using 2d pose detectors, as compared to other joints.

### Path error

We borrow this score from the original Motion Graph (MG) formulation [KGP02]. In MGs, the motion is synthesized by indicating a path on the ground, and the algorithm walks the MG minimizing the difference between the hip projection of the character and the path specified by the user. In our case, the path on the ground can be approximated using a tight bounding box around the person in each frame and the homography of the scene — assuming a flat ground (more details in Section 5.3.2). Formally, the path error is given by the average distance between the points on the player path and the projection of the 3d model center onto the ground. We align the centroids of the two set of points before calculating the error.

$$E_p = \frac{1}{p} \sum_{i=1}^{p} \| \mathbf{t}_i - \mathbf{m}_i \|_2^2,$$

(5.3)

where $\mathbf{t}_i$ is the point on the player path and $\mathbf{m}_i$ is the corresponding projection of the 3d model center on the ground plane at frame $i$. 67
5.2.2 Binary term: ensuring contextual output

The binary term of the LMT encourages a smooth 3d output in space and time. Unlike the unary terms, the binary term defines errors on the overlapping frames of retrieved 3d snippets from neighbouring video queries.

Transition error

This term is identical to the transition error used in MGs [KGP02]. Let \( x_i \) and \( x'_i \) be the 3d location of the joint \( i \) in the overlapping frames of two neighboring nodes. The transition error for each frame is given by summing the error over all \( m \) joints over the overlapping frames

\[
E_t = \frac{1}{m} \sum_{i=1}^{m} \omega_i \cdot \| x_i - q_t(x'_i) \|_2^2 ,
\]

(5.4)

where \( q_t(\cdot) \) is the translation function corresponding to the best alignment of the two 3d snippets parametrized by a translation vector \( t \). Since we only use the same subset of joints as used in 2d pose error \( m = 14 \times p \). \( \omega_i \) is the weight indicating the importance of each joint. We set equal weights for all joints except the elbow and the wrist with relative weights 0.5 and 0.25 respectively to avoid over-penalizing the transition error due to the localization error for these joints. Also, we set \( \omega_i = 0 \) for the head because with the LMT we hope only to capture overall coarse body movement instead of subtle details such as the movement of the head or toes.

5.2.3 Search and 3d pose estimation

The final expression for the overall unary energy is given by

\[
E_u = w_m E_m + w_{2d} E_{2d} + w_p E_p
\]

(5.5)

which acts as the node potential. \( w_{2d} \), \( w_p \), and \( w_m \) are the weights for 2d pose error, path error, and matching error respectively. Similarly, we weigh the binary term with \( w_t \). The binary term is the potential for the LMT edges that is computed for all pair of neighbouring nodes. Subsequently, we find a path through this graph with the smallest sum of these weighted energies. Since the LMT uses a trellis graph, we
can efficiently calculate the minimum energy path using dynamic programming.

The chosen path contains a sequence of 3d mocap snippets. Note that retrieval also returns an azimuthal orientation for each of these snippets. We bring all the snippets in a single global reference frame and stitch them together into one 3d pose sequence. First, we align the returned 3d snippets using translation. To align two neighbouring sequences we ensure that the mean of the joint locations in the overlapping frames coincides. Once all the sequences are aligned, we interpolate body joint rotations (using the quaternion space) in overlapping regions to smooth the transition from one sequence to the next.

5.3 Experiments
We demonstrate the task of synthesizing players’ articulated 3d movements in a monocular sports video sequence. We choose the domain of team sports because players often perform multiple activities (e.g., running, walking, turning, shooting) one after the other — a scenario that has not been well-explored in the previous literature. Also, it is challenging to annotate these video sequences with either 3d pose or action labels per frame. Thus, relying only on the unlabeled data, our approach is particularly well-suited for such an application.

5.3.1 Data and evaluation
We choose a 500 frame sequence from a professional basketball game, taken from a broadcast video. We have chosen the sequence such that there are no scene transitions (i.e., it is one continuous shot). Also, since it is a wide angle shot, all the players are visible in the frame throughout the sequence.

Since it is not possible to reliably annotate 3d pose on a monocular video, we annotate each frame with the 2d pose for evaluation. Instead of evaluating 3d pose error, we project the 3d pose onto the image and compare it to the 2d ground truth. To obtain 2d pose ground truth, we manually label 14 key body joints for all the players in each video frame.

We show the quantitative results for the pose error using the normalized Percentage of Correct Parts (Percentage of Correct Parts (PCP)) as defined by Sapp et al. [ST13], where the distances are normalized such that the torso is 100 pixels
Figure 5.2: Some of the common challenges with broadcast team sports videos. a) Even in case of a high-definition (HD) video, the player height in pixels is often less than 150 pixels in a wide-angle shot. b) There are motion blur artifacts due to the camera motion required to follow the game. c) Also, severe occlusions are common in team sports.

tall. There is one key difference in our evaluation to make it more reliable: our 2d pose annotations distinguish between the left and the right body part. Unlike other 2d pose evaluations, our localization for each joint counts as correct only if it falls within the distance threshold of body part and matches the correct side of the body.

We note that our approach focuses on estimating 3d articulated motion and not on localization, we choose the best translation of our predicted 3d pose before computing PCP. We use the mean of the left and the right hip of the ground truth
2d pose to place the projection of the estimated 3d pose onto the video.

Assumptions

We make some simplifying assumptions. First, we assume that the players have been successfully tracked, and that a tight bounding box is provided for each player per-frame. In our experiments, we use bounding boxes obtained using manual labeling. We also assume that we have a homography of the court at each frame, so we can stabilize the video, accounting for camera motion; this minimizes noise in the optical-flow trajectories that we use for retrieval, and is also necessary to compute the path error (Section 5.2.1). The homography is also manually labeled using a simple GUI to mark correspondences between the court (using the standard NBA dimensions) and each video frame. In the domain of sports, we could alternatively obtain the homography using automatic camera rectification methods [GLW11, GZA12, CC15]. However, the assumption helps us to separate the problem of camera auto-calibration from the rest of the pipeline.

Even with these assumptions, the broadcast videos are challenging for 3d pose estimation. There is significant camera motion in the form of pan, tilt, and zoom. Also, the lossy compression of the video leads to many video compression artifacts. We summarize some of the main challenges in dealing with these videos in Figure 5.2.

5.3.2 Implementation details

Here we describe some of the implementation details that are crucial for reproducing the results presented in this chapter.

Obtaining path on the ground

We use player path in the world coordinates to estimate path error (as described in Section 5.2.1). Since the player bounding box and the frame homography are given, we can obtain an approximate path on the ground. Given a tight bounding box with the top-left corner \((x_o, y_o)\), width \(w\) and height \(h\), we set the player’s location in the image as \((x_o + w/2, y_o + 0.95 \times h)\) (see Figure 5.3(a)). We project this point into the field map using the homography to get a player path in the world.
Figure 5.3: The LMT implementation details. (a) We assume that player bounding box is given at each frame. The red dot is our estimate of the player location based on the current bounding box. The blue line shows the estimated path connecting these locations over time. We transform this path to world coordinates using the homography. (b) We also use the homography at each frame to estimate the camera viewpoint. The left figure shows a square around a player projected to the image using the given homography (solid-magenta) and a local affine approximation to homography (cyan-dotted). Since the camera is located far from the court, the approximation is reasonably accurate in this case. We use the approximate affine transformation to obtain the elevation and the azimuthal angle of the camera under the orthographic projection (right).

coordinates.

**Estimating camera viewpoint**

Again, using the homography of the frame we obtain an approximate viewpoint of the camera for each player location. Since we assume an orthographic projection, we only need to estimate two parameters — elevation and azimuthal angle. We use this viewpoint to calculate the 2d pose error (see Section 5.2.1). Also, the estimated elevation angle helps us reduce the search for viewpoints during retrieval. In the case of video-based retrieval (Chapter 4), we match each video to 3 different projections of mocap sequences along the elevation angle. Here, we can
set the elevation angle to a fixed value. Finally, we also use the estimated elevation and azimuthal angles to synthesize the final 3d pose sequence (described in Section 5.2.3).

To obtain a viewpoint estimate in the video, we locally approximate the homography around a player’s location using an affine transformation. First, we consider a fixed sized square around the player in the world coordinates and project it to the image using the homography. Next, we estimate an affine transformation using the four corners of the square in world coordinates and their projections in the image. Given the affine transformation, the elevation and azimuthal angles can be easily calculated (see Figure 5.3(b)). Note that, since the viewpoint changes for different positions on the court, we need to calculate this for each player location separately.

**Parameter tuning**

There are a total of 10 players in the video. We randomly pick 2 players — one from each team in the game — as our validation set. We use their video sequences to tune the LMT weight parameters \( \{w_{2d}, w_p, w_m, w_t\} \). We do a few rounds of grid search to find parameters that return the best PCP curve on the validation set. We note that the performance is not sensitive to small changes in the above weights. All the test results are averaged over the remaining 8 players. We fix the number of top retrieved examples \( N \) to 500.

We also measure the sensitivity of our results to the LMT parameters — temporal window size used for retrieval (\( k \)) and the number of overlapping frames between two neighboring windows (\( p \)). For more details see Section 5.3.4.

**5.3.3 Baselines**

We compare our method against the Flexible Mixture-of-Parts (FMP) [YR13] and the n-best maximal decoders of Park and Ramanan (nFMP) [PR11], which produce smooth outputs over FMP using temporal consistency. We set the number of n-best solutions to 50, and tune the \( \alpha \) parameter of their objective function (see Equation 7 in [PR11]) on our data. We run both FMP and nFMP in the bounding boxes for each person, and not on the whole frame. As mentioned earlier, we use the ground truth body center to place the projection of our 3d model onto the image. For a fair
comparison, FMP and nFMP algorithms should also be aware of the ground truth body center that the LMT is using. But, the primary focus of this result is not to show that the LMT is better at localizing 2d joints but to give a sense of the quality of the LMT output alongside other 2d methods. LMT’s strength lies in generating natural looking 3d pose sequences that we illustrate in our qualitative results.

We also set-up an oracle called Oracle-pose to give us an upper bound on the performance. This method uses the same approach as LMT but assumes that the ground truth 2d pose is given to the algorithm. The assumption helps us to generate noise-free relational pose features for retrieval. The rest of the pipeline and parameters are kept the same. We use the oracle as an upper bound on accuracy.

5.3.4 Results

For quantitative evaluations, we plot Percentage of Correct Parts (PCP) for different thresholds, i.e., the fraction of predicted joint locations that fall within the threshold distance from the ground truth. For qualitative results, we choose a few sequences and plot the predicted 3d pose as well as its projection on the image.

Different LMT modes

First, we show the PCP results for different variations of our LMT implementation. Figure 5.4 shows the 2d joint localization accuracy for 4 of the joints (knee, foot, elbow, and wrist). We obtain the PCP results for the parameter configuration \( \{w_{2d}, w_p, w_m, w_t\} = \{1, 0.25, 1.2, 1.2\} \). Also, we set the LMT parameter \( k = 35 \) and \( p = 10 \). The same configuration is used for all the experiments.

We make following observations from these results:

- First, we observe that results are consistent across these joints. The consistent gain in accuracy in the full model (unary + binary terms) shows that the LMT is effective in utilizing the contextual information present in the pairwise relationships to resolve ambiguities.

- The comparison with the oracle-pose shows the potential gain in accuracy that we could achieve by improving 2d pose estimation.

- The performance of path-only is consistently below all other variations tried.
Figure 5.4: Percentage of Correct Parts (left-right sensitive) for different variations of the LMT compared against Oracle-pose. Top match only uses the matching error score. Path only minimizes only the path error to find the best 3d sequence for each video snippet. Unary adds 2d pose error along with path and matching error. Full model uses a weighted sum of both unary and binary terms to evaluate the final LMT path. Oracle-pose has access to the ground truth 2d pose in addition to using the same parameters as the full model. We note that adding binary terms (in the full model) leads to a consistent gain in PCP. It demonstrates the importance of pairwise relations in resolving ambiguities.
Figure 5.5: Percentage of Correct Parts (PCP) for the LMT output compared to 2d pose estimation methods FMP [YR13] and nFMP [PR11]. Since FMP and nFMP do not distinguish between left and the right body parts, we do the same for the LMT projections to make the comparison fair. Again, the oracle-pose uses the same parameters as Ours but has access to ground truth 2d pose.
This result shows that the path of the person on the ground — although used as the user input for synthesis in the case of MGs — is not effective in video-based pose estimation. We must incorporate the pose information.

We show the qualitative results for pose estimation in Figures 5.6, 5.7, 5.8, and 5.9. The main observations from the qualitative results are:

- As shown in Figure 5.6 and 5.7, the LMT gracefully deals with transitions between different activities such as walk, turn, and run. We are also able to capture the walk cycle in most cases, i.e., the correct leg is in front in a walk or run sequence. Additionally, the person orientation is matched well. Overall, the LMT output gives us rich information about the player movement and direction in 3d.

- Our full model returns a smoother result, while the unary-only output is easily affected by the local errors in the 2d pose estimation (as shown in Figure 5.6). The temporal context provided by the binary term in the full model also helps in resolving the left-right ambiguity (Figure 5.7).

- Sometimes, the full model may lead to over-smoothing. Figure 5.8 shows one such example.

- Finally, even though the LMT works well for matching coarse motion, and getting the direction of the movement right, the method often fails to recover fast, complex movements in the video as shown in Figure 5.9.

2d pose estimation performance

We also compare the LMT PCP results with FMP and nFMP baselines in Figure 5.5. We use the same 4 joints as the previous experiment. However, to make the PCP results comparable, we give up the left-right sensitivity of our evaluation, e.g., the elbow is considered a single joint rather than two different joints, left and the right elbow, for the purpose of evaluation. This change is necessary because FMP and nFMP do not distinguish between the left and the right body parts.

We observe that the LMT output is comparable or better than the nFMP results for higher thresholds, but it does not perform well when high precision is required.
in localization. Again, this indicates that while LMT is good at capturing the overall coarse movements of a person in 3d, it does not localize the individual body joints very well. We can observe the same pattern in our qualitative results. The reason for a larger error at low thresholds is that the LMT formulation does not focus on localizing individual joints. We always match the full 3d model from one of the exemplars which may match the overall motion, but may not correspond well to the locations of the body parts. Also, the LMT uses a fix-sized 3d model to generate the pose sequence for all the player. Since the players have some variation in their body shape and size, this assumption adds an error that we do not compensate for in the current approach.

LMT parameters

Next, we study the sensitivity of different LMT parameters on the final performance. The two important parameters related to the construction of the LMT are the length of each snippet used to query the database and the number of overlapping frames between two consecutive queries. Figure 5.10 summarizes the results.

5.4 Discussion

To the best of our knowledge, the LMT is the first scalable approach to 3d human pose estimation from realistic monocular video input. Motion retrieval along with the simple time-constraint trellis graph architecture keeps our method efficient and widely applicable. We make no assumptions about the used motion examples and the availability of class labels. Even though exact body joint locations are hard to estimate, the LMT can provide information about the activity, walk-cycle, and the heading direction of the person in the video.

5.4.1 Limitations and future work

One of the main limitations of the LMT is its inability to generalize outside the available exemplars. For instance, we do not provide any mechanism to customize the shape or size of the puppet for the tracked person in the video. The LMT is also not able to spatially localize the estimated 3d pose sequence in the provided video, and hence, we need to rely on the availability of the body center. Even though
we only use the exemplars and perform no further optimization to improve the joint localization, the LMT output can also act as an initialization for further fine-tuning for better 3d pose estimation. To begin with, we can optimize for person size and the location of the output. Also, to make pose level adjustments, we should explore a combination of non-parametric and parametric approaches such as GPDM [WFH08] or parametric motion graphs [HG07].

Also, the LMT is inherently limited by the accuracy of the retrieval pipeline, which in turn depends on the 2d pose estimation accuracy. The large gap between the performance of the oracle and our best result shows that improving 2d pose estimation can lead to better 3d pose output. We can make multiple changes to improve 2d pose estimation. First, we do not exploit the continuity of the appearance, i.e., we detect 2d pose in each frame independently. We can build an appearance model using the initial tracking results to improve the 2d pose tracking for subsequent iterations. Second, as mentioned in the last chapter, we can incorporate the uncertainty of pose prediction in the retrieval pipeline rather than using the MAP estimate for the 2d pose.

Furthermore, we have tested the LMT in a relatively restricted setting of a sports environment with a known geometry. Extending this work to videos-in-the-wild would also be an interesting future direction.
Figure 5.6: A typical 3d pose sequence generated using Localized Motion Trellis. The top row shows a cropped image sequence from the NBA data. The subsequent rows display the output of different methods, including our full mode (using both unary and binary terms). The 3d output corresponding to the full model is presented in the last row. Note that we have rotated the axis in the final row to emphasize the 3d nature of the output. The arrow on the middle frame shows the viewing direction of the camera. Also, we mark the right limbs in red. In this sequence, the player walks, then turns to his right and starts running. Our full model smoothly handles the transition from one activity to the next and accurately captures the walk cycles i.e., left and right legs are correctly aligned even though this information is not available from nFMP. The unary output is affected by the errors in the nFMP pose estimate (see the fourth frame from the left and the last frame), while the full model can correct for these errors.
Figure 5.7: Another example of the 3d pose sequence generated using Localized Motion Trellis. In this sequence, the player runs towards the camera, then turns left. After waiting for a few seconds, he turns around and runs again. Note that the full model is correctly able to capture the walk cycle (see the last three frames), while the player's direction is inconsistent in the unary output (see the sixth frame from the left, and third to the last frame).
Figure 5.8: An error case for our full model. In this example, the player runs, turns around, and keeps going in the same direction. He then stops for a few frames, and the turns left. Our 3d output from the full model faces in the wrong direction when the player turns and keeps facing in the wrong direction when the player is standing. This example is one of the cases when the full model may be failing due to over-smoothing. In contrast, the unary output is able to capture the change in direction.
Figure 5.9: Another error case emphasizing a common limitation of all the LMT variants. In this example, the player is performing fast and complicated movements. Even though the LMT can capture the overall jump motion (frames 2-6), the output does not capture any other movements. There can be multiple reasons for this failure such as a) the absence of an appropriate exemplar in the database that can reasonably approximate this motion; or, b) a failure in retrieval.
Figure 5.10: The effect of the LMT parameters on the average PCP (over the wrist, elbow, knee, and foot joints) for different thresholds.  

a) We plot accuracy as the function of query length. We reconstruct the 3d pose using 50 matches for each video subsequence. Since we are interested in measuring the effect of query length on the quality of retrieved mocaps sequences, we only use 2d pose error to find the best path in the LMT, and no interpolation is done to generate the final 3d output. Based on this result, we choose $k = 35$.  

b) Average PCP as a function of overlap between consecutive queries. In this case, we fix the query length to 35 frames and calculate PCP using the top 500 matches for each video subsequence. To avoid the effect of other terms, we use only transition error to find the best path.
Chapter 6

Discussion and Future Work

Understanding actions, emotions, and intentions of people around us is an important part of human communication. This non-verbal exchange is essential to cooperation, teamwork, and relationships. Intelligent machines also need the ability to understand non-verbal cues if they are to help us and work alongside us. Keeping this motivation in mind, we have focused on the problem of automated human activity understanding in videos.

In Chapter 1, we argued that many crucial applications require a detailed description of human activities under realistic imaging conditions. The three specific problems that we targeted in this thesis are cross-view action recognition, video-based mocap retrieval, and 3d pose estimation. Cross-view action recognition helps in overcoming the viewpoint bias in action recognition, mocap retrieval allows us to efficiently search through a large database of mocap files, and 3d pose estimation in videos can provide a well-localized (in space and time) description of human activities. Next, we identified one of the main obstacles in building such systems — human effort required in labelling data for supervised learning. The rest of the thesis concentrated on various techniques for overcoming this challenge.

In the following sections, we discuss the potential impact of our contributions, identify main limitations of presented approaches, and speculate on possible future directions to address these challenges.
6.1 Contributions and Impact

*View-invariant action recognition:*

- Training data collected from the internet has its biases. For instance, the perspective of a security camera, an autonomous car or a home robot may not match well with the videos downloaded from YouTube. Therefore, a view-invariant representation of human action is important.

- In Chapter 3, we have demonstrated a novel method to add view-invariance to action recognition without using any multi-view data or human annotations. Our approach has shown a significant improvement over the baseline on a standard cross-view action recognition benchmark and has remained competitive to the state of the art.

- We have also presented a method to generate motion features from mocap sequences without photo-realistic rendering. These features are analogous and comparable to the popular dense trajectory features [WKSL11].

- We expect that these contributions will further encourage research towards learning from synthetic sources of data such as CAD models, computer games, and physics-based simulations of the world.

*Video-based mocap retrieval:*

- Retrieval of mocap examples given visual input has been used for 3d pose estimation. Chapter 4 has shown that instead of retrieving individual frames (as demonstrated in [RSH+05]), we can retrieve a sequence using a short video clip as a query. In addition, we have established a new task of video-based mocap retrieval and demonstrated its applications to monocular 3d pose estimation (in Chapter 5) and cross-view action recognition (in Chapter 3).

- We have also presented a set of features for retrieval that is comparable across video and mocap (Section 4.2). Due to temporal aggregation of information in videos, even noisy pose estimates in individual frames are effective in retrieving mocap sequences.
Moreover, we have provided a new benchmark\(^1\) for evaluating video-based mocap retrieval to allow standardized comparisons in the future. Our frame-level annotations to mocap can also be useful for other vision tasks.

3d pose estimation in videos:

- Pose has been shown to be a good feature for activity recognition [JGZ\(^+\)13, CLS15]. There are many advantages to predicting pose per-frame instead of a single action label for the whole video. For instance, given a long video we can segment the corresponding pose sequence to recover a set of action labels along with the temporal localization [MBS09]. Although annotating pose is tedious, to obtain fine-grained labels, action annotations require greater domain knowledge, e.g., people who do not watch basketball may not be familiar with the different kinds of shots in the game.

- Chapter 5 has presented a non-parametric (example-based) method for 3d pose estimation that is scalable to a large number of exemplars and realistic monocular videos.

- A large body of past research in 3d pose estimation has focussed on parametric methods and used constrained environments for testing. We hope that this thesis will help shift the focus towards example-based method and unconstrained videos.

6.2 Future Directions

6.2.1 Robust evaluation of 3d pose estimation

As we noted in the introduction, it is challenging to obtain ground truth for 3d pose in videos. Synchronized video and motion capture can be used to get precise 3d pose, but it is currently restricted to indoor settings only. Moreover, the appearance of such a video is not natural because of the special suit and reflective markers

\(^1\)http://www.cs.ubc.ca/labs/lci/v3dr/
required for tracking. Markerless mocap systems allow for natural clothing but require a large number of carefully calibrated cameras.

Recently Elhayek et al. [EAJ+15] used a small number of calibrated cameras to estimate 3d pose in an outdoor setting with natural lighting conditions, and showed impressive results. Using inertial sensors can add further robustness to such vision-based methods. However, for all the methods above, we still require an interface to clean-up the data and correct mistakes using human input. We can construct a pipeline for obtaining 3d pose ground truth in a realistic setting using multiple cameras, inertial sensors, and a manual clean-up step.

Another challenge lies in measuring the similarity between the predicted and the ground truth pose sequences. Obtaining an accurate ground truth estimate can help us in getting a reliable numerical similarity measure. However, a numerical similarity measure such as $L_2$ distance between the two pose vectors may not imply closeness in the semantic space of activities [MRC05]. The problem of finding a measure of similarity that agrees with human intuition is an interesting area for further exploration.

### 6.2.2 Features for video-based mocap retrieval

The features that we propose for video-based mocap retrieval are effective as demonstrated in Chapter 3 and Chapter 5. However, there is a huge gap between our best results and the upper bound on the recall@N performance (Figure 4.9). One of the potential bottlenecks in the performance can be the features used for comparing a mocap and a video frame. Our features are hand-designed, as opposed to the learned features commonly used for action recognition and pose estimation. Although a retrieval method does not have access to labeled examples for learning, the features learned on similar tasks can be reused for retrieval. Here are some concrete suggestions for improvements:

- We use Flexible Mixture-of-Parts (FMP) for 2d pose estimation. Our results (in Figure 5.4) suggest that an increase in 2d pose estimation accuracy can improve 3d pose estimation. As a first step, we can replace FMP with a ConvNet-based 2d pose detector (such as [TGJ+15]) with a better performance on standard benchmarks.
• Our motion descriptor for retrieval is also an aggregation of hand-designed Dense Trajectories (DT) features. These features can be replaced with ConvNet-based optical flow features used for action recognition [SZ14a]. One of the ways we can adapt our method to ConvNet-based features is by generating synthetic optical flow using different projections of the mocap sequence (similar to [ZRSB13]).

• Additionally, we use relational pose features based on the 2d pose detection output and calculate similarity using dot product. It is possible to transform the feature space such that it becomes discriminative with respect to 3d pose. We can use metric learning [SJ03] for this purpose. It is possible to learn a distance metric, without human supervision, by considering short mocap sequences and their different 2d projections. Since it is easier to establish similarity in 3d pose space, we can use the 3d similarity to supervise metric learning for 2d relational features.

6.2.3 Fine tuning pose estimation in videos

The Localized Motion Trellis (LMT) constructs the 3d pose output by interpolating multiple mocap snippets chosen from a database of mocap examples. One of the main limitations of such example-based methods is their inability to generalize beyond the examples. Therefore, the output has a limited range, in the set of poses, skeleton shapes, and motion patterns. We can make our approach accurate by adding a fine-tuning step at the end of the pipeline. Here are some of the potential future directions:

• One of the most challenging and fundamental problems in human motion analysis is to formulate a flexible, activity-independent generative model of human motion. To simplify the problem, we can instead begin with a bank of generative models to represent the wide variety of human activities. Similar to [BRMS09, YGG12, YKC13], we can use the output of the LMT to choose an appropriate model for each snippet and tune the model parameters to explain the image evidence.

• Many 3d pose estimation methods assume that the 3d model of the person
being tracked in known in advance. On the other extreme, we use the same model provided by the mocap sequence for each subject. The mismatch in the size and shape of the model of the person introduces an error in the 3d pose estimate at each frame. The fine-tuning step can also customize the model to each subject. We can use a parametric model of human shape and pose [ZB15] rather than using a fixed-sized puppet. Again, we can start with a generic shape with the pose sequence returned by the LMT as our initialization and optimize for shape over the whole sequence.

- Finally, one of the biggest limitations of our approach is that it is not probabilistic. For instance, we do not model or incorporate the uncertainty in the 2d pose estimate. Some pose or specific joint estimates can be less reliable than others due to self-occlusion or motion blur. Some of the past approaches deal with this problem by optimizing for 3d pose and updating the underlying 2d pose estimate in an alternating fashion [SSQTMN13, ZZL+16]. Incorporating these ideas into our pipeline is another exciting challenge.
Bibliography


[cmu] Carnegie Mellon University Motion Capture Database. → pages 10, 26, 31, 50


[EMJZF12] Marcin Eichner, Manuel Marin-Jimenez, Andrew Zisserman, and Vittorio Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. IJCV, 99(2), 2012. → pages ix, 5, 16


94


[LTSY09] Rui Li, Tai-Peng Tian, Stan Sclaroff, and Ming-Hsuan Yang. 3d human motion tracking with a coordinated mixture of factor analyzers. *IJCV*, 87(1-2), 2009. → pages 21


[NTTS06] MN Nyan, Francis EH Tay, AWY Tan, and KHW Seah. Distinguishing fall activities from normal activities by angular rate characteristics and high-speed camera characterization. *Medical Engineering & Physics*, 28(8), 2006. → pages 4


