Human Pose Retrieval for Image and Video collections

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

by

Nataraj Jammalamadaka
200999002
nataraj.j@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2017

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Human Pose Retrieval for Image and Video collections" by Nataraj Jammalamadaka, has been carried out under our supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. C. V. Jawahar

Adviser: Prof. Andrew Zisserman

To my parents

Acknowledgments

For me, research as an interesting vocation started at CVIT under the guidance of Prof. C. V. Jawahar over fifteen years ago. From then to now, I am fortunate to have a continued association with him. He has taught me the work ethics required to do serious research and continuously pushed me to raise the bar. Beyond research, his several expositions about various aspects of professionalism continue to benefit my career.

From reading the book "Multiple view geometry" during bachelors for computer vision lectures to actually working with the author, Prof. Andrew Zisserman, closely has been a wonderful journey. From his various instructions to me and from several hours of thinking together, I think I somewhat understand what it means to do world-class research. I will continue to reflect on our interactions and imbibe several aspects of it into my research methodologies. I particularly appreciate the meticulousness required in good research.

This PhD would not have been possible without both my advisers Prof. C. V. Jawahar and Prof. Andrew Zisserman and I thank them for their guidance over the years. I also thank my collaborators Prof. Vittorio Ferrari and Marcin Eichner. I am always inspired by Vitto's energy and creativity.

Visiting University of Oxford as an exchange student for about one and a half years and interacting with students and faculty at Visual Geometry group has been one of my best life experiences. For making this experience so lovely, I thank many of the PhD students at VGG: Yusuf, Relja, Amr, Karen, Yuning, Tomas, and Ken. I particularly acknowledge the trio who made up the bulk of my social and other intellectual pursuits: Varun Gulshan, Arpit Mittal and Rohan Paul. With Varun, I particularly cherish discussions on topics such as politics, society and life. With Rohan, I am amazed and will always remember his dedication and effort towards making a cost-effective cane for visually challenged in developing and under developed countries. I wish him great success for his various social projects. With Arpit, I always felt like I never left home. I thank Yusuf Aytar and Ahsan Alvi for helping me in building the GUI for the web-demo. I thank the post docs at the time who always answered my questions: Dr Andrea Vedaldi, Dr. Matthew Blascho and Dr. Lubor Ladicky. I thank the support staff at VGG and department of engineering for helping me with visas. I thank Oxford, the university town, for being such a wonderful and pleasant place.

For having spent a third of my life on IIIT campus, it was my second home. My B.tech years have been formative time for my career and during this time I have met several interesting people and diverse set of personalities. I particularly would like to acknowledge my social circle who had direct effect on

me: Gopalakrishna, Pranav, Manohar, Anil, Kranthi, Ranganath, Bala, Jitendra, Aravind, Balakrishna and Keerthi. I particularly acknowledge Gopalakrishna, Pranav and Manohar for their unintended role in me pursuing research. Gopals passion for research, Manohars work ethic and Pranavs elaborate and incisive thinking have in some ways inspired me to pursue research. I also would like to thank friends and colleagues at Sarnoff labs: Ranjeeth, Aparna, Sravanthi, Manika, Nalin, Prakash, Hitesh, Gajender, Aastha, Manish, Ankit and Binny, for inspiring me in several ways.

In my experience, PhD is a solitary pursuit (besides the interactions with advisors) for the large parts and one needs socially and intellectually compatible group of individuals to cope with it. During my PhD, I am fortunate to have found several such individuals at IIIT who made the journey easier. Anand Mishra and I, being co-passengers, have become good colleagues and friends over a period of time discussing various aspects of research and life. As the strength of our lab increased, I had good affinity with the BITS gang (Aniket, Sourabh and Pritish), Vijay Kumar, Praveen Krishnan, Yashaswi Verma and Omkar Parkhi. I thank all of them for their interactions with me. I thank Aniket Singh in particular for giving me time to discuss various ideas during my time at IIIT and helping me with Theano. During my PhD, I also had opportunities to interact with students several years junior to me: Digvijay Singh, Ayush Minocha, Mayank Juneja, Siddarth Chandra. Thank you for reminding me that I am a Dinosaur in making. Although we have very little overlap, I thank Pramod Shankar for his various wisdoms.

I would like to thank Prof. PJN, Prof. Jayanthi and Prof. Anoop for creating such a wonderful lab (CVIT). I would like to thank the support staff at IIIT and CVIT who over the years have helped me in several matters. In particular I would like to thank Kishore and Appaji in Admin department and Satya and Phani at CVIT. I also gratefully acknowledge Rajan's and Nandini's contribution in annotating the pose datasets.

I thank my parents for always giving me their support and for their constant patience. I thank my sister and brother-in-law for being there for me. I also would like to thank my little niece Gayatri for bringing joy into our family. She is the first baby that I interacted closely with. I wish I could have made her part of my pose datasets.

Finally, I would like to thank everyone who has contributed towards my thesis and my life but have not been acknowledged.

Abstract

With overwhelming amount of visual data on the internet, it is beyond doubt that a search capability for this data is needed. While several commercial systems have been built to retrieve images and videos using meta-data, many of the images and videos do not have a detailed descriptions. This problem has been addressed by content based image retrieval (CBIR) systems which retrieve the images by their visual content. In this thesis, we will demonstrate that images and videos can be retrieved using the pose of the humans present in them. Here pose is the 2D/3D spatial arrangement of anatomical body parts like arms and legs. Pose is an important information which conveys action, gesture and the mood of the person. Retrieving humans using pose has commercial implications in domains such as dance (query being a dance pose) and sports (query being a shot). In this thesis, we propose three pose representations that can be used for retrieval. Using one of these representations, we will build a real-time pose retrieval system over million movie frames.

Our first pose representation is based on the output of human pose estimation algorithms (HPE) [5, 26, 80, 103] which estimate the pose of the person given an image. Unfortunately, these algorithms are not entirely reliable and often make mistakes. We solve this problem by proposing an *evaluator* that predicts if a HPE algorithm has succeeded. We describe the stages required to learn and test an evaluator, including the use of an annotated ground truth dataset for training and testing the evaluator, and the development of auxiliary features that have not been used by the (HPE) algorithm, but can be learnt by the evaluator to predict if the output is correct or not. To demonstrate our ideas, we build an evaluator for each of four recently developed HPE algorithms using their publicly available implementations: Andriluka *et al.* [5], Eichner and Ferrari [26], Sapp *et al.* [80], and Yang and Ramanan [103]. We demonstrate that in each case our evaluator is able to predict whether the algorithm has correctly estimated the pose or not. In this context we also provide a new dataset of annotated stickmen. Further, we propose innovative ways in which a pose evaluator can be used. Specifically, we show how a pose evaluator can be used to filter incorrect pose estimates, to fuse outputs from different HPE algorithms, and to improve a pose search application.

Our second pose representation is inspired by poselets [13] which are body detectors. First, we introduce deep poselets for pose-sensitive detection of various body parts, that are built on convolutional neural network (CNN) features. These deep poselets significantly outperform previous instantiations of Berkeley poselets [13]. Second, using these detector responses, we construct a pose representation that is suitable for pose search, and show that pose retrieval performance is on par with the state of the art

pose representations. The compared methods include Bag of visual words [85], Berkeley poselets [13] and Human pose estimation algorithms [103, 18, 68]. All the methods are quantitatively evaluated on a large dataset of images built from a number of standard benchmarks together with frames from Hollywood movies.

Our third pose representations is based on embedding an image into a lower dimensional pose-sensitive manifold. Here we make the following contributions: (a) We design an optimized neural network which maps the input image to a very low dimensional space where similar poses are close by and dissimilar poses are farther away, and (b) We show that pose retrieval system using these low dimensional representation is on par with the deep poselet representation.

Finally, we describe a method for real time video retrieval where the task is to match the 2D human pose of a query. A user can form a query by (i) interactively controlling a stickman on a web based GUI, (ii) uploading an image of the desired pose, or (iii) using the Kinect and acting out the query himself. The method is scalable and is applied to a dataset of 22 movies totaling more than three million frames. The real time performance is achieved by searching for approximate nearest neighbors to the query using a random forest of K-D trees. Apart from the query modalities, we introduce two other areas of novelty. First, we show that pose retrieval can proceed using a low dimensional representation. Second, we show that the precision of the results can be improved substantially by combining the outputs of independent human pose estimation algorithms. The performance of the system is assessed quantitatively over a range of pose queries.

Contents

List of Figures

List of Tables

[1] *We are using the PCP measure as defined by the source code of the evaluation protocol in [26], as opposed to the more strict interpretation given in [71].*

Chapter 1

Introduction

The ubiquitous presence of video recording devices saw the exponential growth of the image and video data. This necessitates the need for understanding, indexing and retrieving images and videos. This process is broadly termed as content based information retrieval (CBIR). Humans in particular are objects of great interest in an image. Automatically identifying various aspects of humans have significant commercial applications in surveillance, retrieval from archives etc. Human pose is one such aspect which gives away the action. In this chapter, we describe the importance of image and video retrieval. We also describe what is human pose, how it can be used as a query modality and why it is important for retrieval. The scope and organization of the thesis are stated at the end of the chapter.

1.1 Image and Video Retrieval

The video and image cameras are present in our offices, roads and malls. They are present in our homes, on our office table, in our mobile phones and with us during a vacation. And they generate a lot of data. It has thus become very important to search and sift through them for accessing memories, to see what our favorite celebrity is wearing, or to catch a perpetrator of a crime, a terrorist even. Manually searching the video and image collection is no more possible. While text has been the primary modality through which images are searched, it has severe limitations. Text annotations for most of the videos and images are either very sparse or not present at all. Thus it has become important to search the image using its content.

In the last decade, several methods have been proposed [67, 69, 86] to search images from a database. The seminal work in this line of research is by Sivic and Zisserman [86], where the query is an object marked in an image and all similar instances in the database are retrieved. Several other works have been built on top of this, improving the performance and speed. Other line of research has been to predict the object labels [24, 55] present in the image and thus generate annotations. Some works even generate phrases [47] and sentences [97]. Using these descriptions, the standard text based retrieval methods can be used straight away.

Image and video retrieval systems traditionally have three components: (i) User Interface, (ii) Image/Video Descriptor generator and (iii) Search and Indexing algorithms. Many state-of-the-art systems have additional components such as user feed back mechanism through which results can be improved and personalized. The "User interface" is responsible for taking the query as input and displaying the retrieved results. The most common query modalities are query-by-text and query-by-exemplar image. The user interface facilitates such modalities and displays the retrieved results along with other meta data relevant to retrieved sample (e.g, google gives a URL of the source). The primary concern of this component is to make the user interaction easy, intuitive and aesthetic. The "Search and Indexing" algorithms, which have long history in Algorithms and Information retrieval community, are responsible for finding the best match for the query in the database. The primary objective of this component is accuracy and speed. The norm is to retrieve the results in tens of milli-seconds. From the computer vision standpoint, the most important component is the "Image/Video Descriptor generator". Depending on the entities that are being retrieved, such as general objects, flowers, faces e.t.c., this component has to ensure that each image is represented by feature vector such that images with similar entities are close by and dissimilar ones are farther away. The primary objective of this component is to filter out the irrelevant information, extract the relevant information and encode it effectively and efficiently.

1.2 Human Pose as a Query Modality

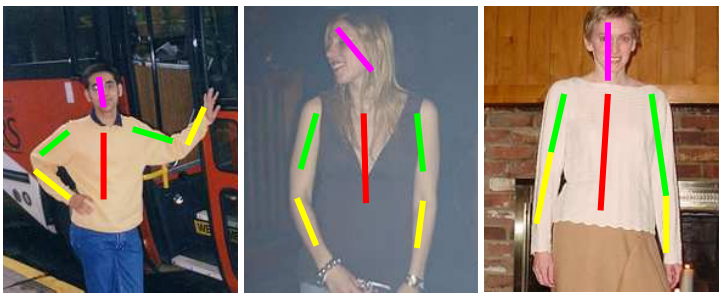


Figure 1.1: **Pose estimates by HPE algorithms.** The six upper body parts, estimated by the algorithms, are color coded as pink for head, red for torso, green for upper arms and yellow for the lower arms.

We explore an alternative modality to query the images. In place of text or an example image, we use human pose as a query modality. ***Human pose*** is the spatial arrangement of human body parts as illustrated in figure 1.1. This is not to be confused with the "pose" of an object which is position and orientation relative to a co-ordinate system. Pose is an important indication of action (e.g., walking, having tea etc.) and sometimes even moods (e.g., brooding with hand under the chin, wiping tears etc.). Thus having human pose as a query modality helps in complementing the existing ones. For certain video collections like dance and sports, human pose as a query is much more intuitive modality.

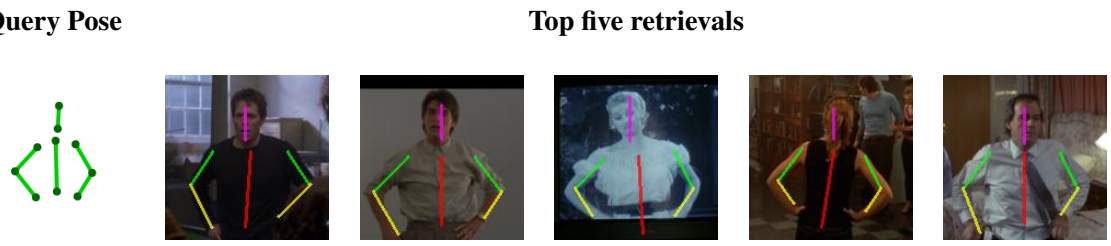


Figure 1.2: **Pose retrieval example:** The query pose (leftmost image) is represented by the upperbody stickman figure. It is followed by the top five retrieved images where the people present have the most similar pose as the query.

The problem of human pose retrieval is very challenging owing to the complexity of human body. This is evident from the fact that human pose detection has over 40 years of history in computer vision [43, 76, 34, 74, 26, 90]. These are some of the challenges with the problem of human body pose:

1. Variation in color and texture: A human can wear different types of clothes with a great mix of colors and textures. Hence the HPE algorithm has to be appearance invariant.

2. Variation in location, orientation and size: The human can occur any where and of any size in the image. In few scenes, like sports, humans can also occur at different orientations too. Hence the HPE algorithm has to be translation, rotation and scale invariant.

3. Variation due to articulation: Human body is not strictly rigid and have parts with a certain degree of freedom in movement. For example arms and legs can move even when the body is stationary. Hence HPE has to model the articulation properly.

In order to build the pose retrieval system, the following challenging problems have to be solved for whom there may not be any existing solution. The first and the most important challenge is to construct a pose-sensitive descriptor for an image with a person in it. The discriminability and comprehensiveness of the descriptor is the most critical aspect for the performance of the retrieval system. While HPE [5, 26, 103] algorithms exist, they are very unreliable. The challenge here is, How do we build descriptors from human pose detector's output when it is very unreliable? It is to be noted that HPE algorithms explicitly give the positions of the body parts. An important direction to explore is, is it possible to generate descriptors which implicitly encode the body joint positions? Another challenging problem is the query modality. How do we take a human pose as a query when describing a pose is very difficult and finding an image with desired pose is very tedious? Apart from these challenges, there are others such as indexing structures for nearest neighbor methods in very high dimensional feature space and the graphical user interface which need to be explored. During the course of this PhD, we explore these questions and propose several novel ways to solve the above challenges.

1.3 Objective of the Proposal

The objective of this proposal is to build the first prototypical large scale pose retrieval system. In this proposal, the following aspects related to this task will be explored in detail:

- **User Interface:** Traditional querying modalities for image and video retrieval are either not useful or too tedious for the pose retrieval system. The query-by-text is the most popular query mechanism where the user provides text as input. This unfortunately is not very useful in case of human pose where the description of the pose is very difficult to give. Query-by-Image is a popular alternative but is very tedious and limiting. In this mechanism, for a desired pose the user has to first find an image with that pose and then upload it to the system to obtain similar poses. This is somewhat paradoxical as there does not exist a efficient mechanism (as ours is the first prototypical system) to obtain an image with desired pose in the first place. Thus the first objectives of this work is to propose a query interface which is very natural and intuitive.

- **Descriptor Generator:** The backbone of any retrieval system are the descriptors. In the image and video retrieval systems, the bag of words descriptors have been very successful. Traditionally they have captured the gradient information. The major drawback with these descriptors is that a fine level spatial reasoning, a critical step for human pose, is not possible. Alternatives are the human pose estimation (HPE) algorithms which give the positions of the various body parts. Unfortunately, these algorithms are prone to frequent errors and hence unreliable. The second objective of this work is to improve the reliability of the HPE algorithms and to propose novel and compact pose descriptors based on HPE algorithms.

1.4 Scope of the Thesis

As described in section 1.1, the three important components of a retrieval system are (a) User interface, (b) Descriptor generator, and (c) the search and retrieval algorithms. In this work, we primarily concentrated on and contributed to "User interface (a)" and "Descriptor generator"(b) components. For " Search and retrieval algorithms", we used a state-of-the-art algorithm [83]. Further in the "Descriptor generator" component, we primarily focussed on methods which obtain an effective descriptor. For the several detection tasks (human detection and pose detection), we used the state-of-the-art methods.

1.5 Our Contributions

Our main contributions in this thesis are (a) a human pose dataset with more than 20,000 annotations, (b) introducing the concept of self evaluation algorithms, (c) introducing several methods to generate pose descriptors and (d) building the prototypical and large scale pose search system. Each of these are briefly described below and are discussed in great detail in the rest of the thesis.

1.5.1 Movie Stickmen Dataset

Learning human pose estimation requires large amounts of data. Towards this objective, we have created a dataset which consists of images from ten Hollywood movies. Our dataset consists of $5,645$ random frames from these movies. We have annotated a total of $11,639$ stickmen of which $3,764$ stickmen have all the parts visible.

1.5.2 Automatic Evaluator

A simple analysis of the previous works on pose estimation [74, 26, 80, 5, 103] revealed that they are not very robust. In the human upper body, while the head, torso and upper arms are reliably detected, the lower arms are frequently incorrectly detected. One of the reasons is that the background is confused with the body parts. For example, a strong edge near the body can be confused with an arm. To address this we have proposed a method to self-evaluate a vision algorithm [50].

Our method is designed for HPE, and considers several indicators all at the same time, such as the marginal distribution over the possible pose configurations for an image, imaging conditions, and the spatial arrangement of multiple detection windows in the same image. While other methods [12, 101] focussed on the performance at a dataset level, we focussed on predicting the performance on individual samples. Specifically, given a pose estimate, our algorithm computes a feature vector using the marginal distribution and other auxiliary information and classifies if the output is correct or not using pre-trained classifier.

1.5.3 Pose Descriptors

Pose descriptors are key to the performance of the human pose search system. In this thesis, we proposed the following three novel pose descriptors: (a)*HPE descriptor:* This descriptor is constructed after the image is processed using human pose estimation algorithms. Using the part angles obtained from the pose estimate, a very compact $12D$ pose descriptor is generated, (b)*Deep poselet descriptor:* This descriptor is obtained by pooling the scores of the deep poselet detectors. Our deep poselets, a variant of the poselet detectors [13], model a subset of body parts in a particular pose, and (c)*Deep pose embedding descriptor:* This descriptor is obtained by projecting the image into a $4096D$ space where similar poses are close by and dissimilar poses are farther away. The projection function is convolutional neural network, one of the deep learning methods.

1.5.4 Retrieval System

To demonstrate the viability of pose retrieval, we built a prototypical pose retrieval system [51] which is indexed on 22 movies containing three million frames. In this system, we have addressed the issues of (a) Reliability, (b) Scalability and (c) User interface for pose retrieval system.

- **Reliability:** As mentioned earlier, the human pose estimation is very unreliable. Typically pose estimation algorithms decide on one particular location for each body part and can potentially make a wrong choice. Clearly this affects the pose retrieval as a mistake in one part can effectively render this detection useless and can potentially worsen the performance of the retrieval system. To overcome this problem, we use the auto-evaluator proposed by us in Jammalamadaka *et al.* [50] and reject incorrect pose estimates.

- **Scalability:** The descriptors used in Ferrari *et al.* [40] have few thousands of dimensions. Indexing and retrieving such high dimensional pose representations of three million frames is challenging when fast retrieval is the objective. In our work, we use a much smaller representation of just $12d$ vector which can effectively capture the pose. For each of the six body parts, we compute the sine and cosine of the body part angle which form the descriptor. These descriptors are indexed using a randomized KD-tree to make the system scalable.

- **User interface:** The standard query mode for content based image retrieval systems is query-by-image. But it is very inconvenient to find an appropriate image every time a user queries the system. Thus we developed an intuitive interface, displayed in figure 5.2, consisting of interactive stickman. The sticks, which represent parts, can be re-arranged to construct the desired pose and the new stickman configuration becomes the new query.

1.6 Thesis Organization

The rest of the thesis is organized in the following way. In chapter 2, we discuss the previous work on pose retrieval and related work necessary to understand our contributions. While it may not be necessary to understand the material in this chapter, we recommend that this material be read as specific concepts and techniques that we use in our work are clearly explained. In chapter 3, we discuss about the new human pose dataset that we introduce and human pose datasets introduced by others that we use in our experiments. We also discuss the existing and proposed similarity measures for comparing two given poses. Further, we discuss the performance measures used in our experiments. In chapter 4, we describe the novel self evaluation algorithm for human pose estimation algorithms which significantly improves the precision. We also describe a method to obtain pose descriptor from a given human pose estimate. In chapter 5, we describe prototypical large scale human pose search system that we built and is currently online for the general public to experience. The aspects related to user interface and scalability are explained in detail. In the recent years, deep learning based methods have taken over computer vision with very impressive results. In chapters 6,7, we describe two methods to obtain pose descriptors using deep learning methods. In chapter 6, we propose a set of body part detectors which are sensitive to the pose. The detection scores of these body part detectors are pooled to obtain the pose descriptor. In chapter 6, we describe an embedding method which maps an image to a pose-sensitive space using CNNs [60]. Finally in chapter 8, we give concluding remarks.

Chapter 2

Background

In this chapter, we review the previous work on the topic of human pose retrieval. We also review the background necessary to understand the topics in the succeeding chapters. We primarily focus on those topics which we directly use in our work. In the below few paragraphs, we note the chapters in which they have been used. For the topics which are considered basic or have not been directly used in our work, we point to useful resources.

The most successful model for representing human body and its articulations are the pictorial structures [34] model. This method models parts and relations between them. Structured output SVM [94] is used for learning and belief propagation [38] is used for inference. Human pose estimation algorithms [5, 26, 80, 103], which we use in chapter 4, are based on these models. We describe the model and the inference algorithm in section 2.1. In particular, we describe how to efficiently compute marginals and max-marginals, both of which are used in chapter 4. For understanding the pictorial structures in its entirety, background in few topics in machine learning are needed. For the sake of completeness, we list the topics and sources from which they can be studied. Pictorial structure model uses Markov random field (MRF). MRFs are graphical models which are very useful in modelling the spatial relations between entities in an image. These tutorials [56, 10] give excellent introduction to MRFs. To learn the parameters, structured output SVMs, an extension of SVMs are typically used. This tutorial [16] gives a insightful explanation about SVMs and structured output SVMs can be studied from the following works [95]. For the features, the popular histogram of oriented gradients (HOG) [21] is typically used.

In the recent past, convolutional neural networks (CNNs) [60] has seen a resurgence and has performed extremely well on vision tasks. The current state-of-the-art human pose estimation algorithms [90, 92] use CNNs. Convolutional neural networks model an image as a composition of functions. Visually the model resembles a series of layers each of which is processed by a function to form a next layer. It is argued that CNN first models the low level features such as edges and joints and then expresses higher level features as a composition of these low level features. Our work in chapters 6,7 use CNNs for obtaining pose descriptors. Thus the CNN model, learning algorithm and a variant of CNN known as Siamese networks have been reviewed in section 2.2.

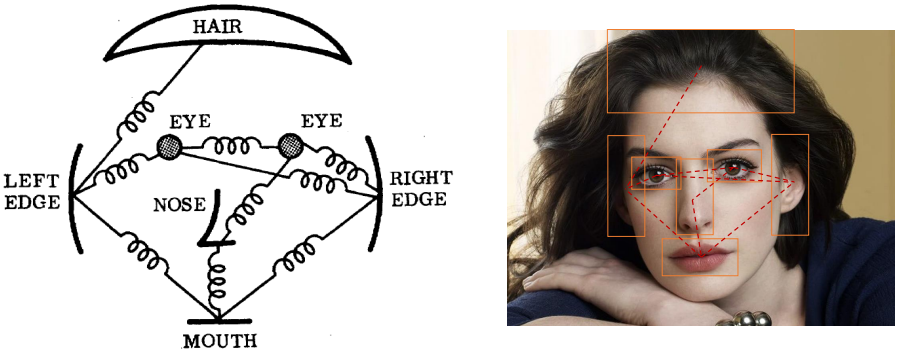


Figure 2.1: In the left image, the pictorial structure model for the face illustrated by a spring model in Fischler and Elschlager [43]. When viewed as a graph, the left edge, right edge and other parts on the face form the nodes. The edges between them are illustrated as a spring. The deformation function d_{ij}, like a spring, should be flexible in allowing the parts to take a wide range of relative positions while penalizing them as they move away from the expected relative positions. In the right image, the various parts of the face, which form the nodes in the graph, are depicted by the rectangles. The connections between them, which form the edges, are depicted by the red line segments.

An essential component in human pose retrieval is upper body detection. In section 2.3, we review two upper body detection methods: (i) poselets [13] and (ii) ETHZ UBD [28]. These methods are extensively used throughout the thesis. The abstract concept of poselet has been explained in section 2.3.1. Human pose estimation algorithms are naturally an important component in human pose retrieval. Human pose estimation can be defined as a task of locating the different anatomical parts of the people present in the image. It should also establish the correspondence between a particular person and his body parts. In this chapter, we describe several state-of-the-art human pose estimation (HPE) algorithms [5, 26, 45, 46, 74, 80, 90, 92, 103]. Finally, we review the previous work human pose search [40]. Human pose search can be defined as an image or video retrieval task using human pose as a query.

2.1 Pictorial Structures (PS)

Most of the objects are composed of several parts: tyres, wind shield, door and body compose a car and similarly eyes, nose, ears and hair compose a face. Many such parts are visually distinct and can help in locating the object. The pictorial structures, first proposed by Fischler and Elschlager [43], model the appearance of parts, inter-part spatial configuration spatial configuration. Since there are large intra-object variations, these relations have to be statistical or probabilistic models. In the following subsections, the statistical framework of the model, the learning algorithms and the inference algorithms are described.

2.1.1 The Model

The parts and their relations are modelled using the undirected graphical models. Each part is assigned a vertex v and each pair of object having a spatial relation are connected by an edge e_{ij} forming a graph $G = (V, E)$. Each vertex v_i corresponds to a random variable X_i and the graph of N vertices models the joint probability distribution $P(X_1, X_2, ..., X_n)$. Each random variable X_i takes the location l_i of the part as the state. An instance of an object is given by the configuration $L = (l_1, ..., l_n)$.

The object configuration L is scored by using two functions, the appearance model m_i and deformation model d_{ij}. The unary function m_i measures how well the image content at location l_i agrees with the appearance model of part i. The deformation function d_{ij} quantifies if the two parts are at an appropriate relative distance. The final score is the sum of unary functions over vertices and binary functions over all the edges. The optimal configuration has the minimum score and is given be the following equation:

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \tag{2.1}$$

While the above framework is from energy minimization point of view, a statistical setting would model the pictorial structures using probability distributions. Let θ be the parameters for describing the pictorial structures model. Given the image I and the parameters θ, the posterior probability of the configuration L is modelled as

$$p(L|I, \theta) \propto p(I|L, \theta)p(L|\theta). \tag{2.2}$$

The distribution $p(I|L, \theta)$ models the likelihood of the image generated given the configuration L. The distribution $p(L|\theta)$ models the probability of the configuration L and is called the prior.

The pictorial structures model is parameterized by $\theta = (u, E, c)$, where $u = (u_1, ..., u_n)$ are the appearance parameters, E are the set of edges connecting parts and c are the connection parameters. The likelihood distribution is factorized as follows and it depends purely on the appearance model,

$$p(I|L, \theta) = p(I|L, u) \propto \prod_{i=1}^{n} p(I|l_i, u_i). \tag{2.3}$$

The prior probability distribution is factorized as follows and it only depends on edges E and the connection parameters c_{ij} and is given by,

$$p(L|\theta) = p(L|E, c) = \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \tag{2.4}$$

Substituting the equations 2.3 and 2.4 into equation 2.2, we get

$$P(L|I, \theta) \propto \left(\prod_{i=1}^{n} p(I|l_i, u_i) \prod_{(v_i,v_j)\in E} p(l_i, l_j|c_{ij}) \right). \tag{2.5}$$

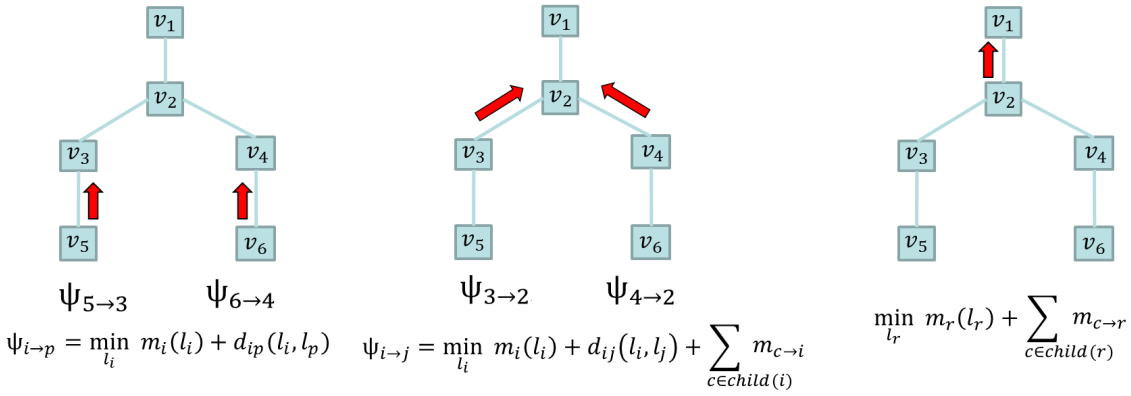


Figure 2.2: The example graph displayed above has six nodes with v_1 as the root, $\{v_5, v_6\}$ as the leaves and the rest as intermediate nodes. The belief propagation algorithm proceeds from the left to right as follows. First (left image) the messages $\psi_{5->3}$ and $\psi_{6->4}$ are computed at the leaf nodes $\{v_5, v_6\}$ using the equation 2.7. Next (center image) the messages at the intermediate nodes $\{v_3$,$v_4\}$ are computed using the equation 2.8. Note that the messages passed by the leaf nodes are used in computation at the intermediate nodes. Finally (right image) the message at the root node is computed using the equation 2.9. Note that each message computation has a complexity of $O(d^2)$ where d is the number of states of each random variable.

Note that the logarithm of equation 2.1 is equation 2.5.

As noted in the introduction, Pictorial structures is used to model the human body. For each body part (e.g, head, arms, torso and legs) a random variable X_i taking the part location l_i is assigned. Thus the configuration $L = (l_1, ..., l_n)$ forms the human pose. By solving equation 2.1, we can obtain the human pose in an image. Pictorial structures for the face is illustrated and explained in detail in figure 2.1.

2.1.2 Inference

As noted in the section 2.1.1, the pictorial structure model can be written as the following log-linear model of Markov Random Field,

$$L^* = \arg\min_{L} \left(\sum_{i=i}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right). \tag{2.6}$$

Given the data x, we would like to infer the configuration L^* with the maximum a-priori probability. In the following sections, the algorithms for the inference over an undirected graphs are presented. First the belief propagation is described which is essentially a dynamic programming algorithm and works only for acyclic graphs. For the cyclic graphs, loopy belief propagation algorithm is described.

The general acyclic graphs are trees. After choosing a particular node as root, the leaves of the tree are then ascertained. Belief propagation begins at the leaves and makes its way up to the root. This algorithm belongs to the class of message passing algorithms. At the leaf node, for each possible label

of the parent node the label with the least score is determined by using the following equation and its score is sent to its parent as the message.

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) \right). \tag{2.7}$$

For the intermediate nodes, a similar procedure is used with a slight modification in the equation. The score of its children are also added to its score. The equation for this computation is,

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{v_c \in C_j} B_c(l_j) \right). \tag{2.8}$$

Note that the score B_c has already been computed. Finally in the case of root node, which does not have any parent, its score is given by

$$B_r = \min_{l_r} \left(m_r(l_r) + \sum_{v_c \in C_r} B_c(l_j) \right). \tag{2.9}$$

The final configuration is noted by backtracking the above minimizations. The figure 2.2 illustrates the above procedure with an example and detailed explanation. The complexity of the algorithm is $O(nd^2)$ where n is the number of nodes and d is the number of labels at each node.

Marginals and Max-marginals: Given a distribution $P(x_1, ..., x_N)$, the marginal distribution over the variable $x_i = c$ is sum of all the probabilities over the sample space of all the variables except x_i. Thus the marginal $P(x_i = c)$ is given by,

$$P(x_i = c) = \sum_{x_1,...,x_i=c,...,x_N} P(x_1, ...x_N), \tag{2.10}$$

where each x_i denotes all of its sample space. The marginal distribution over the variable x_i is obtained by computing the above equation for each element in the sample space of x_i. A max-marginal has similar form except that the sum operator is replaced by minima operator and P is replaced by its negative log-linear. Both of these can be efficiently computed using the above belief propagation algorithm with minor changes. The max-marginal is in fact the solution of the equation 2.9. In that instance, it has been computed by making a particular node as the root node. By repeating this computation over all the nodes in the graph, we get the max-marginals for all the nodes in the graph. Marginals are obtained by replacing the minima operator in equations 2.7,2.8, 2.9 by sum operator, the sum of terms involving $m(l_i), d(l_i, l_j)$ by their product and the terms itself by their exponential of their negatives. In case of marginals though, care has to be taken about the partition function. The order complexity for computing both the marginal distributions and max-marginals over all the nodes is $O(N^2d^2)$.

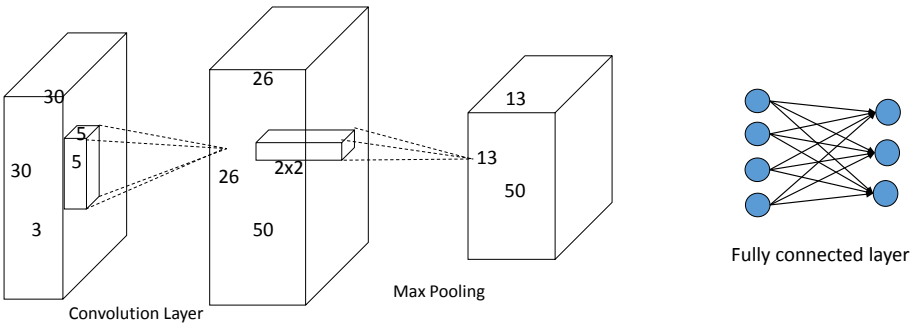


Figure 2.3: **Various layers in CNN:** The convolutional layer is taking an input of dimensions $30 \times 30 \times 3$, applying 50 filters each of dimension $5 \times 5 \times 3$ and to output an array of dimension $26 \times 26 \times 50$. Here, the number of parameters relative to the inputs and outputs are significantly small at 3750. On the convolutional layer's output, max pooling with filter size 2×2 is applied to obtain $13 \times 13 \times 50$ array. Finally in a fully connected layer, all the M input neurons and N output neurons are connected to each other requiring MN parameters.

2.2 Convolutional Neural Networks

In the recent past, convolutional neural networks (CNNs) [58, 23, 44] have been shown to outperform and significantly improve the object classification on the challenging ImageNet dataset [22]. Further this network, trained for object detection, has been shown [75] to improve the state-of-art on several other unrelated tasks like scene recognition, fine grained detection and so on. Since then CNNs have been used for solving practically all the computer vision problems.

The basic idea behind the CNNs is composition of patterns. An image can be viewed as a composition of higher level parts like faces, wheels and other distinct image patterns. These in turn are again composed of other intermediate visual patterns going up to lower level visual elements like edges and textures. Convolutional neural networks, first proposed by Lecun *et al.* [60], model an object as composition of patterns starting from edges to higher level parts like faces. CNNs as illustrated in the figure 2.3 can be visualized as a series of layers. The initial set of layers respond to discriminative low level patterns. The next set of layers respond to intermediate patterns which are composed of low level patterns and so on. The inspiration for CNNs and neural networks in general has been the biological understanding of the brain. It has been known for quite some time that the brain is made of over 100 billions neurons and these neurons are densely connected. The CNNs mimic neurons and their connections. A layer in CNN is made up of $m \times n$ neurons and neurons of the neighbouring layers are connected. In this section, we will describe about neurons, the connections and various types of layers that the modern CNNs have.

2.2.1 The model

Neurons: The basic computational unit in a CNN is a neuron. A neuron takes a linear combination of responses of the neurons from the previous layer as input and applies a non-linear function on it. In the past, $tanh$ and sigmoid functions were very popular for the non-linear function. But both of them suffer from saturation and vanishing gradients. The rectified linear unit or $ReLU$ unit [66] is the current choice for the non-linear function. The $ReLU$ and its variant leaky $ReLU$ are given by,

$$ReLU(x) = max(0, x) \tag{2.11}$$

$$LReLU(x) = max(0, x) + \delta(x < 0)kx. \tag{2.12}$$

Layers: A typical CNN has three types of layers: (a) convolutional layer, (b) pooling layer and (c) fully connected layer. A *convolutional layer* consists of K 3-D filters which are applied to the input to obtain K feature maps. These filters respond to specific visual patterns like edges and texture at lower level to faces and wheels at higher level. The output of the convolutional operations are K feature maps. At each location of the feature map, a nonlinear function called a neuron activation function is applied. The convolutional layers can be viewed as hidden layers which take localized inputs from the neurons of the previous layer. The convolutional layers are followed by pooling layers which pool the inputs in a local neighbourhood and typically down-sample the input, thus introducing translation invariance. Finally, fully connected layers take input from all the neurons of the previous layer and act as the reasoning units. Taking input from such a wide context helps in making better informed decisions about the class labels. The figure 2.3 illustrates the various layers in CNN.

2.2.2 Training

The CNN is trained using stochastic gradient descent. Depending on the task at hand, a loss function $L(X, W)$ is defined, where X is the input and W are network parameters to be learnt. The parameters are updated using the following rule:

$$W^t \leftarrow W^{t-1} - \eta\frac{\partial L}{\partial W} - \eta\beta W + \alpha W^{t-1}_{mom} \tag{2.13}$$

$$W^t_{mom} \leftarrow \alpha W^{t-1}_{mom} - \eta\frac{\partial L}{\partial W} - \eta\beta W \tag{2.14}$$

where W_{mom} is the momentum term, α is the momentum weight, η is the learning rate and β is the decay rate. The above equation is updating the weights by subtracting the gradient, the $L2$ regularizer or weight decay and adding the momentum term. The rates for these terms, which control the magnitude of these terms, have to be found using cross validation. The standard values are $\eta = 0.01$, $\beta = 0.0005$ and $\alpha = 0.9$. A more efficient version of the above weight update rule is the back propagation algorithm [78].

Here, we will briefly give some of the good practices for training CNNs. Stochastic gradient descent by design is a mini-batch update rule. Given N training samples, a small subset of M samples are used for one update. It is advisable to keep this number relatively small (typically 128). Thus a total of $\frac{N}{M}$ mini-batch updates are performed to cover the entire training data. One such scan through the data is

called as epoch and typically the network is trained over several epochs. To improve the generalization performance, the data is usually augmented by jittering the training samples through small affine transformations such as translation, rotation and scaling. The RGB data is itself changed by adding random but structured noise for samples per batch. To get the best result, the architecture of CNN is very important. Usually more number of convolutional layers help as does the number of filters. A very detailed empirical analysis can be found in the work by Chatfield *et al.* [17]. The drop-out regularization scheme introduced by Srivatsava *et al.* [48] has been shown to improve the generalization. The idea is to retain or discard the neurons with a probability of p. Thus at any given instance during training, a subset of neurons are active. Such a regularization breaks the neuron correlations and forces the network to learn more meaningful parameters. Monitoring the performance of the network on validation set is very important. The plot of loss function value on the validation data at regular intervals indicates the generalization performance. If the values of the loss function on the validation set is decreasing, then it indicates that the network is learning the right parameters. But when the values flattened or are showing signs of ascendency, the learning rate η has to be decreased at this point.

2.2.3 Feature Learning

Over the last few years, feature representations based on the CNN models have significantly outperformed traditional hand-tuned features. These features are learnt using the procedure described in the previous subsection. Given that several tasks have very limited amounts of data, first a CNN is learnt on a task such as image classification which has large amounts of data. This neural network is further trained on the task at hand to obtain representations which are particularly optimized for this task. While training on the particular vision task, typically only the last few fully connected layers are updated using stochastic gradient descent. These representations, the outputs of various layers (typically the last fully connected layer), are used for classification.

2.2.4 Siamese Networks

Siamese network, introduced in [15], is a pair of neural networks who, at any given time instance, have the same architecture and same weights. The network takes in a pair of images, forward propagates through the neural network pairs to obtain the projections and learns the weights based on a loss function on these projections. Typically the supervision given is whether the pair of images are similar or dissimilar. This method has been applied for signature verification [15], face verification [20], and image retrieval [100]. The work by Taylor *et al.* [89] using a loss function which takes a label in $[0, 1]$ interval. They have applied this method for character recognition and pose retrieval. Our work extends this method by taking image triplet and defining a ranking loss [54] on them. We further demonstrate the method on large dataset.

2.3 Human Body Detection

2.3.1 Poselets

Poselets [13] are part based models which was originally proposed for person detection. They are classifiers which model a subset of body parts. A poselet, for example, can model the head and the left shoulder together. The different poselet types are derived from data by randomly selecting a large number of potential candidates and then successively pruning them using various heuristics. Several such classifiers are trained with the objective of detecting a person. All these classifiers are then run on a test image. Based on the relative locations between the detections, the location of the person is estimated. In theory, such a method can be used for human pose estimation when the detected poselets model arms in a specific pose. This of course requires large amounts of data and hence not feasible. The poselets method has recently [14] been improved using CNNs. The key take-away of the poselets method is that a particular object can be detected using the parts that constitute it. These parts can be discovered from the training data automatically.

2.3.2 Upper Body Detection

Upper body detectors (UBD) are often used as a preprocessing stage in HPE pipelines [5, 26, 80]. Here, we use the publicly available detector of [32]. It combines an upper-body detector based on the part-based model of [35] and a face detector [98]. Both components are first run independently in a sliding window fashion followed by non-maximum suppression, and output two sets of *detection windows* (x, y, w, h). To avoid detecting the same person twice, each face detection is then regressed into the coordinate frame of the upper-body detector and suppressed if it overlaps substantially with any upper-body detection. As shown in [32] this combination yields a higher detection rate at the same false-positive rate, compared to using either component alone.

2.4 Human Pose Estimation

The human pose estimation problem can be defined as locating the different anatomical parts of the human body in a given image. This problem is of great consequence to understanding the action and even intent of the person. In the last few years many HPE algorithms have been proposed in the literature. Most of the HPE algorithms use pictorial structure model to describe a human. They differ in the way the definition of what constitute parts and their parameterization. The standard learning and inference algorithms listed in the previous sections are used in all the algorithms. In the following sections, some of the popular methods are described with details on parts and their parameterizations. Example human pose estimations are shown in figure 2.5.

(a) (b)

Figure 2.4: The images above illustrate the two ways in which parts and their relations have been defined. In (a), the whole anatomical body parts form parts in the pictorial structures. The body parts modelled are head, the two arms and the torso. The state space for each part would be the location and orientation. In (b), the body joints and few other points are modelled as parts in the pictorial structures. The state space for each part would be the location and orientation. In both figures, the rectangles represent the parts and the two neighboring rectangles connected by line segments have an edge in the pictorial structures.

2.4.1 Pictorial Structures based Methods

The four popular human pose estimation algorithms are described below. All of these methods use pictorial structures (section 2.1) as the model for human body. Broadly they differ in the definition of parts as well as definitions of unary potentials and pairwise potentials. Figure 2.4 shows the two different ways in which the human body is modelled using pictorial structures. For each of the algorithm described below, the model, unary and pairwise potentials will be clearly described.

Andriluka *et al.* [5]. This method uses the part definitions and their connections as displayed in figure 2.4a. It has a total of six parts and each part is parameterized by both location and orientation. This method solves the optimization problem described in equation 2.1.

The core of this technique lies in proposing discriminatively trained part detectors for the unary potential m_i. These are learned using Adaboost on shape context descriptors [8]. The authors model the kinematic relation between body parts d_{ij} using Gaussian distributions. For the best configuration of parts, they compute the marginal distributions for each part and take the location corresponding to the maximum in the marginal distribution. The marginal distribution is computed using a variant of the belief propagation (section 2.1.2).

Eichner and Ferrari [26]. This method too uses the part definitions as illustrated in figure 2.4a. It has a total of six parts and each part is parameterized by both location and orientation. This method solves the optimization problem described in equation 2.1.

In this method, first a set of pre-processing steps are applied. First a standard human detector is employed to detect the person. The bounding box is then expanded by a fixed ratio to include all the parts, no matter the pose. Then within this expanded bounding box, a fore-ground/back-ground

Figure 2.5: Pose estimations by (a) Andriluka *et al*. [5], (b) Eichner and Ferrari [26], (c) Sapp *et al*. [80] and (d) Yang and Ramanan [103]

segmentation algorithm [77] is used to detect the fore-ground. The algorithm is restricted to this area. Both these steps significantly reduce the search space of the algorithm in scale and locations.

After pre-processing the image, unary potential m_i is obtained at each position and orientation within the search area. Unary potential is a sum of two terms: i) a person independent edge model and ii) a person dependent color model. For the edge model, edge filters for each part are learnt using conditional likelihood approach. These filters are convolved over the search space to get the scores. Color models are challenging as the location of the parts are not known a priori to estimate the color models. Here two important observations help: a) certain body parts have rather stable location w.r.t. the detection window and b) often a person's body parts share similar appearance. First a segmentation prior is learned for each body part, which assigns a prior probability of part being located at pixel (x, y). Using this prior probability, a color model for part is learnt. It is often observed that the color model of face matches with that of lower arms or torso with that of upper arms. For each part i, its color model is learnt as a linear combination of color models of all other parts. These color models are used to obtain the second term for the unary potential.

For the pair-wise term d_{ij}, the following model is used,

$$d(l_i, l_j) = \alpha_i^T bin(l_i - l_j) \tag{2.15}$$

where $bin(.)$ is a vectorized count of spatial and angular histogram bins. Here α_i is a model parameter that favors certain (relative) spatial and angular bins for part i with respect to its parent. Using the above defined unary and pairwise potentials, the inference is done to obtain the pose estimate using belief propagation (section 2.1.2).

Sapp *et al.* [80]. This method too uses the part definitions as illustrated in figure 2.4a. It has a total of six parts and each part is parameterized by both location and orientation. This method solves the optimization problem described in equation 2.1.

The premise of this method is that the HPE algorithms do not use rich features for unary potentials and sophisticated pairwise potentials because the state space is so large that it is computationally very expensive to compute them. This work proposes to prune the state space by learning a cascade of models each trained to balance between, retaining the correct states and filtering as many states as possible.

Given an image and configuration pair (x, l), the score for the configuration $\theta_x(l)$ and max-marginal $\theta_x^*(l_i)$ are given by,

$$\theta_x(l) = \theta^T \phi(l, x) = \textstyle\sum_{ij} \theta_{ij}^T d_{ij}(l_i, l_j, x) + \sum_i \theta_i^T m_i(l_i, x) \tag{2.16}$$

$$\theta_x^*(l_i) = \max_{l' \in L} \{\theta_x(l') : l'_i = l_i\}, \tag{2.17}$$

where θ_{ij} and θ_i are the parameters to be learnt. Max-marginal is score of the best configuration when the part i is fixed to l'_i. The MAP score, $\theta_x^* = \max_{l \in L} \theta_x(l)$, is the score of best configuration with no constraints. Both these are obtained using belief propagation described in section 2.1.2.

The state space reduction is done using the max-marginals. The original state space for each part is $80 \times 80 \times 24$. This state space is sub-sampled to $10 \times 10 \times 12$ for the first stage of the cascade and max-marginals are computed for all the parts. All those states whose max-marginal is less than the threshold t_x given by,

$$t_x(\theta, \alpha) = \alpha \theta_x^* + (1 - \alpha) \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|L_i|} \sum_{l_i \in L_i} \theta_x^*(l_i), \tag{2.18}$$

are pruned. When $\alpha = 1$, only the state corresponding to MAP configurations is kept and when $\alpha = 0$, all the states whose score is more than the mean of the max-marginals remain. Those coarser states which remain are then up-sampled by a pre-determined factor and then passed to the next step of the cascade. Each model in the cascade is different and is learnt from the data. At the end of the cascade, about 500 states remain per part and computationally expensive features including those which measure contour completion and non-parametric pairwise potential are applied in this final round inference to obtain the final configuration.

The challenge is to learn threshold t_x which retains all the correct states and filters as many states as possible. It is to be noted that from the definition of t_x , it is clear that it is convex in the parameters.

The learning algorithm uses the fact that when $\theta_x(l) > t_x(\theta, \alpha)$ then $\forall i,\ \theta^*_x(l_i) > t_x(\theta, \alpha)$, so no part state of l is pruned. Hence it is sufficient that $\theta_x(l) > t_x(\theta, \alpha)$ to ensure that no correct part is pruned. Thus the learning task is formulated as

$$\min_{\theta} \frac{\lambda}{2}||\theta||^2 + \frac{1}{N}\sum_{n=1}^{N} H(\theta; x^n, l^n) \tag{2.19}$$

where H is a hinge upper bound and is given by $H(\theta; x, l) = \max\{0, 1 + t_x(\theta, \alpha) - \theta_x(l)\}$. The above optimization is solved using the stochastic sub-gradient descent algorithm.

Yang and Ramanan [103]. The previous approaches parameterized each part both by location and orientation. In order to realize the unary potential for a particular orientation, they had to rotate the part template and then compute the score. It has been observed in object detection [34] that parameterizing on just the location improves the performance and also makes the learning and inference simple and effective. Yet the orientation cannot be done away with as it is a major component in human articulation. Thus this method introduces a mixture of non-oriented affinely warped pictorial structures as illustrated in figure 2.4b. To handle the variations in articulation, each part is represented as a mixture model to handle rotations. The pictorial structures is then augmented by a co-occurrence constraint which favors a particular combination of part mixture components.

More precisely, this method takes the joints of the human body (e.g, elbow and shoulder) as the parts. Additionally it takes the midpoint of several joints too. Each part is modelled as a mixture model and the training data for each component of the mixture is obtained after clustering. Let each part be represented by i, its location by $p_i = (x, y)$ and the mixture type by t_i. Then $i \in \{1, ..., K\}$, $p_i \in \{1, ..., L\}$ and $t_i \in \{1, ..., T\}$. The compatibility across different parts with their associated types is given by,

$$S(t) = \sum_{i \in V} b_i^{t_i} + \sum_{ij \in E} b_{ij}^{t_i, t_j} \tag{2.20}$$

where $b_i^{t_i}$ is a bias for type given by t_i of part i and $b_{ij}^{t_i,t_j}$ is bias for two parts (i, j) with type (t_i, t_j) co-occurring together. Given the configuration (I, p, t) which specifies the image, the part locations and their types, the score is given by

$$S(I, p, t) = S(t) + \sum_{i \in V} w_i^{t_i}.m_i(I, p_i) + \sum_{ij \in E} w_{ij}^{t_i, t_j}.d_{ij}(p_i - p_j) \tag{2.21}$$

where $m_i(I, p_i)$ is the feature vector represented by HOG and $d_{ij}(p_i - p_j) = [dx\ dx^2\ dy\ dy^2]$ is relative location of part i relative to part j. The parameters (W, b) are learnt using a variant of the latent-SVM algorithm [103]. This algorithm searches over multiple scales, locations and all the part mixtures using the belief propagation algorithm described in section 2.1.2.

2.4.2 Pose Estimation using other Methods

Poselets [13]: Poselets are classifiers which model a subset of body parts. A poselet, for example, can model the head and the left shoulder together. The different poselet types are derived from data

by randomly selecting a large number of potential candidates and then successively pruning them using various heuristics. Several such classifiers are trained with the objective of detecting a person. In theory, such a method can be used for human pose estimation when the poselets model arms in a specific pose. This of course requires large amounts of data and hence not feasible. To alleviate this problem, Gkioxari *et al.* [45] propose to discover the poselets by using only the image patches corresponding to the arms. Thus each poselet, termed armlet in [45], models an arm in a particular pose. These classifiers are then run on the image to obtain detection bounding boxes. Using a pre-trained transfer model, the human joint locations are transferred to the armlet detections.

Convolutional neural network (CNN) based methods: Gkioxari et al [46] have proposed a naive approach for jointly learning human detection, pose estimation and action recognition using convolutional neural networks. They train a single neural network for all the above tasks and use different loss functions for each of the tasks. The pose estimation task is formulated as a regression task where the network takes an image as input and regresses to the human joint locations. The loss used is given by,

$$loss_P = \frac{1}{|K|}\sum_{k=1}^{|K|} v_k((\hat{x}_k - x_k)^2 + (\hat{y}_k - y_k)^2). \tag{2.22}$$

Toshev *et al.* [92] propose to learn a body pose regressor which maps an image to a $2K - D$ body pose vector, where K is the number of body joints. For this task, they use a massive network with five convolutional layers interleaved with pooling layers and three fully connected layers. The L2 loss, given in equation 2.22 between the predicted output and the ground truth is used for training the network. Clearly the pose vector obtained might not be accurate. To improve the precision of the detection, a cascade of individual body joint regressors are learnt where each stage takes the input from the previous stage and improves the precision of the detected point. The input to these regressors is the image patch centered at the detection from the previous stage. The dimensions are taken such that it has enough context from other body joints.

The work by Thompson *et al.* [90] report that the obvious methods such as the one described above, which maps the input image to the articulated pose, gave very poor results on standard datasets. Three reasons have been cited: (a) pooling operations destroy the precise spatial information necessary to accurately predict pose, (b) image to pose space mapping is highly non-linear and not one-to-one and (c) valid poses represent a much lower dimensional manifold in the possible set of poses potentially leading to incorrect poses. This method [90] instead proposes to learn individual body joint detectors. Thus for K body joints, K CNN models are learnt. A CNN classifier for detecting individual body joint is relatively much simpler task than learning a full body articulated pose with many degrees of freedom. This addresses the concern (b) which states that the image to pose space is highly non-linear. Further the method does inter body part spatial reasoning to filter out false positives from the individual pose detector outputs. This address the concern (c) by ensuring valid body poses. The details of the method [90] are given below.

For the part detectors, this method [90] employs a multi-resolution sliding window based convolutional neural network which takes as input an image patch at higher resolution and image patch with

surrounding context at lower resolution. It then forward propagates these patches through the network to obtain a heat map indicating the location of the body part. To improve the efficiency of sliding window detector, the fully connected layers are replaced by the equivalent convolutional layers. Since the body joints are independently trained, the output does not follow any kinematic constraints among various body joints. An example of kinematic constraints is that the location of face and shoulder has to be at a appropriate distance. To enforce such constraint this approach integrates MRF into CNN framework. The topology of the MRF graph consists of body joints being the vertices with random variables on the location and with edges between all the pairs. This model then approximates the marginal distribution given by,

$$\bar{p}_A = \frac{1}{Z} \prod_{v \in V} (p_{A|v} * p_v + b_{b->A}). \tag{2.23}$$

This update of probability is akin to a single round of loopy belief propagation. The final marginal distribution $\bar{p}_A$ is output of the algorithm. This method boasts the state-of-art on the two important datasets for human pose estimation.

2.5 Pose Retrieval

Pose retrieval is an image and video retrieval task where the query is a human pose (spatial configuration of human body parts) and the retrieved list is a set of video clips in which the people-of-interest are in the same pose as the query. This query modality gives an ability to do a content based image retrieval on video collections with humans present in it. Potential applications where this system can be used are dance retrieval, where the query could be a dance posture (*e.g.,* various postures in Bharatanatyam), and sports retrieval, where the query could be a famous shot in a game (*e.g.,* cover drive in Cricket and backhand in Tennis).

Pose retrieval system, like other retrieval systems, can be understood in terms of two phases, (a) Offline and (b) Online. The *Offline* phase is primarily about processing the videos and indexing them. It consists of three steps: (a) Pre-processing videos, (b) Computing a pose descriptor for each video frame, and (c) Indexing these descriptors. The *Online* phase takes the query as an input, computes the descriptor and then retrieves video shots using the indexing structure built during the *Offline* phase.

Ferrari *et al.*, [40] first introduced the problem of pose retrieval. In their work, first the video is processed using a person detector. Then a human pose estimation algorithm [26] is run on these person detections to obtain pose estimates. A by-product of HPE algorithms is marginal distributions for each part. It gives a probability distribution over part locations. It assigns a probability that part i is located at a location (x, y, θ). The following three descriptors have been proposed to efficiently encode this marginal probability.

Descriptor A: part positions. The marginal distribution is simply downscaled to make it more compact and robust to small shifts and intra-class variations. The initial distribution of dimensions $141 \times 159 \times 24$ is downsized to $20 \times 16 \times 8$ making it a 15360 dimensional vector.

Descriptor B: part orientations, relative locations and relative orientations. This descriptor encodes the absolute orientations of the individual body parts and relative locations and orientations between a pair of body parts.

The probability $P(l_i^o = \theta)$ that part l_i has orientation θ is obtained by marginalizing out location

$$P(l_i^o = \theta) = \sum_{(x,y)} P(l_i = (x, y, \theta)) \tag{2.24}$$

The probability $P(r(l_i^o, l_j^o) = \rho)$ that the relative orientation $r(l_i^o, l_j^o)$ from part l_i to l_j is ρ is

$$P(r(l_i^o, l_j^o) = \rho) = \sum_{(\theta_i,\theta_j)} P(l_i^o = \theta_i).P(l_j^o = \theta_j).\delta(r(\theta_i, \theta_j) - \rho) \tag{2.25}$$

where $r(\cdot, \cdot)$ is a circular difference operator and δ is the dirac-delta function. The dimensions of this descriptor is 1449. The probability $P(l_i^{xy} - l_j^{xy} = \delta)$ of relative location $\delta = (\delta_x, \delta_y)$ is has a similar definition to relative orientation given in the above equation.

Descriptor C: part soft-segmentations. The third descriptor is based on the soft segmentation. For each body part l_i, a segmentation mask is computed at each pixel which indicates if the body part is present or not. The dimension of this feature vector is 1920.

The above descriptions are stored for each frame in the video. Given a query pose representation, it is compared against all the frames in the database using the Bhattacharya distance and best shots in a video are retrieved. Here, a shot is simply a contiguous set of frames with minor incremental changes. The score of the shot is set to the best scoring frame.

2.6 Summary

In this chapter, we reviewed all the important material that is necessary to understand the coming chapters. We described in detail those material which we explicitly use in our work. We also indicated the chapters in which a particular material is used. Specifically, we reviewed the state-of-art human pose estimation algorithms [5, 26, 45, 46, 74, 80, 90, 92, 103] and human pose search algorithms [40, 51]. We also described pictorial structures [34], an important model in computer vision. Any entity which is composed of visually different parts can be modelled using pictorial structures. Prominently human upper body is modelled using pictorial structures. The pictorial structures model [34], learning algorithms [94, 36] and inference algorithms [34] are described in detail. We also described poselets [13] which is used in this thesis. We also described the convolutional neural networks [60] and Siamese networks [89], a variant of it. In the next chapter, the datasets and evaluation metrics used in this work are described.

Chapter 3

Datasets and Evaluation Measures

In this chapter, we describe the datasets that are used throughout the work for training, validation and testing. We describe the "Movie Stickmen dataset" that we built for the work in the succeeding chapters. We also describe other publicly available datasets that we use in our work. For understanding the performances of various tasks in our work such as pose retrieval, pose comparison and reliability of pose estimation we need evaluation measures. In this chapter, we describe the evaluation measures and pose comparison measures.

3.1 Datasets

For training the pose representation modules, a large amount of high quality annotations are required. In this section, we describe the datasets that we built and datasets publicly released by others.

Figure 3.1: **Annotated ground-truth samples (stickman overlay)** from the Movie Stickmen dataset (all except last row, last column) and Buffy-2 dataset (last row and last column).

Dataset	Movie [50]	Buffy2 [50]	Total
# Images	5984	499	6483
# Annotations	11639	775	12414
# Annotations with no occlusions	3764	399	4163

Table 3.1: **Our dataset statistics.** The number of images and stickmen annotations for the two newly introduced datasets *viz.,* Movie stickman dataset and Buffy2 dataset are given above. While the total number of stickman annotations are 12414, only 4163 of them have all the parts visible.

Dataset	#Images	#Annotations	#Annotations with no occlusion
H3D [13]	381	1002	238
ETH pascal [27]	549	549	549
Buffy stickmen [41]	748	748	748
FLIC [79]	5003	5003	5001
MPII human pose [4]	17408	28883	7729
Poses in the wild [19]	830	830	717
We are family [29]	525	3131	1290
Synchronic Activities [30]	357	1128	1112
Total	25801	41274	17384

Table 3.2: **Off-the-shelf dataset statistics.** Samples from publicly available datasets are added to the newly introduced datasets to increase the dataset size. The number of annotations do not include the flipped versions.

3.1.1 Our Datasets

We introduce two new datasets, called *Buffy2 Stickmen* [50] and *Movie Stickmen* [50]. These datasets are challenging as they show people at a range of scales, wearing a variety of clothes, in diverse poses, lighting conditions and scene backgrounds. Buffy2 Stickmen is composed of frames sampled from episodes 2 and 3 of season 5 of the TV show *Buffy the vampire slayer* (note this dataset is distinct from the existing Buffy Stickmen dataset [39]). This dataset is created to both supplement the poses which are fewer in number and also to add new poses to Buffy Stickmen dataset [39]. Movie Stickmen contains frames sampled from ten Hollywood movies (*About a Boy, Apollo 13, Four Weddings and a Funeral, Forrest Gump, Notting Hill, Witness, Gandhi, Love Actually, The graduate, Groundhog day*).

The data is annotated with upper body stickmen (6 parts: head, torso, upper and lower arms) using the following criteria: all humans who have more than three parts visible are annotated, provided their size is at least 40 pixels in width (so that there is sufficient resolution to see the parts clearly). For near frontal (or rear) poses, six parts may be visible; and for side views only four. See the examples in figures 3.1,3.3. This fairly complete annotations includes self occlusions and occlusions by other objects and other people. Table 3.1 gives the number of images and annotated stickmen in each dataset.

3.1.2 Off-the-shelf Datasets

We use the following publicly available pose stickmen datasets in our works: Buffy stickmen dataset [41], ETH PASCAL dataset [27], the H3D dataset [13], FLIC dataset [79], MPII Human pose dataset [4], Poses in the wild dataset [19], We are family dataset [29] and Synchronic Activities dataset [30]. Each of these datasets contains images and stick figure annotations of the humans. For the convenience of pose search method, we consider only those annotations in which all parts are visible. For a partially occluded person, defining a positive instance for retrieval is ambiguous. In all, there are $17,384$ fully visible annotations. The statistics are given in the Table 3.2. To further enhance the dataset size, each image and annotation is horizontally flipped effectively doubling the corpus to $34,768$ stickmen. Using the stickman annotations, the bounding box of the upper body is constructed and transformed into the expanded bounding box. To understand the efficacy of various pose representation schemes, the ground truth bounding box is assumed.

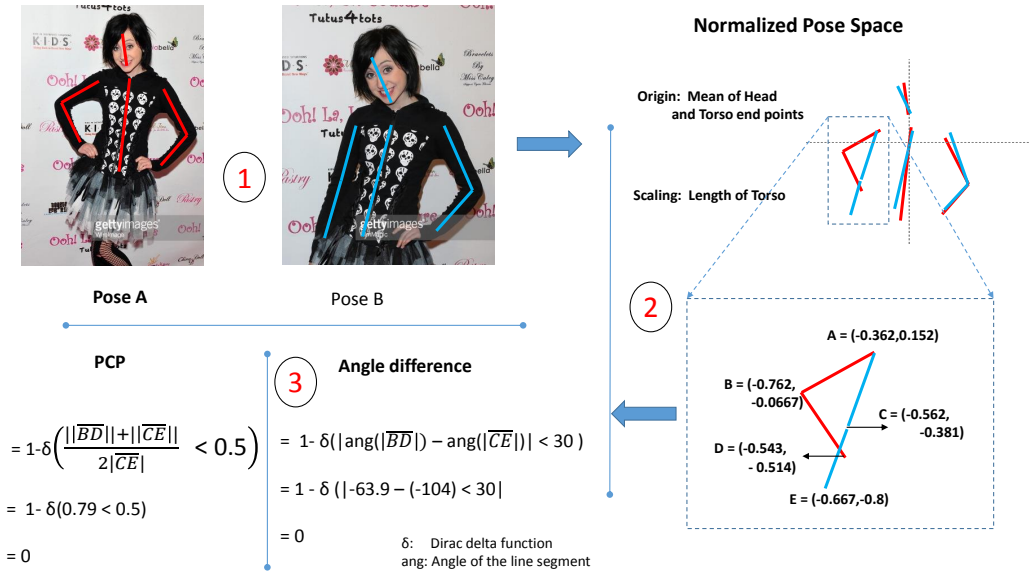


Figure 3.2: **Pose similarity measures:** The PCP and angle difference measures are computed on pair of example poses here. Given a pair of poses (1), they are transformed into a normalized pose space (2). The origin for this normalized pose space is the mean point of the head and torso line segments. The scaling factor is the length of the torso. In some implementations, the mean lengths of all the body parts are also taken. Finally the PCP and angle difference measures (3) are calculated. The exact formulas are give in the above image. In this image only the calculation for the lower left arm is depicted. (Best viewed in color)

3.2 Evaluation Measures

In this section, the following evaluation metrics are described in detail: (i) Average precision and (ii) Area under the ROC curve. Average precision is a standard retrieval measure and we use it for measuring the performance of pose retrieval system. It has also been used to measure the performance of detection tasks [33]. "Area under the ROC curve" is an evaluation measure used for the classification tasks.

3.2.1 Average Precision

Given a set of samples, where each sample being the tuple (c_i, g_i) where $c_i \in R$ is the confidence score and $g_i \in 0, 1$ is the ground truth, the average precision is computed in the following way. First all the tuples are ordered based on the confidence score (traditionally highest to the lowest). At each sample s_i in this ordered list, precision p_i and recall r_i is computed on the first i samples. These series of precision and recall values are plotted with recall as X-axis and precision as Y-axis to obtain precision-recall (PR) curve. The area under the PR curve is the average precision (AP). The higher the value of AP, the better the algorithm's performance. The average precision is high when the positive samples ($g_i = 1$) have very high confidence. This measure is natural for the ranking tasks where the positive sample is one which is either similar to query or belongs to the same category as the query. This measure is also very useful in measuring the performance of detection tasks where the algorithm is supposed to score the positive samples higher than negative samples. The PR curve does not have any properties such as monotonicity and can be quite chaotic.

3.2.2 Area under the ROC Curve

Given a set of samples, where each sample being the tuple (c_i, g_i) where $c_i \in R$ is the confidence score and $g_i \in \{0, 1\}$ is the ground truth, the area under the ROC curve is computed in the following way. First, all the tuples are ordered based on the confidence score (traditionally highest to the lowest). At each sample s_i in this ordered list, true positive rate (TPR) t_i and false positive rate (FPR) f_i are computed on the first i samples. These series of TPR and FPR values are plotted with FPR as X-axis and TPR as Y-axis to obtain receiver operator characteristic (ROC) curve. The area under the ROC curve is the measure AUC. The higher the value of AUC, the better the algorithm's performance. The AUC is high when the positive samples ($g_i = 1$) have very high confidence. This measure is very useful in measuring the performance of classification tasks where the algorithm is supposed to score the positive samples higher than negative samples. The ROC curve is monotonically non-decreasing function.

3.3 Pose Similarity and Evaluation measures

In both pose estimation task and pose retrieval task, measuring the similarity and the degree of it or dissimilarity of two poses is critical. Below, two pose similarity measures are described. When comparing two poses which belong to two people from very different frames, both the poses have to be brought to the same frame of reference. Below, the pose normalization scheme is described. The pose normalization method and the two pose similarity measures are illustrated in the figure 3.2.

3.3.1 Normalized Pose Space

Given a pose in terms of stickmen co-ordinates, it is transformed both by translation and scaling. First, the co-ordinates are translated with the origin as the mid point of the left and right shoulder points. Next, the co-ordinates are scaled by the length of the torso obtaining the normalized pose. Two such normalized poses can now be compared using any pose similarity measure.

3.3.2 Percentage of Correct Parts (PCP)

The PCP measure proposed by Eichner *et al.* [32] gives a discrete measure of how similar two human pose stickmen are. The first stickmen is denoted as *reference pose* while the second pose is denoted as *sample pose*. PCP computes the number of corresponding body part line segments which are in agreement. Two corresponding body part line segments are said to be in agreement if the end points of the sample pose segment are within a fraction of the length of the reference pose segment. The typical threshold used is 0.5. Yang *et al.* [104] point out two important difficulties with this measure. Firstly, PCP is sensitive to the amount of foreshortening of a limb. Secondly, PCP needs correspondence between candidate poses and ground truth poses. Eichner *et al.* [32] prescribe matching all the candidate poses with all the ground truth poses. While scoring, unmatched ground-truth poses (false negatives) are penalized but unmatched candidates (false positives) are not. As a consequence, algorithms that predict a large number of candidate poses unfairly benifit from this measure.

Yang *et al.* [104], instead propose a new metric termed percentage of correct keypoints (PCK). A candidate keypoint (e.g., a body joint) is considered as correct if it falls within $\alpha.\max(h, w)$ pixels of the ground-truth keypoint, where h and w are the height and width of the tight bounding box containing the person. For each body part, two measures are considered: (a) accuracy and (b) the average precision. The value of α is determined empirically.

3.3.3 Angle Difference

Given the two poses, the sample is classified as similar, dissimilar or ignored using body part angle (angle made by a body part with the image axis). The samples belonging to ignore class are neither considered while training nor while testing. The classification is done using the following procedure: (a) the sample pose is considered as similar if its individual part angles do not deviate by more than

τ_1 from the reference pose, (b) the sample pose is considered as dissimilar if its individual part angles deviate by more than τ_2 degrees from the reference pose, and (c) Finally the sample pose is in ignore class if its individual part angles deviate by less than τ_2 degrees but with at-least one part which deviates between τ_1 and τ_2 degrees. Using cross validation, the thresholds τ_1 and τ_2 are set at 20 and 30 degrees respectively.

3.4 Summary

In this chapter, we described the various datasets used through out the work. In particular we described the "Movie stickmen dataset" and "Buffy2 stickmen dataset" that we built. We also described the performance measures for the pose retrieval, pose comparison and reliability of pose estimation tasks. In the next chapter, we will describe the pose descriptor based on the output of the human pose estimation. This descriptor is used as the representation for each sample while performing the pose retrieval. There the problem of unreliability of the human pose estimation is explained and our novel concept of "Automatic human pose evaluator" is described.

Figure 3.3: Ground-truth samples (stickman overlay) from all the movies present in Movie Stickmen dataset (first row) and Buffy-2 dataset (second row).

Chapter 4

Pose Representation using Human Pose Estimation Algorithms

4.1 Introduction

In this chapter, we describe our first pose representation. This pose representation is derived from the output of human pose estimation algorithms (HPE). The HPE task is to predict the 2D (image) stickman layout of the person: head, torso, upper and lower arms. While HPE algorithms [26, 80, 5, 103] have a reasonable success, it suffers from unreliability. With good frequency, a few body parts of the person (typically lower arms) are incorrectly estimated. This clearly effects the human pose retrieval task. In this chapter, we address the issue of reliability and describe the procedure to obtain the compact $12D$ pose representation from the output of the human pose estimation algorithm. This work [50] is published in ECCV 2012.

Computer vision algorithms for recognition are getting progressively more complex as they build on earlier work including feature detectors, feature descriptors and classifiers. A typical object detector, for example, may involve multiple features and multiple stages of classifiers [33]. However, apart from the desired output (e.g. a detection window), at the end of this advanced multi-stage system the sole indication of how well the algorithm has performed is typically just the score of the final classifier [35, 98]. In this chapter, we describe how to redress this balance and obtain pose representations. We argue that algorithms *should* and *can* self-evaluate (illustrated in figure 4.1). They should self-evaluate because this is a necessary requirement for any practical systems to be reliable. That they can self-evaluate is demonstrated in this work for the case of human pose estimation (HPE). Such HPE evaluators can then take their place in the standard armoury of many applications, for example removing incorrect pose estimates in video surveillance, pose based image retrieval, or action recognition.

In general four elements are needed to learn an evaluator: a ground truth annotated database that can be used to assess the algorithm's output; a quality measure comparing the output to the ground truth; auxiliary features for measuring the output; and a classifier that is learnt as the evaluator. After learning, the evaluator can be used to predict if the algorithm has succeeded on new test data, for which ground-truth is not available. We discuss each of these elements in turn in the context of HPE.

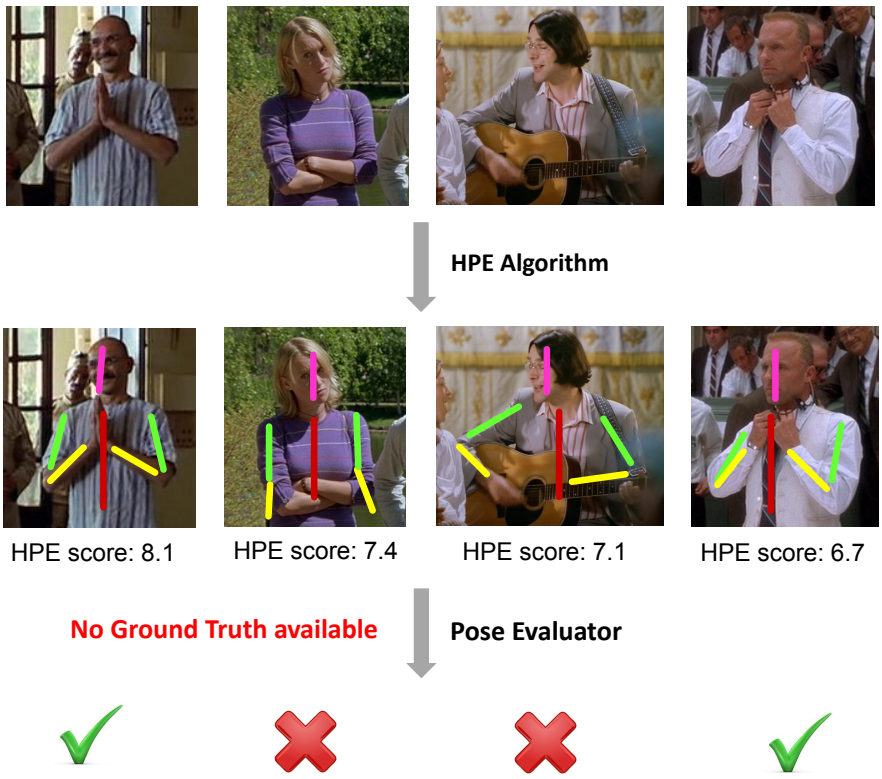


Figure 4.1: **Pose evaluator illustration.** The HPE algorithm is run on images in the top row to output the pose estimates and a confidence score (row two). It incorrectly assigns low confidence score to a correct pose estimate (Ed Harris's image). Pose evaluator accurately distinguishes (row three) the correct pose estimates from the incorrect ones.

The ground truth annotation in the dataset must provide the positions of various body parts. The pose quality is measured by the difference between the predicted layout and ground truth – for example the difference in their angles or joint positions. The auxiliary features can be of two types: those that are used or computed by the HPE algorithm, for example max-marginals of limb positions; and those that have not been considered by the algorithm, such as proximity to the image boundary (the boundary is often responsible for erroneous estimations). The estimate given by HPE on each instance from the dataset is then classified, using a threshold on the pose quality measure, to determine positive and negative outputs (e.g. based on the number of correctly estimated limbs). Given these positive and negative training examples and both types of auxiliary features, the evaluator is learnt using standard methods (here an SVM).

We apply this evaluator learning framework to four recent publicly available methods: Eichner and Ferrari [26], Sapp *et al.* [80], Andriluka *et al.* [5] and Yang and Ramanan [103]. The auxiliary features, pose quality measure, and learning method are described in section 4.2. For the datasets, section 4.3, we use existing ground truth annotated datasets, such as *ETHZ PASCAL Stickmen* [26] and *Humans in*

3D [13], and supplement these with additional annotation where necessary. We assess the evaluator features and method on the four HPE algorithms, and demonstrate experimentally that the proposed evaluator can reliably predict when the algorithms succeed. In this chapter, we extend our preliminary work [50] on this subject with new ways in which pose evaluator can be used to filter out incorrect poses produced by a pose estimator and to combine the outputs of multiple different estimators (section 4.4).

Note how the task of a HPE evaluator is not the same as that of deciding whether the underlying human detection is correct or not. It might well be that a correct detection then leads to an incorrect pose estimate. Moreover, the evaluator cannot be used directly as a pose estimator either – a pose estimator predicts a pose from an enormous space of possible structured outputs. The evaluator's job of deciding whether a pose is correct is different, and easier, than that of producing a correct pose estimate.

On the theme of evaluating vision algorithms, the most related work to ours is Mac Aodha *et al.* [64] where the goal is to choose which optical flow algorithm to apply to a given video sequence. They cast the problem as a multi-way classification. In visual biometrics, there has also been extensive work on *assessor* algorithms which predict an algorithm's failure [81, 99, 1]. These assess face recognition algorithms by analyzing the similarity scores between a test sample and all training images. The method by [99] also takes advantage of the similarity within template images. But none of these explicitly consider other factors like imaging conditions of the test query (as we do). [1] on the other hand, only takes the imaging conditions into account. Our method is designed for another task, HPE, and considers several indicators all at the same time, such as the marginal distribution over the possible pose configurations for an image, imaging conditions, and the spatial arrangement of multiple detection windows in the same image. Other methods [101, 12] predict the performance by statistical analysis of the training and test data. However, such methods cannot be used to predict the performance on individual samples, which is the goal of this work.

4.2 Pose Evaluation Method

We formulate the problem of evaluating the human pose estimates as classification into 'success' and 'failure' classes. First, we describe the novel features we use (section 4.2.1). We then explain how human pose estimates are evaluated. For this, we introduce a measure of quality for HPE by comparing it to a ground-truth stickman (section 4.2.2). The features and quality measures are then used to train the evaluator as a classifier (section 4.2.3)

4.2.1 Features

We propose a set of features to capture the conditions under which an HPE algorithm is likely to make mistakes. We identify two types of features: (i) based on the output of the HPE algorithm – the score of the HPE algorithm, marginal distribution, and best configuration of parts L^*; and, (ii) based

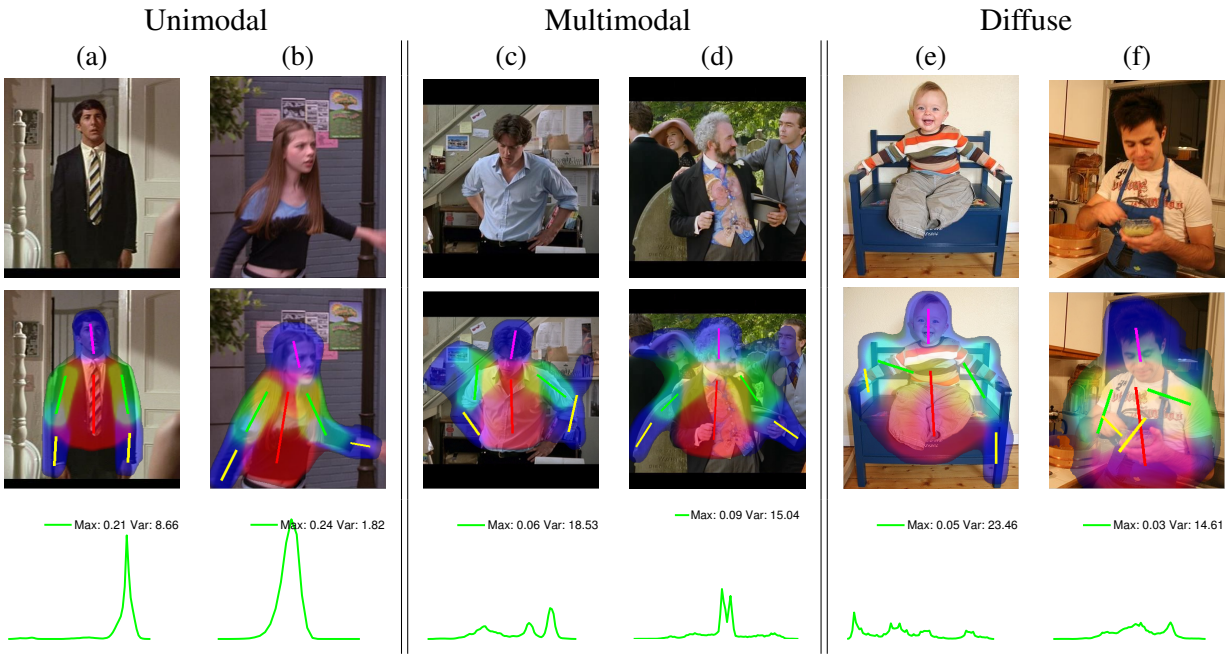


Figure 4.2: **Features based on the output of HPE.** Examples of unimodal, multi-modal and large spread pose estimates. Each image (first row) is overlaid with the best configuration (sticks) and the posterior marginal distribution over the body part position in a semi-transparent mask (displayed in second row). 'Max' and 'Var' features are measured from this distribution (third row). In the case of unimodal distributions, as the above examples (a and b) indicate, the mode almost always corresponds to the correct part location. While in the case of multimodal distributions, typically one of the modes correspond to the correct part location as in example (c), but again they may not as in example (d). Finally in the case of diffuse distributions (e and f), the many modes that barely stand out do not convey any information about correctness of part locations. Thus unimodal distributions are good indicators of correct part configurations as opposed to the other two types. The type of distribution is determined using the 'Max' and 'Var' features. As the distribution moves from peaked unimodal to more multi-modal and diffuse, the maximum value decreases and the variance increases.

on the extended detection window – its position, overlap with other detections, etc – these are features which have not been used directly by the HPE algorithm. We describe both types of features next.

1. Features from the output of the HPE. The outputs of the HPE algorithm consist of a marginal probability distribution P_i over (x, y, θ) for each body part i, and the best configuration of parts L^*. The features computed from these outputs measure the spread and multi-modality of the marginal distribution of each part. As can be seen from figures 4.2 and 4.3, the multi-modality and spread correlate well with the error in the pose estimate. Features are computed for each body part in two stages: first, the marginal distribution is pose-normalized, then the spread of the distribution is measured by comparing it to an ideal signal.

The best configuration L^*, predicts an orientation θ for each part. This orientation is used to pose-normalize the marginal distribution (which is originally axis-aligned) by rotating the distribution so

that the predicted orientation corresponds to the x-axis (illustrated in figure 4.4). The pose-normalized marginal distribution is then factored into three separate x, y and θ spaces by projecting the distribution onto the respective axes. Empirically we have found that this step improves the discriminability.

A descriptor is then computed for each of the separate distributions which measure its spread. For this we appeal to the idea of a matched filter, and compare the distribution to an ideal unimodal one, P^*, which models the marginal distribution of a perfect pose estimate. The distribution P^* is assumed to be Gaussian and its variance is estimated from training samples with near perfect pose estimate. The unimodal distribution shown in figure 4.2 is an example corresponding to a near perfect pose estimate.

The actual feature is obtained by convolving the ideal distribution, P^*, with the measured distribution (after the normalization step above), and recording the maximum value and variance of the convolution. Thus for each part we have six features, two for each dimension, resulting in a total of 36 features for an upper-body. The entropy and variance of the distribution P_i, and the score of the HPE algorithm are also used as features taking the final total to $36 + 13$ features.

Algorithm specific details: While the procedure to compute the feature vector is essentially the same for all four HPE algorithms, the exact details vary slightly. For Andriluka *et al*. [5] and Eichner and Ferrari [26], we use the posterior marginal distribution to compute the features. While for Sapp *et al*. [80] and Yang and Ramanan [103], we use the max-marginals. Further, in [103] the pose-normalization is omitted as max-marginal distributions are over the mid and end points of the limb rather than over the limb itself. For [103], we compute the max-marginals ourselves as they are not available directly from the implementation.

2. Features from the detection window. We now detail the 10 features computed over the extended detection window. These consist of the scale of the extended detection window, and the confidence score of the detection window as returned by the upper body detector. The remainder are:

- *Two image features:* the mean image intensity and mean gradient strength over the extended detection window. These are aiming at capturing the lighting conditions and the amount of background clutter. Typically HPE algorithms fail when either of them has a very large or a very small value.

- *Four location features:* these are the fraction of the area outside each of the four image borders. Algorithms also tend to fail when people are very small in the image, which is captured by the scale of the extended detection window.

  The location features are based on the idea that the larger the portion of the extended detection window which lies outside the image, the less likely it is that HPE algorithms will succeed (as this indicates that some body parts are not visible in the image).

- *Two overlap features:* (i) the maximum overlap, and (ii) the sum of overlaps, over all the neighbouring detection windows normalized by the area of the current detection window. As illustrated in Figure 4.5 HPE algorithms can be affected by the neighbouring people. The overlap features capture the extent of occlusion by the neighbouring detection windows. Overlap with other people

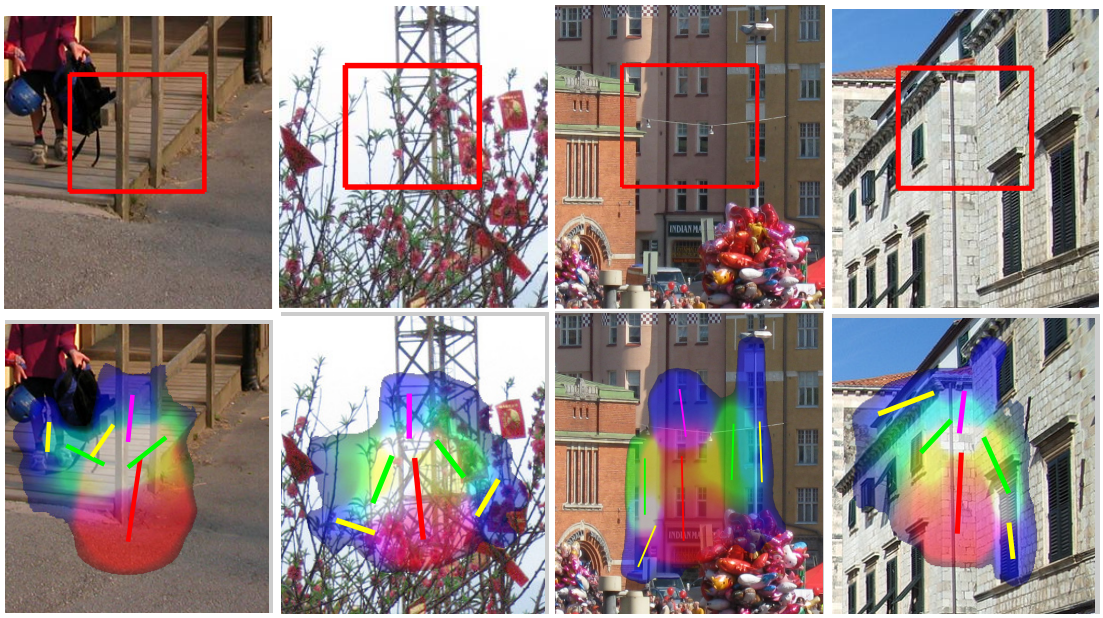


Figure 4.3: **Human pose estimates on incorrect upper-body detections**: *typical examples of human pose estimates in a false positive detection window. The pose estimates (MAP) are unrealistic (shown as a stickman, i.e. a line segment per body part) and the posterior marginals (PM) are very diffuse (shown as a semi-transparent overlay: torso in red, upper arms in blue, lower arms and head in green). The pose estimation outputs are from Eichner and Ferrari [26].*

indicates how close the they are. While large overlaps occlude many parts in the upper body, small overlaps also affect HPE performance as the algorithm could pick the parts (especially arms) from their neighbours.

While measuring the overlap, we consider only those neighbouring detections which have similar scale (between 0.75 and 1.25 times). Other neighbouring detections which are at a different scale typically do not affect the pose estimation algorithm.

4.2.2 Pose Quality Measure

For evaluating the quality of a pose estimate, we devise a measure which we term the *Continuous Pose Cumulative error* (CPC) for measuring the dissimilarity between two poses. It ranges in $[0, 1]$, with 0 indicating a perfect pose match. In brief, CPC depends on the sum of normalized distances between the corresponding end points of the parts. Figure 4.6 gives examples of actual CPC values of poses as they move away from a reference pose. The CPC measure is similar to the *percentage of correctly estimated body parts* (PCP) measure of [26]. However, CPC adds all distances between parts, whereas PCP counts the number of parts whose distance is below a threshold. Thus PCP takes integer values in $\{0, \ldots, 6\}$. In contrast, for proper learning in our application we require a continuous measure, hence the need for the new CPC measure.

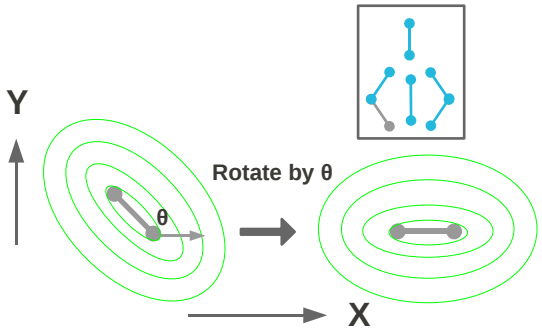


Figure 4.4: **Pose normalizing the marginal distribution.** *Marginal distribution (contour diagram in green) on the left is* pose normalized *by rotating it by an angle θ of the body part (line segment in grey) to obtain the transformed distribution on the right. In the inset, the pertinent body part (line segment in grey) is displayed as a constituent of the upper body.*

In detail, the CPC measure computes the dissimilarity between two poses as the sum of the differences in the position and orientation over all parts. Each pose is described by N parts and each part p in a pose A is represented by a line segment going from point $\mathbf{s}_p^a$ to point $\mathbf{e}_p^a$, and similarly for pose B. All the coordinates are normalized with respect to the detection window in which the pose is estimated. The angle subtended by p is θ_p^a. With these definitions, the $CPC(A, B)$ between two poses A and B is:

$$\begin{aligned} CPC(A,B) &= \sigma\left(\sum_{p=1}^{N} w_p \frac{\left\|\mathbf{s}_p^a - \mathbf{s}_p^b\right\| + \left\|\mathbf{e}_p^a - \mathbf{e}_p^b\right\|}{2\left\|\mathbf{s}_p^a - \mathbf{e}_p^a\right\|}\right) \\ \text{with } w_p &= 1 + \sin\left(\frac{|\theta_p^a - \theta_p^b|}{2}\right) \end{aligned} \tag{4.1}$$

where σ is the sigmoid function and $\theta_p^a - \theta_p^b$ is the relative angle between part p in pose A and B, and lies in $[-\pi, \pi]$. The weight w_p is a penalty for two corresponding parts of A and B not being in the same direction. Figure 4.7 depicts the notation in the above equation.

4.2.3 Learning the Pose Quality Evaluator

We require positive and negative pose examples in order to train the evaluator for an HPE algorithm. Here a positive is where the HPE has succeeded and a negative where it has not. Given a training dataset of images annotated with stickmen indicating the ground truth of each pose, (section 4.3.1), the positive and negative examples are obtained by comparing the output of the HPE algorithms to the ground truth using CPC and thresholding its value. Estimates with low CPC (i.e. estimates close to the true pose) are the positives, and those above threshold are the negatives.

In detail, the UBD algorithm is applied to all training images, and the four HPE algorithms [5, 26, 80, 103] are applied to each detection window to obtain a pose estimate. The quality of the pose estimate

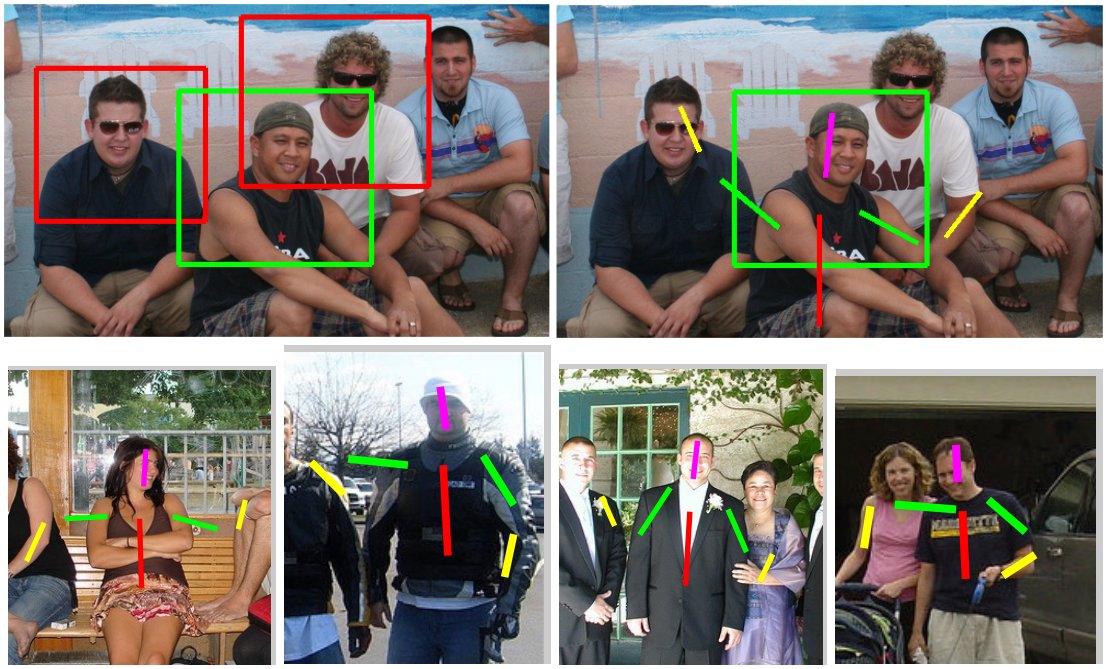


Figure 4.5: **Overlap features.** *First row, Left: Three overlapping detection windows. First row, Right: the HPE output is incorrect due to interference from the neighbouring people. This problem can be flagged by the detection window overlap (output from Eichner and Ferrari [26]). Second row: Few more examples of incorrect HPE outputs due to interference from the neighbouring people.*

is then measured by comparing it to ground truth using CPC. Analog to the use of PCP [26], CPC is only computed for correctly localized detections (those with IoU > 0.5). Detections not associated with any ground-truth are discarded. Since all of the HPE algorithms considered here cannot estimate partial occlusion of limbs, CPC is set to 0.5 if the ground-truth stickman has occluded parts. In effect, the people who are partially visible in the image get high CPC. This is a means of providing training data to the evaluator showing the HPE algorithms will fail on these cases.

A CPC threshold of 0.3 is used to separate all poses into a positive set (deemed correct) and a negative set (deemed incorrect). This threshold is chosen because pose estimates with CPC below 0.3 are nearly perfect and roughly correspond to PCP = 6 with a threshold of 0.5 (figure 4.6). A linear SVM is then trained on the positive and negative sets using the auxiliary features described in section 4.2.1. The feature vector has 59 dimensions, and is a concatenation of the two types of features (49 components based on the output of the HPE, and 10 based on the extended detection window). This classifier is the evaluator, and will be used to predict the quality of pose estimates on novel test images. The performance of the evaluator algorithm is discussed in the experiments of section 4.3.3.

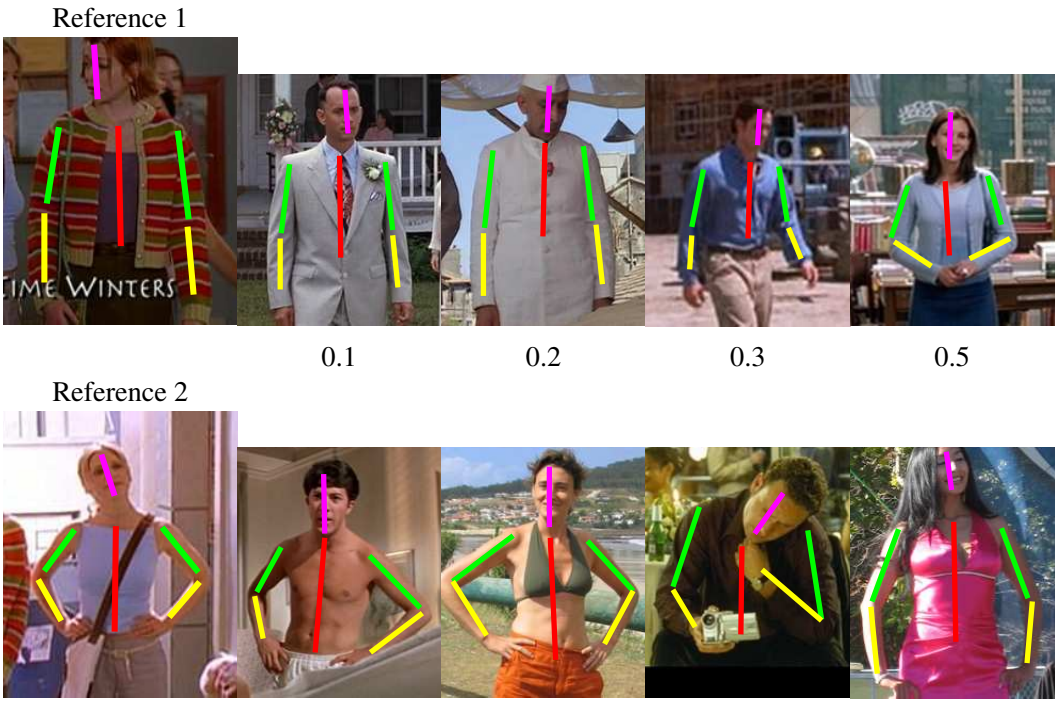


Figure 4.6: **Poses with increasing CPC.** *An example per CPC is shown for each of the two reference poses for CPC measures of 0.1, 0.2, 0.3 and 0.5. As can be seen example poses move smoothly away from the reference pose with increasing CPC, with the number of parts which differ and the extent increasing. For 0.1 there is almost no difference between the examples and the reference pose. At 0.2, the examples and reference can differ slightly in the angle of one or two limbs, but from 0.3 on there can be substantial differences with poses differing entirely by 0.5.*

4.3 Pose Evaluator Performance

After describing the datasets we experiment on (section 4.3.1), we present a quantitative analysis of the pose quality evaluator (section 4.3.3).

4.3.1 Data

We experiment on four of the datasets introduced in chapter 3: *ETHZ PASCAL Stickmen* [26], *Humans in 3D* [13], *Buffy2 Stickmen* and *Movie Stickmen*. All four datasets are challenging as they show people at a range of scales, wearing a variety of clothes, in diverse poses, lighting conditions and scene backgrounds. Table 4.1 gives the number of images and annotated stickmen in each dataset. As a *training set* for our pose quality evaluator, we take Buffy2 Stickmen, Humans in 3D and five movies (*About a Boy, Apollo 13, Four Weddings and a Funeral, Forrest Gump, Notting Hill*) of Movie Stickmen. ETHZ PASCAL Stickmen and five movies (*Witness, Gandhi, Love Actually, The graduate, Groundhog day*) of

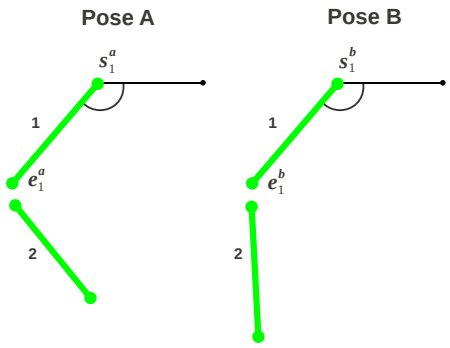


Figure 4.7: **Pose notation:** *For poses A and B corresponding parts p are illustrated. Each part is described by starting point* $\mathbf{s}_p^a$*, end point* $\mathbf{e}_p^a$ *and the angle of the part* θ_p^a *are also illustrated.*

Dataset	#images	#ground truth-train	#ground truth-test
ETHZ Pascal [27]	549	0	549
Humans in 3D [13]	429	1002	0
Movie [50]	5984	5804	5835
Buffy2 [50]	499	775	0
Total	7270	7581	6834

Table 4.1: **Train-test split.** *For our experiments, two publicly available datasets and two newly introduced datasets are used. The train-test stickmen annotations along with other statistics are given above.*

Movie Stickmen form the *test set*, on which we report the performance of the pose quality evaluator in the next subsection.

4.3.2 Performance of the HPE Algorithms.

For reference, table 4.2 gives the average PCP across different datasets. Table 4.3 gives the percentage of samples where pose is estimated accurately, i.e. the CPC between the estimated and ground-truth stickmen is < 0.3 In all cases, we use the implementations provided by the authors [26, 80, 5, 103] and all methods are given the same detection windows [32] as preprocessing. Both measures agree on the relative ranking of the methods: Yang and Ramanan [103] performs best, followed by Sapp *et al*. [80], Eichner and Ferrari [26] and then by Andriluka *et al*. [5]. This confirms experiments reported independently in previous works [26, 80, 103]. Note that we report these results only as a reference, as the absolute performance of the HPE algorithms is not important in this work. What matters is how well our newly proposed evaluator can predict whether an HPE algorithm has succeeded.

Datasets	Andriluka [5]	Eichner [26]	Sapp [80]	Yang [103]
Buffy stickmen	78.3	81.6 (83.3)	84.2	86.7
ETHZ Pascal	65.9	68.5	71.4 (78.2)	72.4
Humans in 3D	70.3	71.3	75.3	77.8
Movie	64.6	70.4	70.6	76.0
Buffy2	65.0	67.5	74.2	81.7

Table 4.2: **Pose estimation evaluation (PCP).** *The PCP* [1] *of the four HPE algorithms (Andriluka* et al. *[5], Eichner and Ferrari [26], Sapp* et al. *[80], Yang and Ramanan [103]) averaged over each dataset (at PCP threshold 0.5, section 4.2.2). The numbers in brackets are the results reported in the original publications. The differences in the PCPs are due to different versions of UBD software used. The performances across the datasets indicate their relative difficulty.*

Dataset	Andriluka [5]	Eichner [26]	Sapp [80]	Yang [103]
Train	8.5	10.7	11.6	18.5
Test	9.9	11.1	12.0	16.5
Total	9.1	10.9	11.8	17.6

Table 4.3: **Pose estimation evaluation (CPC).** *The table shows the percentage of samples which were estimated accurately (CPC < 0.3) on the training and test sets, as well as overall, for the four HPE algorithms (Andriluka* et al. *[5], Eichner and Ferrari [26], Sapp* et al. *[80], Yang and Ramanan [103]). These accurate pose estimates form the positive samples for training and testing the evaluator.*

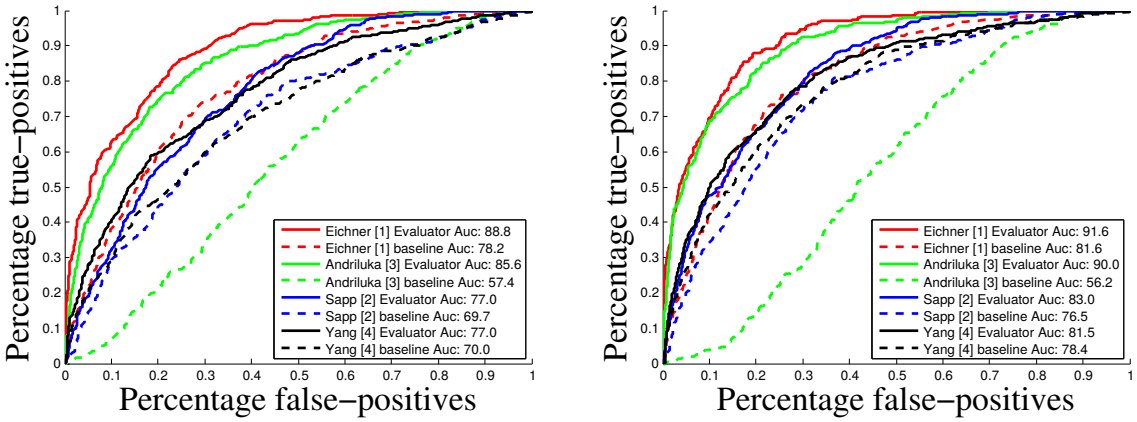
4.3.3 Assessment of the Pose Quality Evaluator

Here we evaluate the performance of the pose quality evaluator for the four HPE algorithms. To assess the evaluator, we use the following definition: A pose estimate is defined as *positive* if it is within CPC 0.3 of the ground truth and as *negative* otherwise. The evaluator's output (*positive* or *negative* pose estimate) is defined as *successful* if it correctly predicts a *positive* (true positive) or *negative* (true negative) pose estimate, and defined as a *failure* when it incorrectly predicts a *positive* (false positive) or *negative* (false negative) pose estimates. Using these definitions, we assess the performance of the evaluator by plotting an ROC curve.

The performance is evaluated under two regimes: (A) only where the predicted HPE corresponds to one of the annotations. Since the images are fairly completely annotated, any upper body detection window [32] which does not correspond to an annotation is considered a false positive. In this regime such false positives are ignored; (B) all predictions are evaluated, including false-positives. The first regime corresponds to: given there is a human in the image at this location, how well can the proposed method evaluate the pose estimate? The second regime corresponds to: given there are wrong person detections, how well can the proposed method evaluate the pose estimate? The protocol for assigning a HPE prediction to a ground truth annotation was described in section 4.2.3. For regime B, any pose on a false-positive detection is assigned a CPC of 1.0.

	CPC 0.2		CPC 0.3		CPC 0.4	
HPE	BL	PA	BL	PA	BL	PA
Andriluka [5]	56.7	90.0	56.2	90.0	55.8	89.2
Eichner [26]	84.3	92.6	81.6	91.6	80.5	90.9
Sapp [80]	76.5	82.5	76.5	83.0	76.9	83.5
Yang [103]	79.5	83.7	78.4	81.5	78.4	81.2

Table 4.4: **Performance of the Pose Evaluator in Regime B.** *The pose evaluator is used to assess the outputs of four HPE algorithms (Andriluka* et al. *[5], Eichner and Ferrari [26], Sapp* et al. *[80], Yang and Ramanan [103]) at three different CPC thresholds. The evaluation criteria is the area under the ROC curve (AUC).* BL *is the AUC of the baseline and* PA *is the AUC of our pose evaluator.*



(a) Regime A

(b) Regime B

Figure 4.8: **(a) Performance of HPE evaluator in regime A:** *(no false positives used in training or testing). The ROC curve shows that the evaluator can successfully predict whether an estimated pose has CPC < 0.3.* **(b) Performance of HPE evaluator in Regime B:** *(false positives included in training and testing).*

Figures 4.8a and 4.8b show performance for regime A and regime B respectively. The ROC curves are plotted using the score of the evaluator, and the summary measure is the Area Under the Curve (AUC). The evaluator is compared to a relevant baseline that uses the HPE score as a confidence measure (i.e. the energy of the most probable (MAP) configuration L^*). For Andriluka *et al*. [5], the baseline is the sum over all parts of the maximum value of the marginal distribution for a part. The plots demonstrate that the evaluator works well, and outperforms the baseline for all the HPE methods. Since all the HPE algorithms use the same upper body detections, their performance can be compared fairly.

To test the sensitivity to the 0.3 CPC threshold, we learn a pose evaluator also using CPC thresholds 0.2 and 0.4 under the regime B. Table 4.4 shows the performance of the pose evaluator for different CPC thresholds over all the HPE algorithms. Again, our pose evaluator shows significant improvements over the baseline in all cases. The improvement of AUC for Andriluka *et al.* [5] is over 33.5. We

Figure 4.9: **Example evaluations.** *The pose estimates in the first two rows are correctly classified as successes by our pose evaluator. The last two rows are correctly classified as failures. The pose evaluator is learnt using the regime B and with a CPC threshold of* 0.3*. Poses in rows 1,3 are estimated by Eichner and Ferrari [26], and poses in rows 2,4 are estimated by Yang and Ramanan [103].*

believe that this massive increase is due to the suboptimal inference method used for computing the best configuration. For Eichner and Ferrari [26], Sapp [80], and Yang and Ramanan [103] our pose evaluator brings an average increase of 9.6, 6.4 and 3.4 respectively across the CPC thresholds. Interestingly, our pose evaluator has a similar performance across different CPC thresholds.

Our pose evaluator successfully detects cases where the HPE algorithms succeed or fail, as shown in figure 4.9. In general, an HPE algorithm fails where there is self-occlusion or occlusion from other objects, when the person is very close to the image borders or at an extreme scale, or when they appear in very cluttered surroundings. Analyzing the learnt weight vector from the linear SVM, we see that the HPE evaluator too has identified these as the failure cases (sec. 4.2.3). The magnitude of the components of the weight vector suggests that the features based on marginal probability and the location of the detection window are very important in distinguishing the positive samples from the negatives. On the other hand, the weights of the upper body detector, image intensity and image gradient are low and thus do not have much impact.

4.4 Applying Pose Evaluator

In many applications, we are interested only in fully correct pose estimates, i.e. with all parts correctly estimated. For instance, a pose estimate with some incorrect parts can severely lower the perceived performance of a retrieval system, if it appears high up in the ranked list. Here we use the assessment criterion of [26] and define a part as correctly estimated if its segment end-points lie within 50% of the length of the ground-truth annotation.

In the following experiments we only consider two HPE algorithms [26, 103] as their computational speed is high, but it can easily be extended to other algorithms [5, 80] as well. Unfortunately, for both HPE algorithms the percentage of fully correct pose estimates is very low (10.9% and 17.6% from table 4.3). Note that we are applying a very severe test for a correct pose – that *all* parts are correct. In contrast, the quantitative measure used to assess performance in [26, 103] counts the *average* number of correctly estimated parts (PCP). For example, a pose with 5 out of 6 parts correctly estimated has a PCP score of $5/6 = 0.83$, but scores zero under our criterion.

In order to improve the percentage of correct pose estimates in a given set of estimates, we employ the pose evaluator in three different ways. Before applying these methods, all the images are first processed with the upper body detector [32]. Then, on each upper body detection the two pose estimation algorithms Eichner and Ferrari [26], Yang and Ramanan [103] are run. The confidence scores given by these algorithms are normalized to $[0, 1]$ range using Platt [73]. The parameters for the normalization are learned separately for each HPE algorithm.

4.4.1 Methods

For different problems, the number of algorithms with good performance may vary. To address this variation the following three methods have been developed which work with one or multiple algorithms. 'Filter I' works with one algorithm, while 'Filter II' and 'Hybrid pose estimation' methods work with two or more algorithms.

Filter I: *Removing samples with low score.* In this method, we use the pose evaluator to remove incorrect pose estimates. By construction, the pose evaluator tends to give a high confidence score to correctly estimated poses and a low score to incorrect ones. Hence, we use the pose evaluator to give a confidence score to each pose estimate in the input set. We can then remove pose estimates by thresholding this confidence score. Below we use the confidence score in other ways as well.

Filter II: *Agreement of pose estimates.* In this method, we consider the agreement between poses produced by two different HPE algorithms on the same human detection. The intuition is, while the incorrect pose estimates of the algorithms can be very different, the correct pose estimates would be the same. Since the algorithms are based on different features and inference algorithms, they are rather complementary in their failure modes. Hence we expect that our agreement criterion should reliably identify correct pose estimates.

We consider two poses to be in agreement if the PCP between them is perfect (i.e. 1.0). Suppose the algorithms of Eichner and Ferrari [26] and Yang and Ramanan [103] generate pose estimates $A1$ and $A2$ respectively, then $A1$ and $A2$ are in agreement if the PCP between them is perfect (PCP=1.0). If this is the case, then the pair $(A1, A2)$ is added to the agreement set. The new confidence score of the pair $(A1, A2)$ is the linear combination of both the confidence scores given by the pose evaluator.

Hybrid pose estimation: In this method, we make another use of outputs from multiple HPE algorithms on each detected person. The key observation is that often different algorithms fail on different images. Hence, chances are higher that at least one algorithm out of several has correctly estimated the pose, than when relying on a single algorithm. We use our pose evaluator to identify the correct output from among these multiple pose estimates. This is done by taking the highest scoring pose estimates from all the pose estimates belonging to a person.

Note that after processing the input set using the above three methods, we can either threshold the pose estimates or rank them using the new confidence scores assigned by the respective methods.

4.4.2 Results

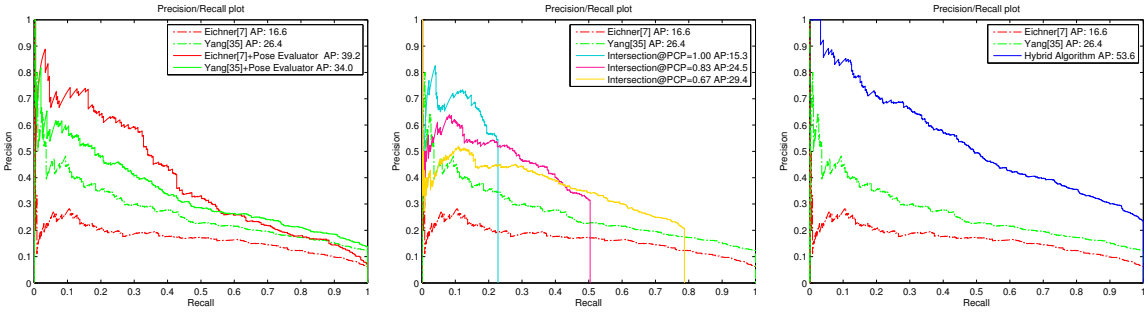
We evaluate the three methods proposed in the previous subsection individually and then compare their relative performances. The three methods are evaluated on the same test set (section 4.3.1) used to test the pose evaluator.

The evaluation measure used is average precision, which is the area under the precision recall plot. First the pose estimates, denoted by S, are sorted based on the confidence score assigned by a method described in previous subsection. At each possible recall point, precision is computed and precision-recall curve is plotted. Precision is defined as the total number of positives in S divided by the cardinality of S. Recall is defined as the total number of positives in S divided by the total number of ground truth stickmen.

Filter I: This method is evaluated by measuring average precision over the test set, on a range of thresholds, by removing the pose estimates with pose evaluator scores less than threshold. Figure 4.10a shows that the average precision improved significantly from, 16.6% (before filtering) to 39.2% (after filtering) for [26] and from 26.4% (before filtering) to 34.0% (after filtering) for [103].

Filter II: This method is evaluated by first performing the procedure described in section 4.4.1 to obtain an agreement set S. An element $e = (A1, A2)$ in S is deemed to be positive if both $A1$ and $A2$, pose estimates from [26, 103] respectively, are in full agreement (PCP=1.0) with the ground-truth stickman. The confidence score , as mentioned in the previous section, is the linear combination of baseline scores $baseline_{EF}, baseline_{YR}$ of both the HPE algorithms [26, 103] respectively. Empirically we have found that the linear combination of $0 * baseline_{EF} + 1 * baseline_{YR}$ gives the best results.

Figure 4.10b shows that the agreement operation, with legend 'Intersection@PCP=1.00', improves precision significantly. It outperforms both baseline [26, 103] at most recall points. Since this method rejects a considerable number of pose estimates, the curves end before reaching recall 1.0. This method is expected to have high precision and low recall as it aggressively filters out incorrect pose estimates and



(a) Filter I (b) Filter II (c) Hybrid pose estimation

Figure 4.10: **Improving average precision using the pose evaluator.** *The figure shows the precision and recall of HPE on the test set described in section 4.3.1 . We see that all the methods have a significant positive affect on average precision (AP). Pose evaluator has successfully filtered incorrect pose estimates as shown in 4.10a and fused the pose estimates from Eichner and Ferrari [26], Yang and Ramanan et al [103] as shown in 4.10c.*

in the process loses some correct pose estimates too. We also tried weaker criteria requiring agreement in at least five parts (PCP ≥ 0.83) or at least four parts (PCP ≥ 0.67). As figure 4.10b suggests, these weaker criteria improve the best attainable recall while worsening precision significantly.

Hybrid pose estimation: Finally, this method is evaluated by combining the pose estimates of [26] and [103] as described in section 4.4.1. Using the new confidence score assigned by this method, average precision is computed. Plot 4.10c shows that the average precision improved greatly from, 16.6% [26] and 26.4% [103] to 53.6%.

Comparing the three methods we can see from figure 4.10 that the 'Filter II' method has high precision and low recall, 'Filter I' method has good precision and tolerable recall and 'Hybrid pose estimation' method has good precision and high recall (relative to the HPE algorithms). Hence, when only one HPE algorithm is available, the 'Filter I' method is a better option than using the HPE output 'as is'. When multiple HPE algorithms are available, the 'Hybrid pose estimation' algorithm is the best option, as it outperforms the 'Filter II' method.

4.5 Pose Representation

From the human pose estimation algorithm, we obtain pose estimates consisting of line segments corresponding to the upper body parts namely head, torso, right and left arms. These depend on the size of the person and their location. To compare two pose estimates we require a representation that is invariant to scale and location in the image. Previous representations used for retrieval purposes were high dimensional, for example the three descriptors proposed by [40] have 15360, 1449 and 1920 dimensions. These descriptors are very high dimensional because they represent either full probability distributions over possible part positions and orientations, or soft-segmentations of the parts.

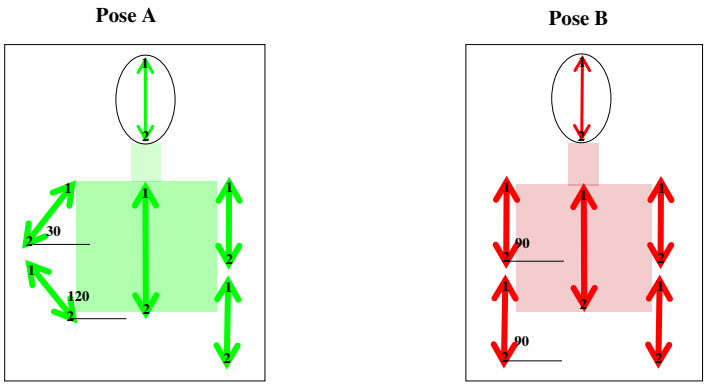


Figure 4.11: **Pose Representation:** *The two poses A and B differ in two parts, the upper and lower left arms. The pose representation based on the angles clearly distinguishes pose A from pose B. The endpoints of each stickmen are consistently numbered.*

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

When comparing two poses, we would like to form a distance measure based on the sum of differences of angles of corresponding parts. To achieve this we encode the angle θ of each part as the 2D vector $(\cos\theta, \sin\theta)$. For six parts this results in a 12 dimensional feature vector. Then the Euclidean distance between two such representations gives the cosine of the angular difference. This is elaborated in more detail below. Figure 4.11 illustrates the pose representation.

4.6 Summary

In this chapter we described a pose representation based on the output of human pose estimation algorithms. We highlighted the fact that human pose estimation algorithms are very unreliable. To improve the reliability, we have introduced the concept of an evaluator algorithm which we developed for human pose estimation methods. Our evaluator accurately predicts if the vision algorithm has succeeded or not when usually for such algorithms no confidence score (only a MAP score) is provided. We have also shown that the evaluator can be used to effectively filter incorrect pose estimates, fuse outputs from different pose estimation algorithms and improve the quality of pose representations derived from pose estimation algorithms.

More generally, we have cast self-evaluation as a binary classification problem, using a threshold on the quality evaluator output to determine successes and failures of the HPE algorithm. An alternative

approach would be to learn an evaluator by regressing to the quality measure (CPC) of the pose estimate. We could also improve the learning framework using a non-linear SVM.

We believe that our evaluator has wide applicability. It works for any part-based model with minimal adaptation, no matter what the parts and their state space are. We have shown this in the work, by applying our method to various pose estimators [5, 26, 80, 103] with different parts and state spaces. A similar methodology to the one given here could be used to engineer evaluator algorithms for other human pose estimation methods e.g. using poselets [13], and also for other visual tasks such as object detection (where success can be measured by an overlap score).

In the next chapter, we describe a prototypical large scale human pose retrieval system using the pose representation developed in this chapter. All the aspects pertaining to a retrieval system such as query modality, user interface and nearest neighbor search in high dimensional space are explored in detail.

Chapter 5

Human Pose Retrieval System

5.1 Introduction

In this chapter, we describe the real time retrieval of human poses from a large collection of videos. [1] The last decade has seen considerable progress in 2D human pose (layout) estimation on images taken in uncontrolled and challenging environments. There are now several algorithms available [5, 27, 80, 103] that can detect humans and estimate their poses in typical movie frames, where humans can appear at any location, at any scale and wear clothing of varying sizes, colors and texture; the illumination can vary and the backgrounds can be cluttered.

There are numerous reasons why detecting humans and obtaining their pose is useful. A fundamental one is that often the pose, or a pose sequence, characterizes a person's attitude or action. More generally, applications range from video understanding and search for suspicious activity detection in surveillance videos to search for particular sport shots or dance poses. In this work, we enable, for the first time, the real time retrieval of poses in a scalable manner. Being able to retrieve video material by pose provides another access mechanism to video content, complementing previous work on searching for shots containing a particular object or location [87], person [6, 61, 84], or action [11, 59].

The real time system we propose is applicable on top of most 2D pose estimation algorithms. The only condition is that their output can be mapped into the simple pose representation, which enables rapid Euclidean distance based nearest-neighbor matching. We demonstrate our system here on the output of two existing pose estimation algorithms, Eichner & Ferrari [27] and Yang & Ramanan [103]. The functionality of the system is illustrated in figure 5.1, which shows the three query modalities that we have developed and the type and quality of the output.

While an early system for pose search was first proposed by [40], in this chapter we extend the idea and methods substantially. First, we introduce a low dimensional pose representation and approximate nearest neighbor matching, making our system much more efficient both in terms of computational cost and memory consumption. This enables real-time search in a large scale database (over 3 million frames). Second, our system has a better retrieval engine: by combining independent pose estimation

[1] The live demo of the pose retrieval system is at the following location: http://zeus.robots.ox.ac.uk/posesearch/index.html

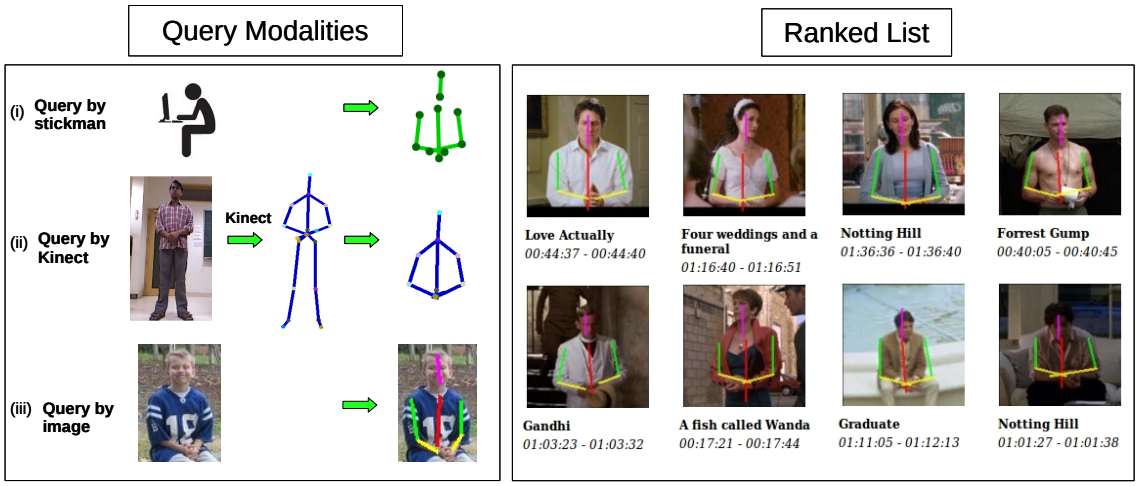


Figure 5.1: **Pose retrieval overview.** An illustration of the three query modalities: interactive stickman GUI, pose from Kinect, and pose estimated from an image. The output ranked list is obtained instantaneously as the query is varied. This can be tailored in various ways, for example to select only poses in different shots or within a particular movie. The results of the query pose are shown here at the video level.

algorithms, we can reject frames where the pose estimate is likely to be incorrect. This leads to higher precision in the top returns (but not higher recall). Third, our approach allows for new query modes: query-by-stickman (with an interactive GUI), and query-by-Kinect, in addition to the query-by-image of [40].

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

5.2 System Overview

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

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

5.2.1 Query modes

For a user to query the system, we have developed three querying modes (figure 5.1). (i) The first is a so-called *stickman*. The interface is intuitive where the user can move the joints of a iconic stick figure freely and the pose retrieval system responds to these movements in real-time by retrieving the matching frames from the database. Imagine that the user is moving the stickman from the famous 'Titanic' pose (arms stretched side ways and parallel to the ground) to a 'Hands up' pose, passing through a series of intermediate poses. Then the pose retrieval system follows this sequence of poses, and continuously updates the results. This mode is very useful as a web-based application and can be deployed anywhere with a basic computer and an Internet connection. (ii) In the second mode, the user himself can move by waving his/her arms around and making different poses. The pose retrieval system tracks his movements using the Kinect sensor and displays the retrieved results in real-time. This mode is suitable for entertainment applications. (iii) Finally in the third mode, the user can upload an image of a person and then request the pose retrieval system to display similar poses in the database. For mobile application, this mode is very useful.

5.2.2 Output Presentation

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

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

5.2.3 Off-line Processing Stages

Video processing: Frames are grouped into shots, and an upper body detector is run on all the frames to detect people. The upper body detections are then grouped through the temporal sequence into a track. Finally two human pose estimation algorithms [27, 103] are run on each upper body detection in all the tracks, to estimate a stickman pose (section 5.3).

Improving pose estimation: Unfortunately the precision of these pose estimation algorithms is not high and this significantly affects the perceived quality of the pose retrieval system. To improve precision, we propose to use the human pose evaluator which takes into account the human pose estimate and auxiliary information to decide if the pose estimate is correct or not. Several ways of using the pose evaluator are described in section 4.4. Poses which survive this filtering step are then considered for retrieval.

Pose representation and matching: For representing the pose we propose to use the the low dimensional 12D vector pose representation described in section 4.5. The pose representation is crucial to the performance of the retrieval system as it simultaneously affects both the accuracy and speed of the system. The 12D vectors are recorded in a forest of randomized K-D trees for fast approximate nearest neighbor retrieval (section 5.5).

5.2.4 On-line Processing Stages

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

Client-server architecture: For the retrieval system we use a standard client-server architecture (section 5.6). While the client interfaces with the user, the server performs the back-end operations of fast search and ranking, stores the randomized K-D tree structure, and serves the thumbnails, frames and videos.

5.3 Video Processing

The videos are processed off-line over a series of three stages: (i) shot detection using a standard color histogram method [62]; (ii) upper-body detection to localize the people in each video frame and track them within the shot; (iii) human pose estimation (HPE) algorithms to determine the human pose within the upper-body detection area. In the current system we have 18 videos, 17 are Hollywood movies and one is a season of the TV serial 'Buffy the Vampire Slayer'. In all, there are about 3 million frames. The statistics are given in table 5.1.

5.3.1 Upper Body Detection and Tracking

An upper body detector (UBD) is run on every frame of the video. An UBD algorithm detects and gives a bounding box around the people in the image. These are often a prerequisite for human pose estimation algorithms [5, 27, 80].

Here, we use the publicly available detector of [28]. It combines an upper-body detector based on the model of Felzenszwalb *et al.* [34] and the OpenCV face detector [98]. Both detectors run in a sliding window fashion followed by non-maximum suppression, and output a detection window (x, y, w, h). To avoid detecting the same person twice, each face detection is then regressed into the coordinate frame of the upper-body detector and suppressed if it overlaps substantially with any upper-body detection. As shown in [31] this combination yields a higher detection rate at the same false-positive rate, compared to using either detector alone.

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

Table 5.1: Dataset statistics. We consider 18 movies for the retrieval system. For each video, the number of images, shots, tracks and human pose estimates are reported. Movies with * are abbreviations for 'Buffy the Vampire Slayer' , 'Four weddings and a Funeral', 'Living in oblivion', 'Lost in Translation', 'My Cousin Vinny', 'Desperately Seeking Susan' and 'A fish called Wanda'.

In order to reduce the false positives, the detections in a shot are grouped into tracks and short tracks are discarded. These tracks are obtained using the temporal association strategy described in [40]. Visually, the tracks form a continuous stream of a person's bounding box in the shot.

5.3.2 Human Pose Estimation and Representation

We use here the algorithms of Eichner and Ferrari [27] and Yang and Ramanan [103]. Both of these have publicly available implementations. The algorithm [27] is run on each upper body detection in a track. The upper body detection is used to determine the scale of the person. The algorithm [103] is run over the whole frame to obtain multiple pose estimates. As many of the pose-estimates are false-positives, we use the upper body detections to filter them out. In detail, all detections returned by [103] which overlap less than 50% with any UBD detection are discarded. Overlap is measured using the standard "intersection over union" criterion [33]. Each of the pose estimate is then passed through pose evaluator described in section 4.4. Poses which survive this filtering step are then considered for retrieval.

For representing the pose, the $12D$ vector pose representation described in section 4.5 is used. The representation is constructed by taking the $sine$ and $cosine$ of the six body parts obtaining $12D$ vector.

5.4 Pose Matching

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

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

5.5 Pose Retrieval

After the database has been pre-processed off-line with the stages detailed above, it is ready to be searched by the user. The user enters a query (section 5.2.1) which is converted to the same 12-D representation as the poses in the database (section 4.5).

For searching the query in the database we use the approximate nearest neighbor (ANN) method proposed by Silpa-Anan *et al.* [83]. The method has one of the best recall rates among algorithms that index high dimensional data with significant search speed gain over exhaustive search [65]. The method organizes the database as a collection of randomized K-D trees. To construct a K-D tree, the data is split at the median value of a pre-assigned dimension d and the two halves are passed down to the left and right subtrees. This procedure is recursively followed to further split the data. In [83] the splitting dimension is randomly chosen among the T dimensions with the largest variances. The technique constructs a set of randomized trees by running the random selection multiple times.

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

Implementation details We discuss the important parameters of the number of trees used and the number of nearest neighbors returned from each below in the evaluation. To construct a forest of two

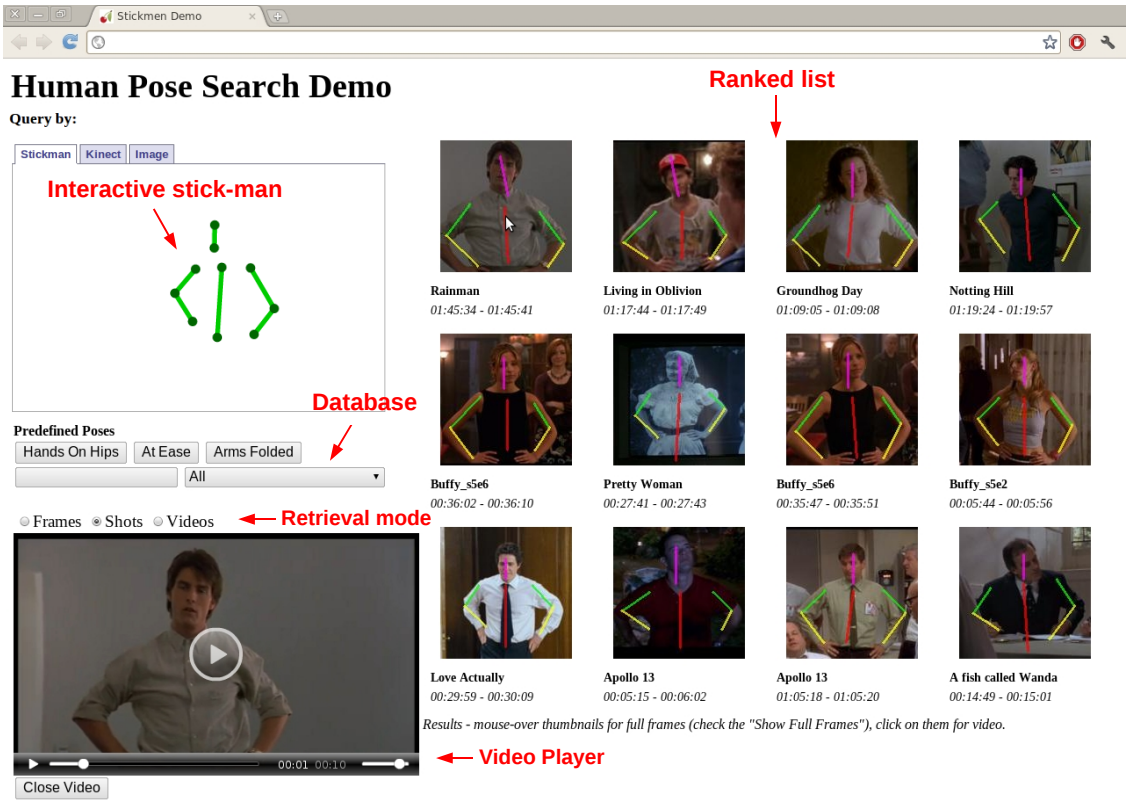


Figure 5.2: **Screen-shot of the pose retrieval system.** The elements of the web page such as the three query modes, options for selecting the database and the level of retrieval and video player, amongst other things, are indicated by text annotations and pointers in light red color. The *interactive stickman* can be moved around and the results are instantly updated as a *ranked list*. Clicking on a thumbnail plays a video of that shot.

K-D trees over a database of $54,000$ pose estimates requires 350 MB of memory and a retrieval time of $5ms$ for 2000 poses.

5.6 Client-Server Architecture

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

5.6.1 Client

The client provides an interface for the user to interact with the system. It is responsible for taking the query from the user, sending it to the server and displaying the retrieved results returned by the

server. Figure 5.2 shows the pose retrieval web demo that we have built. The upper-left corner shows the tabbed interface for three query modes query-by-stickman, query-by-Kinect and query-by-image respectively. In the query-by-stickman mode, the user can choose any pose by moving the arms of the interactive stickman in the image. As the user moves the stickman, the client continuously sends coordinates of stickman as queries to the server. The server responds in real time with the database poses that best match the query. This provides a gratifying and interactive pose search experience. In the query-by-Kinect mode as the user moves, the skeleton of the user output by the Kinect sensor is uploaded to the server. We observe that the Kinect's pose estimation is stable when the user is in full view of the Kinect. To further improve the stability, the skeleton is uploaded to the server only when the Euclidean distance between successive poses is greater than 5 pixels. In the query-by-image mode, the query image is uploaded to the server. The server then detects human poses in the image and relays back the best human pose to the client. The user is also given the option select the database and level (frame/shot/video) to retrieve at. The thumbnail of the matching result, the position of the shot and the movie to which it belongs are displayed. If the user clicks on any thumbnail, the corresponding video shot is played.

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

5.6.2 Server

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

If the input is a stickman, the 12D representation is simply computed by measuring the orientation of the body parts. If the input is an image, the server applies the human pose estimation algorithms and then derives the 12D descriptor just as if it were a database video frame. If the input is a Kinect skeleton, the locations of the neck, head, base of the spine, two wrists, two elbows and two shoulders are used to construct the stickman (see the Kinect example in figure 5.1) The stick corresponding to the torso, for example, is constructed by forming a line segment with the end points as location of the neck and the base of the spine.

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

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

Table 5.2: **Approximate nearest neighbor search vs Exhaustive search.** *The recall of the top 100 ground truth matches is averaged over* 1000 *queries.*

5.7 Experimental Evaluation

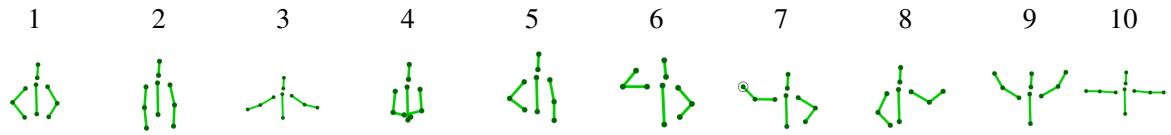
We evaluate the real time pose search system on a large video collection consisting of 18 video and about 3 million frames. Of these 18 videos, 17 are Hollywood movies and one is a season of the TV serial 'Buffy the Vampire Slayer'. The statistics are given in table 5.1.

We first evaluate how accurately does the ANN method retrieve the K nearest neighbors, with exhaustive search as reference. The number of trees used and number of nearest neighbors to be returned are also discussed. We then evaluate how many of the samples retrieved by the pose search system are indeed a true match to the query.

5.7.1 Approximate Nearest Neighbor

To evaluate the ANN algorithm's performs compared to exhaustive search, 1000 random pose estimates are sampled from the whole movie database as queries. Each query is searched in the database using both exhaustive and approximate nearest neighbor search. In the exhaustive search, the query pose is compared using the Euclidean distance to all the elements in the database and the best 100 pose estimates are retained. Next, the search is repeated with the ANN algorithm. These ANN algorithms suffer from low recall. To address this, the standard practice is to retrieve more points and retain the desired number of nearest neighbors closest to the query. In this experiment, the desired number of nearest neighbors is 100 and multiples of 100, $100 * m \; where \; m > 1$, are retrieved using ANN. The performance is measured using *recall at* 100, i.e. the proportion of the ground-truth nearest neighbors that are in the top 100 neighbors returned by the ANN algorithm.

Experiment A: In the first experiment, the recall of the ANN is observed while varying the multiple m. The multiple m is varied with values $\{1, 5, 10, 20\}$, but the number of trees is fixed to 2. As shown in table 5.2 ('Experiment A'), the recall at 100 rapidly improves with m.



	1	2	3	4	5	6	7	8	9	10
Fl2	91.7	83.3	41.7	41.7	25	16.7	8.3	8.3	0	0
Hpm	91.7	75.0	8.3	50.0	41.7	8.3	25	8.3	8.3	8.3

Figure 5.3: **Pose retrieval evaluation:** *The poses displayed in the second row are used to evaluate the performance of the system. The corresponding numbers below are the precision values of the top 12 results. The third and fourth row respectively are the precision values using the 'Filter II (Fl2)' and 'Hybrid pose estimates (Hpm)' methods. The mean precision over the ten pose queries is 31.7% and 32.5% respectively.*

Experiment B: In the second experiment the number of trees, N, is varied, but K is fixed at 2000. Thus on average, K/N neighbors are requested from each tree. As shown in table 5.2 ('Experiment B'), the performance is best for two trees.

Experiment C: In the third experiment the size of the database is varied, for $K = 2000$ and $N = 2$. As shown in table 5.2 the recall is largely unaffected by the database size.

Experiment D: In the fourth experiment the size of the database is varied, for $K = 2000$ and $N = 2$. As shown in table 5.2 ('Experiment D'), the speed gain over exhaustive search is orders of magnitude better and significantly improves with the size of the database.

5.7.2 Retrieval

The pose retrieval system is quantitatively evaluated by posing ten queries to the system and measuring the precision of the retrieved results on the first page of the web application, i.e. the top 12 returns. The queries are chosen to cover a diverse set of poses that people often make.

For this evaluation, we use all the 18 videos listed in table 5.1 and process them as described in section 5.3. We then employ the two best methods (section 4.4), 'Filter II' and 'Hybrid pose estimation', separately to improve the precision. Figure 5.3 shows the queries posed to the system and the corresponding precisions for both the methods, and figure 5.4 shows the first five retrieved results for the three top performing queries.

Analyzing the precision values in figure 5.3 we see that for the best query both 'Filter II (Fl2)' and 'Hybrid pose estimation (Hpm)' methods have 91.7%. Similarly, the mean precision is nearly the same at 31.7% and 32.5% respectively. But a closer look at the precision values of individual queries reveals the differences between both the methods. While 'Filter II' method retrieves high precision results for many queries, particularly for whom many examples in the database, it fails to retrieve any pose for the last two queries. 'Hybrid pose estimation' manages to both retrieve poses at high precision and also retrieve the less frequent ones (queries 9 and 10). In fact for the queries 9 and 10, as far as we can tell there is only one example each in the collection.

Note how these include diverse poses, with arms next to the torso, overlapping with the torso, and stretched out from the torso. This illustrates the versatility of the system. The major impediment to the performance of the system at the moment is the failures of the pose estimation algorithms, which both the methods ameliorate to the extent possible.

Figure 5.4: **Pose retrieval examples:** For the query poses displayed in the first row, five retrieved results are displayed.

5.8 Summary

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

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

Chapter 6

Pose Representation using Deep Poselets

6.1 Introduction

In this chapter, we propose "Deep Poselets", a pose representation based on deep neural networks. This representation implicitly encodes human pose information, thus doing away with explicitly locating the body parts. The 'Deep poselets' can be described as classifiers which detect a subset of body parts in a specific pose. The response of these deep poselets are used to construct a feature representation of the pose, which is used for the pose retrieval. The main contributions of this method are, (a) demonstrating that explicitly clustering the pose space of arms is useful for encoding the pose, (b) demonstrating that a similar architecture to ImageNet-CNN [58] is able to work on the unrelated task of poselet classification, (c) finding areas in the image that have high probability of deep poselets being present, and thereby improving their performance, and (d) empirically demonstrating that deep poselet based pose search is on par with the state-of-the-art pose descriptors. This work [52] is published in FG 2015.

Here we briefly give the outline of the various sections. Deep poselets described in section 6.3, inspired by poselets [13], model a subset of parts (e.g, left upper and lower arm) appearing in a particular pose. The specific poselets and their positive instances are obtained using a data driven process described in section 6.3.1. Given the poselets and instances belonging to them, a classifier is trained to discriminate positive instances from the negatives ones. The features for these classifiers are learnt using CNNs. Motivated by Razavian *e*t al. [75], we use an architecture similar to [58] to learn features. The details of the feature extraction and training are described in section 6.3.3. Given an input test image, all the poselet classifiers are run using the procedure described in section 6.3.4. During the detection stage, mutually exclusive poselet types (e.g., those corresponding to the left arm) fire at the locations with a significant overlap in their detections. This conflict is resolved by spatial reasoning, described in section 6.3.5. Using these deep poselets and their detection scores, a representation for a pose is constructed. The representation is then used to perform pose search as described in section 6.4. In the experimental section 6.5, we evaluate the deep poselet method and the pose search method by comparing them with relevant baselines.

6.2 Related Work

Poselets: Poselets [13] are classifiers which model a subset of body parts. The key difference between Bourdev *et al.* [13] and our method is that Bourdev while *et al.* [13] is for person detection, and ours is for pose detection. A poselet, for example, can model the head and the left shoulder together. The different poselet types are derived from data by randomly selecting a large number of potential candidates and then successively pruning them using various heuristics. Several such classifiers are trained with the objective of detecting a person. All these classifiers are then run on a test image. Based on the relative locations between the detections, the location of the person is estimated. In theory, such a method can be used for human pose estimation when the detected poselets model arms in a specific pose. This of course requires large amounts of data and hence not feasible. The poselets method has recently [14] been improved using CNNs.

To alleviate this problem, Gkioxari *et al.* [45] propose to discover the poselets by using only the image patches corresponding to the arms. Thus each poselet, termed armlet in*et al.* [45], models an arm in a particular pose. These classifiers are then run on the image to obtain detection bounding boxes. Using a pre-trained transfer model, the human joint locations are transferred to the armlet detections. This work by Gkioxari *et al.* [45] is the closest to ours. Both our approach and [45] use body part detectors which are sensitive to pose. While the main focus of [45] is on key point detection, ours is on implicit pose encoding. Further, while we train CNN features specifically for body part detection task using CNNs, Gkioxari *et al.* [45] have used HOG features.

Human pose: The pose retrieval methods of [40, 51, 50] use HPE algorithms. Among the many HPE algorithms, pictorial structures [34] based methods [27, 41, 103] are very popular. Methods such as [70] have integrated a modified version of Berkeley poselets [13] with pictorial structures, while other methods such as [79] have used the poselets for inferring the pose. With the success of convolutional neural networks, a few methods [93, 91] have been proposed using CNN architectures. Even though the performance of HPE is improving, it is not good enough to be used as base technology for tasks such as action recognition and pose retrieval. A single mistake by the algorithm, say a mistaken wrist position, renders the whole pose estimate wrong. Our proposed approach addresses this by softly encoding several locations for each body part.

6.3 Deep Poselets

In this work, a deep poselet is defined as a model which consists of subset of the seven body parts present in a particular pose. The seven body parts used are the left and the right upper arms, the left and the right lower arms, the left and right hip, and the head. Figure 6.1 illustrates a few example deep poselets.

Figure 6.1: **Discovered deep poselets:** Six deep poselets and instances belonging to them are shown. For each deep poselet, an average image marked with stickman and example instances are displayed. A deep poselet is composed of subset of body parts in a particular pose as indicated by the stick figure on the average image. The body parts and their poses in each example instance matches its corresponding deep poselet.

Deep poselet method consists of discovering the deep poselets from the data, training the poselets and finally detecting and post processing them on test images. All these steps are described in great detail in the next few sections. Figure 6.2 illustrates all these four steps.

6.3.1 Deep Poselet Discovery

The deep poselet framework can be understood as a discretization of the pose space, where each state is captured by one deep poselet. We formulate this discretization as a data driven process by clustering the body joints.

In a hypothetical scenario where all poses are equally likely, all possible subsets do occur equally likely. But in reality, the pose distribution of an arm is uneven owing to the factors such as feasibility, energy conservation, popular gestures (e.g., Namaste) and actions. Further, the poses of left arm and right arm are largely decoupled. Thus given any dataset drawn I.I.D. from natural world, poses of just the left arm or the right arm occur much more frequently than say a combination of upper left arm and lower right arm. The head and torso do not contribute much to the human pose (the pose 'Namaste' is the same whether head is straight or tilted) but help in improving the spatial context. With these observations in mind, the following seven subset of body parts, represented by S_i, are clustered. The seven subsets used are (1) the left arm and the left hip, (2) the left arm, left hip, and the head, (3) the left arm and the right hip, (4) the right arm and the right hip, (5) the right arm, right hip, and the head,

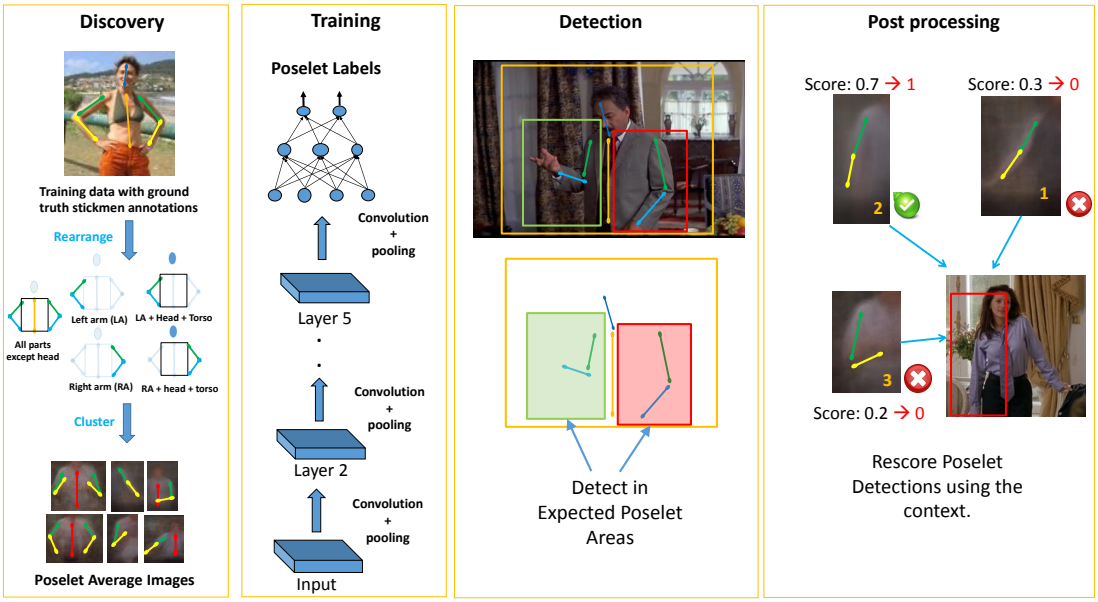


Figure 6.2: **Deep poselet method:** The proposed deep poselet method has four parts: (a) Discovery: First, poselets of various body joint configurations (illustrated in the figure) are discovered by clustering in the pose space (b) Training: These poselets are then trained using convolutional neural networks. (c) Detection: Each poselet has been observed to have a localized area within the upper body bounding box. We term this area as "Expected poselet area (EPA)". The poselet detection is performed within this area. (d) Post processing: The EPA of several poselets intersect (e.g., all poselets belonging to the left arm). Thus within the same area, several poselets have a detections while only small number of them are correct. Using linear regression we re-score the poselets detections using the context of other poselet detections. (Best viewed in color)

(6) the right arm and the left hip, and (7) all body parts minus the head. The left and the right arm are modelled, in three different spatial contexts, by the subsets $\{S_1, S_2, S_3\}$ and $\{S_4, S_5, S_6\}$ respectively. These three spatial contexts are (a) itself, (b) with torso, and (c) with head and torso. The subset S_7 models both the arms and captures the popular poses in the database. The resultant cluster means form an atomic unit of pose and a combination of them describes an upper body pose. Since the body parts modelled by a subset S_i can only take one of N distinct poses and clustering algorithms give unique means, *these cluster means are mutually exclusive to each other*.

Clustering each subset S_i is performed in the following way. First the dataset is pre-processed by computing a bounding box of the person from the stickman annotation. This bounding box is then expanded by extents learnt from the data such that all possible human poses, with their various articulations and extensions of body parts, are contained within the *expanded bounding box*. Next, body parts annotations of subset S_i are x-y normalized by the dimensions of the *expanded bounding box*. These normalized coordinates are concatenated and passed onto a K-means algorithm for clustering. It is observed that when the number of clusters are low, perceptually dissimilar poses fall into the same cluster. But when the number of clusters are large, the similar poses fragment. It is empirically found that for

subsets $s_1, .., s_6$, number of clusters as 20 worked well and for s_7 the number of clusters as 40 worked well. All the cluster which have less than 50 members are discarded. These cluster means are taken as the canonical deep poselets. In our experiments, a total of 122 deep poselets are obtained. Figure 6.1 illustrates a few deep poselets discovered using the above process.

While it is sensible to consider the samples belonging to the deep poselet cluster as positive samples, some of these are perceptually dissimilar to the cluster mean. Further, there are samples whose membership is perceptually ambiguous. Thus for a deep poselet, each sample is classified as belonging to positive class, negative class or ignore class using body part angle (angle made by a body part with the image axis). The samples belonging to ignore class are neither considered while training nor while testing. The classification is done using the procedure described in section 3.3.3.

6.3.2 Expected Poselet Area (EPA)

As deep poselets use CNNs, the sliding window approach for locating the body parts is very expensive during test time. Previous CNN based methods for image classification have solved this problem by using unsupervised object proposal methods like objectness [2] and selective search [96]. Unfortunately, poselets are not whole objects but parts of a specific object (*e*.g, arms as part of human). Thus the above object proposal methods are not useful for the task. We solve this problem by finding the 'expected poselet area (EPA)' in an image. The EPA gives the highly probable location of the deep poselet within the bounding box of the person.

Deep poselets typically occur in a localized region within expanded bounding box. For example, a deep poselet modelling the left arm typically lies in the left half of the bounding box. The search space of the deep poselet can be restricted to this EPA, which improves both the performance and time complexity. The extent of the EPA of a deep poselet is learnt from the positives in the training data. This is done by taking 5 percentile and 95 percentile of the normalized coordinates (normalized w.r.t expanded bounding box) as the extent of EPA respectively. Experiments show that over 95% of the positive instances in both training and test data are encompassed by expected poselet area. This highly precise spatial locality property of poselets ensures that searching only in this area and avoiding the rest of the image (exhaustive search) decreases the probability of false positive occurrence. Thus this improves the accuracy and since the search space is reduced it is computationally efficient.

While EPA encompasses the positives instance well, it also has background area within it. Thus the ground truth area can be any of the possible sub-windows of the EPA. A way to deal with this would be to search for the true detection in the EPA over all possible scales and locations. We simplify the search procedure by fixing the scale of deep poselet to 90% of the EPA and translations to 9 equally spaced sub-windows.

6.3.3 Training

As mentioned before, each deep poselet models a subset of parts in a specific pose. We train a discriminative classifier which can tell apart image regions belonging to this deep poselet from other image regions. We use linear SVMs to train the deep poselets. For the features, we use the representations from CNNs.

In our experiments, we use the implementation of the ImageNet-CNN network by Donahue *et al.* [23]. The ImageNet-CNN [58] is a deep neural network with five convolutional layers and three fully connected layers. Below, the feature extraction and training are explained

6.3.3.1 Feature Extraction

The nine sub-windows of the EPA are passed through ImageNet-CNN in a feed forward manner and the feature maps of the fifth pooling layer (pool5), the first and the second fully connected layers (fc6 and fc7 respectively) are noted. From these three feature maps, the best performing one (details in section 6.5) is used as the representation for the deep poselet.

Further, we fine-tune the ImageNet-CNN to the task of poselet classification so that the CNN takes an image region as input and outputs the poselet class label or background. For fine-tuning, the last fully connected layer of the ImageNet-CNN is replaced by a 123 (122 deep poselets and a background class) neuron fully connected layer. The weights of the newly added layer are randomly initialized. The weights of the rest of the layers are initialized from the ImageNet-CNN [23]. It has been observed that the sample strength ratio between the largest poselet class and the smallest poselet class is 80. To compensate for this skew, the data of the classes with low strength are augmented by their translated versions. The original learning rates are decreased by a factor of 10 so that the existing weights do not significantly change. For the first two fully connected layers, a dropout rate [88] of 0.5 is used. For training the network, the cuda-convnet software [57] is used.

6.3.3.2 Learning SVMs

Typically an EPA has significant background areas. Thus the ground truth area can be any of the possible sub-windows of the EPA. To select the correct sub-window a Multiple Instance Learning (MIL) approach is used [3]: After extracting the feature representations from the nine sub-windows of all EPAs, an initial linear SVM model is trained. For this, all the sub-windows are given the same label as the EPA. Using this initial SVM, the best scoring sub-windows are selected and a new SVM model is trained. This process is repeated until there is no change in the AP on the validation set. In practice, it is found that three iterations suffice. Empirically, this procedure improved the AP by 7% over the method in which all candidate windows are used for training. This procedure is reminiscent of best positive bounding box selection used in Felzenswalb *et al.* [37].

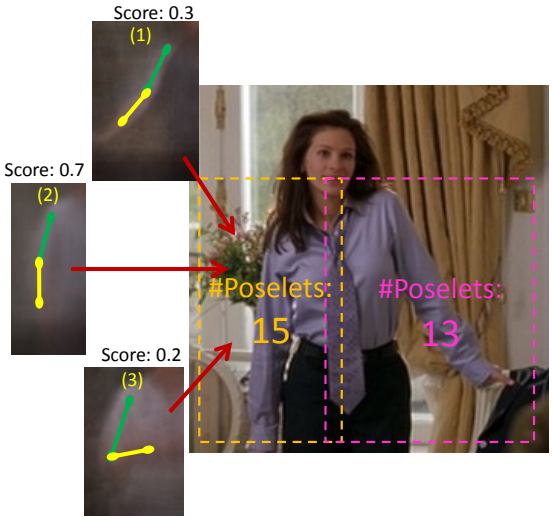


Figure 6.3: **Spatial reasoning:** For a given test sample, three deep poselet detections and their scores are shown as belonging to the area marked by an orange rectangle. Detections 1 and 3 are partially correct as the pose of the left upper arm matches that of the test sample. Detection 2 is the correct one. Typically many such deep poselet detections, often mutually exclusive, have significant overlap. Using spatial reasoning, these detections are rescored such that correct ones (detection 2) get a score of nearly 1 and the partially or totally incorrect ones (detection 1 and 3) get a score of nearly 0. The image also shows that area around the left arm (orange rectangle) has 15 unique deep poselets while area around the right arm (pink rectangle) has 13 unique deep poselets.

6.3.4 Testing

Given a test image, it is processed using the human detector algorithm to obtain upper body detections. Each upper body detection is then transformed to obtain the expanded bounding box. For each deep poselet, the corresponding EPA (expected poselet area) is computed using the learnt transformation (section 6.3.2). The EPA is then divided into nine equally spaced sub-windows with the scale of each sub-window at 90% of EPA. Each sub-window is passed onto the deep poselet model to obtain a score. The sub-window with the best score is noted as the deep poselet detection.

6.3.5 Spatial Reasoning

On an image with a person in it, typically most of the deep poselets fire, when only a few of them are correct. Many of these deep poselet detections significantly overlap, while being mutually exclusive. Figure 6.3 illustrates this behavior. In the figure, three deep poselet detections corresponding to the left arm are displayed. Clearly they are mutually exclusive because the arm can be present in only one of the three poses represented by them. This conflict is resolved by rescoring the deep poselet detections using other mutually exclusive deep poselet detections as context. The expected outcome is that the correct detections (detection 2 in the figure 6.3) have a score of nearly 1 and incorrect ones (detections

1 and 3 in the figure 6.3) have a score of nearly 0. For this rescoring, a RBF kernel based regression model [25] is learnt for each deep poselet type P. The input to this model is a feature vector comprising of calibrated scores (procedure in the next paragraph) of the P's own detection and its mutually exclusive deep poselets and the output is the new score. For training, the above feature is provided as input and the binary label of the deep poselet detection is provided as target value. Given a test sample, first all the deep poselets are run on the sample and then the above regression models are applied to re-score each deep poselet detection. Below the procedure for calibration and finding mutually exclusive poselets are described.

Calibration: Calibration ensures that scores of various deep poselets are comparable. This is achieved by mapping the scores of all deep poselets to the $[0, 1]$ interval. We use the method proposed by Platt [72], in which a logistic regression model is learnt with the deep poselet score as input. Let $X \in R$ be the scores of the deep poselet detections D. A mapping $\sigma : X \rightarrow Y$ where $X, Y \in R$ is learnt. The function $\sigma(x)$ is parameterized by w_0, w_1 and is given by,

$$\sigma(x) = \frac{1}{1 + e^{(w_1 x + w_0)}}. \tag{6.1}$$

Mutually exclusive deep poselets: For each deep poselet type P, a mutually exclusive poselet is defined as one which occupies the same area in the person bounding box. For example, the three detections in figure 6.3, which are mutually exclusive, occupy the same area. The following procedure is used to find the mutually exclusive deep poselets. First the 'expected poselet areas' (section 6.3.3) of all the 122 deep poselets are collected. These deep poselets are then clustered using the cluster partitioning algorithm proposed by Ferrari *et al.* [42]. The algorithm returned 31 clusters, where poselets in each cluster form a mutually exclusive set.

6.4 Pose Representation and Pose Search

In this section, we first describe our pose search approaches. We then review three standard retrieval methods for the pose search task. Later in the chapter (section 6.5.3), we compare the proposed pose search method against standard retrieval schemes described below. All the methods below take an expanded bounding box as input.

Proposed deep poselets: Given a test image, all the deep poselets are run on it using the procedure described in section 6.3.4 and the detection scores are noted. Using the human detector's output, all the deep poselet detections are clustered by the person to which they belong. These deep poselet detections are then rescored using spatial reasoning (section 6.3.5). Finally a feature vector of K dimensions, where K is the number of deep poselet detectors, is constructed by max pooling the detections. The feature is then l_2 normalized. Thus for each upper body in the dataset, a feature vector is constructed.

Given a query image, a feature representation is created using the method described above and it is compared against all the samples in the dataset using Euclidean distance. The samples in the dataset are sorted by distance and presented to the user.

Dataset	Train	Validation	Test	Total
H3D [13]	238	0	0	238
ETH PASCAL [27]	0	0	548	548
Buffy [41]	747	0	0	747
Buffy-2 dataset [50]	396	0	0	396
Movie dataset [50]	1098	491	2172	3756
FLIC [79]	2724	2279	0	5003
MPII Human pose [4]	6742	0	0	6742
Poses in wild [19]	660	0	0	660
We are family [29]	1290	0	0	1290
Synchronic Activities [30]	1112	0	0	1112
Total	15007	2764	2720	20491

Table 6.1: The contributions of various datasets before adding the flipped versions.

Bag-of-visual words models [85]: Given a training data composed of images with people in various poses, the SIFT features are extracted at the key points and 1000 visual words are obtained. Given a test upper body detection, the SIFT features are extracted in the expanded bounding box and bag of words representation is obtained using the visual words computed from the training data. This representation is then compared against all the images in the database. The distances or similarity scores are sorted to obtain the ranked list.

Human pose estimators [103], [18], [68]: Following the method proposed by Jammalamadaka *et al.* [51], the HPE algorithms are used for the pose search task as described below. First the pose estimation algorithms Yang and Ramanan [103], Chen and Yuille [18] and Pfister *et.al,* [68] are run on all the expanded versions of the upper body detections in the database to obtain the pose estimates. These HPE algorithms give the locations of various body joints by efficiently searching over multiple scales and all possible translations. For each pose estimate, the sine and cosine of upper and lower parts of both the arms are extracted to form a pose representation. Given a test upper body bounding box, the above procedure is applied to obtain the pose representation. It is then compared against all the instances in the database and the ranked list is obtained after sorting the scores.

Berkeley poselets [13]: Here, all the poselet classifiers are run on an image to obtain poselet detections. These poselet detections are then pooled into clusters based on the person bounding box, and are max pooled to obtain a description of the human pose. The above procedure is applied on the database and the representations are stored. Given the query sample the above representation is obtained and is compared against all the samples in the database. The ranked list is obtained by sorting the scores.

Layer	Before fine tuning	After fine tuning
pool5	**67.5**	69.5
fc6	59.7	69.6
fc7	47.4	**69.6**

Table 6.2: Performance of five randomly chosen deep poselets on various CNN features over the test data.

Method	AP-test
HOG poselets	32.6
Deep poselets before fine-tuning	48.6
Deep poselets after fine-tuning	**56.0**

Table 6.3: Comparison of our method with state-of-art poselet methods on the test data.

6.5 Experiments

In this section, we present the experimental evaluation of the deep poselet method and the pose search method. First the data used for both the tasks is described in detail. Then the experimental setup and results for the deep poselet method and pose search method are described.

6.5.1 Data

Training deep poselet classifiers require moderately large amounts of data. For the convenience of pose search method, we consider only those annotations in which all parts are visible. For a partially occluded person, defining a positive instance for retrieval is ambiguous. In all, there are $20,491$ fully visible annotations. The statistics are given in the Table 6.1. To further enhance the dataset size, each image and annotation is horizontally flipped effectively doubling the corpus to $40,982$ stickmen. Using the stickman annotations, the bounding box of the upper body is constructed and transformed into the expanded bounding box. To understand the efficacy of various pose representation schemes, the ground truth bounding box is assumed.

The combined dataset of $40,982$ samples is divided into training, validation and test datasets. The training dataset consists of Buffy stickmen dataset [41], H3D dataset [13], Buffy-stickmen II dataset [50], five movies from the movie stickmen dataset [50] and twenty movies from FLIC dataset [79]. The validation dataset consists of one movie from movie stickmen dataset [50] and ten movies from FLIC dataset [79]. The testing dataset consists of ETH PASCAL dataset [27] and the remaining five movies from the movie stickmen dataset [50]. This division of data ensures that training and testing datasets have no overlap in movies and helps in evaluating the methods on unseen data. The individual contributions of various datasets to the train, validation and test data are given in table 6.1.

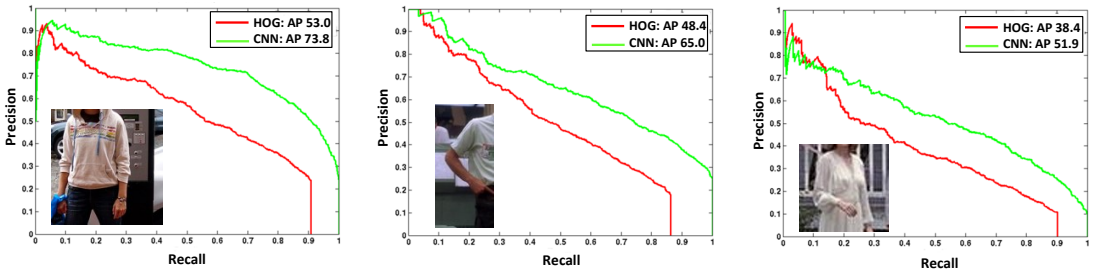


Figure 6.4: **Deep poselets vs HOG poselets:** The graphs show the performance of three deep poselets on test data. The red curve in each graph corresponds to HOG poselet while the green curve corresponds to the deep poselet. As can be seen, the deep poselet outperforms the HOG poselet.

6.5.2 Deep Poselets

Given a set of deep poselet detections and ground truth bounding boxes, the deep poselet performance is reported in terms of average precision (AP) in the following way. First all the deep poselet detections in an image are compared against the ground truth bounding boxes using the intersection over union measure (IOU). All the detections which have more 0.35 IOU, a value used in [13], are considered as positive. All the detections are then sorted in the decreasing order of score and AP is calculated using the labels.

Deep poselets: Using the procedure described in section 6.3.3, deep poselets are trained using CNN features extracted from the ImageNet network [23], before and after fine-tuning it. The hyper-parameters are set using 3-fold cross validation. We experiment with the features from last pooling layer (pool5), the first (fc6) and second (fc7) fully connected layers. Table 6.2 shows the performance of deep poselets using features from different layers averaged over five randomly chosen deep poselets on the testset. For deep poselets using features before fine tuning the network, the last pooling layer (pool5) works best. This is expected as the network is trained on a very different task of object detection. For the deep poselets using the features after fine tuning the network, the features from second fully connected layer (fc7) works best. The deep poselets using features after fine tuning consistently outperform those which use features before fine tuning.

HOG poselets: To baseline the performance of the deep poselets, we compare it with poselets which use HOG features. In this method, a linear SVM is trained using the standard hard-negative mining approach [37]. For the positive samples, the HOG feature is extracted in the bounding box. For the negative samples, the HOG feature of all possible bounding boxes in scale and translation space are considered. Given a test sample, the classifier is run on all scales and locations. All the detections which are above a pre-determined threshold (95% recall on the training data) are deemed as positive detections. Further, all the poselet detections which do not overlap more than 0.35 IOU with the 'expected poselet area' (section 6.3.3) are discarded. This step improves the average AP by 10%.

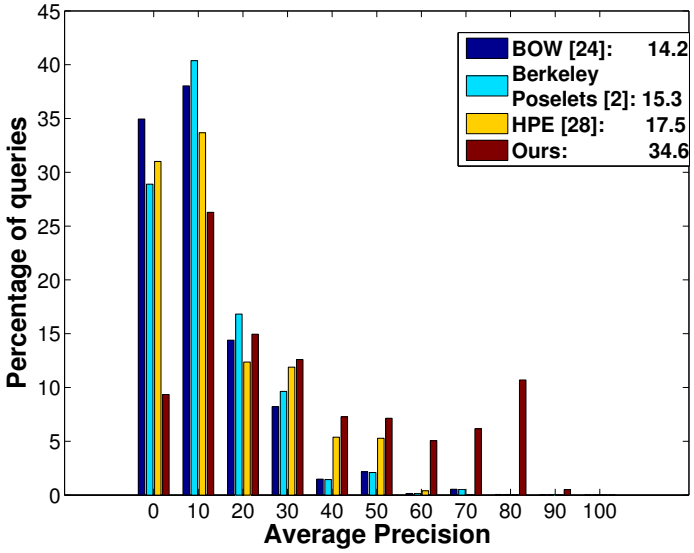


Figure 6.5: **Pose search performance:** The distribution of query performances by various retrieval methods are shown. Each bar in the graph shows the percentage of queries (Y-axis) having an average precision (X-axis). Thus the more the number of queries on the right side of the graph the better the method. This is also reflected by the mean of the distribution (mAP) of various methods given in the top right corner. It is clear that the proposed method significantly outperforms other methods.

Table 6.3 shows the performances of HOG poselets and deep poselets. These values are averaged across all the 122 classifiers. It is apparent from the numbers that deep poselets outperform the HOG poselets. It is also observed that out of 122 deep poselets, 118 of them using features before fine-tuning and 120 of them using features after fine-tuning outperform the HOG poselets. Figure 6.4 compares the AP curves of HOG poselets and deep poselets. Figure 6.6 shows the example detections of three deep poselets. As illustrated in the figure, the performance of the deep poselet improves with more training data.

6.5.3 Pose Representation and Pose search

Given a query image, the feature representation is computed and its similarity score or distance is computed with all samples in the test data. These scores are then sorted to obtain a ranked list. The label for each sample in this list, which indicates if the sample has a similar pose as the query, is determined using the part angles as described in section 3.3.3. Using the ranked list and labels, average precision (AP) is calculated. Each sample in the test data is used as a query to retrieve the results, thus evaluating the various retrieval methods on a total of 5440 queries, the size of test data. The pose search task is evaluated using mean average precision (mAP), which is the average of APs over all the queries.

Table 6.4 shows the mAPs of various methods over all the queries and the dimension of the pose representation. As is evident, the proposed deep poselet method, with a mAPs of 34.6%, significantly

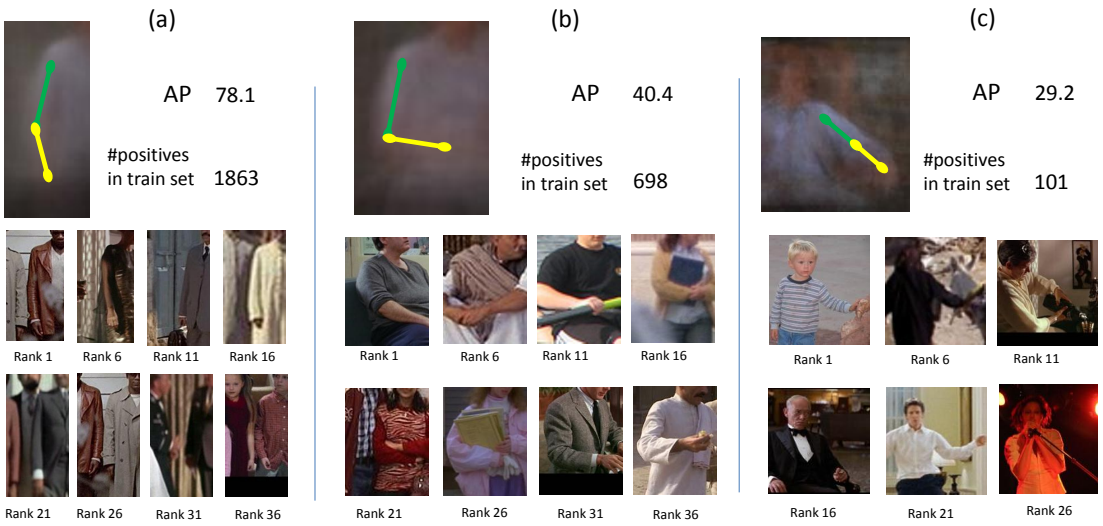


Figure 6.6: **Top deep poselet detections:** Three deep poselets and top detections by them are shown. For each deep poselet, every fifth detection is displayed. In the top 50 detections, while there are no mistakes in deep poselet (a), there are 4 mistakes in deep poselet (b) and 20 mistakes in deep poselet (c). In the deep poselets (b) and (c), the first mistakes occur at ranks 20 and 10 respectively. It can be seen that the performance of deep poselets improve as the number of training samples increases.

Methods	#Dimension	mAP
Bag of Visual Words [85]	1000	14.2
Berkeley Poselets [13]	150	15.3
Human Pose Estimation [103]	8	17.5
CNN-HPE I [68]	8	23.8
CNN-HPE II [18]	8	**37.1**
Ours - Deep Poselets	122	32.9
+ Spatial Reasoning	122	**34.6**

Table 6.4: Pose search performance (mAP) and pose representation's dimensions of various methods.

outperforms the traditional methods with the best of them at 17.5%. The table also shows that applying spatial reasoning for deep poselets has improved the mAP from 32.9% to 34.6%, an improvement of 1.7%. The new CNN based human pose estimation algorithms 'CNN-HPE I' and 'CNN-HPE II', which are currently the state-of-the-art on several datasets, do better than their traditional counter-parts. The 'CNN-HPE II' algorithm mildly outperforms our algorithm by 2.5%. We have to note that these both algorithms have used sophisticated modelling while ours uses a standard and relatively small neural network. We strongly believe that our method will significantly benefit from initializations with a pre-trained model and increasing the depth of the network. The pose representations of HPE algorithms are obtained using the method described in section 4.5. For this experiment, we have encoded just the arms and obtain $8D$ feature vector. Figure 6.5, which shows the distribution of pose search APs over all the queries, gives an insight into our method's better performance. Our methods perform extremely well on

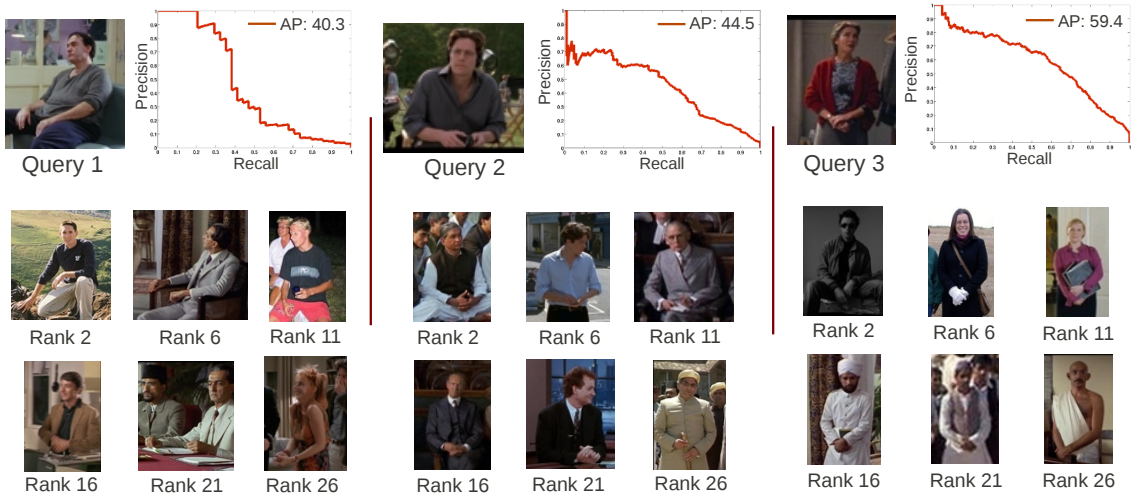


Figure 6.7: **Example retrievals:** Top retrievals and AP curves for three queries are displayed. For the top retrievals every fifth sample from the top in retrieved list is displayed. The first mistake occurs at ranks 11, 4 and 33 respectively for the above queries.

queries such as query 3 in figure 6.7 with APs in the excess of 50%. Such queries have low intra-class variation and high frequency. The second mode on the right in figure 6.5 corresponds to these poses. On queries with rare poses, our method gives better APs, while other methods post near zero APs. Few examples queries and their top retrievals are displayed in figure 6.7.

Each class of methods used for baselining in table 6.5 have weaknesses, analysis of which is presented here.

Bag of visual words [85]: While these methods perform very well for general object retrieval, their performance on pose search suffers because, (a) the loss of geometric context when histogramming the visual words, (b) distracting SIFT detections on clothes, and (c) disproportionately small area of arms and legs with respect to the rest of the bounding box. Our method overcomes this problem by learning to ignore distracting patterns like clothing and identifying the key areas in the bounding box where the arms and outline of the human are present.

Berkeley poselets [13]: A pose sensitive poselet describes the body pose of a person. For example, a poselet corresponding to the whole left arm in a certain pose is pose sensitive while that of face and shoulder is not. A scan through the set of poselets detected by [13] shows that most of the detected poselets are not pose sensitive. This renders the method incapable of detecting the human pose. While, in theory, this method is capable of discovering poselets which model the arms in various poses, it would output far more pose-insensitive poselets. Our method and [45] output a compact set of entirely pose sensitive poselets.

Human pose estimators (HPE) [103]: Most HPE algorithms are modelled as a CRF and the pose estimate is obtained by inferring a maximum a posteriori estimate. Typically maximum a posteriori estimation algorithms decide on one particular location for each body part and can potentially make

a wrong choice. Clearly this affects the pose retrieval as a mistake in one part effectively renders this detection useless and can potentially worsens the performance of the retrieval system. Our method solves this by taking into account several likely alternative locations, while constructing a representation for the pose. Soft coding of pose is the key to the performance of our algorithm.

6.6 Summary

In this chapter, we proposed a novel pose representation, "Deep Poselets", which is based on the convolutional neural networks. We have shown that pose space can be discretized by using 'pose-sensitive' deep poselets. These deep poselet detectors model a subset of body parts in a particular pose. We have shown that using the state-of-the-art CNN [23] features, these detectors perform very well. They have been used as a basic building blocks in constructing a feature representation for pose. We then empirically demonstrated that pose retrieval method based on representations from "Deep Poselet" method are on par with competing pose retrieval methods.

In the next chapter, we describe another novel pose representation based on convolutional neural networks. This method, termed as "deep pose embedding", maps an image to a low dimensional pose-sensitive space. This pose representation too performs on par with other state-of-art pose descriptors on the pose retrieval task.

Chapter 7

Pose Representation using Deep Pose Embedding

7.1 Introduction

In this chapter, we describe a novel way of representing the image of a person striking a pose using the whole image as input. These representations are then used for the retrieval. Inspired by the work of Taylor *et al.* [89], we propose 'Deep Pose Embedding' model which takes an image triplet consisting of a reference image, an image with the similar pose and an image with the dissimilar pose and learns a projection function to a pose-sensitive lower dimensional space. In contrast to our 'Deep poselet' method, this method looks at the complete image and maps it to a lower dimensional space. The main contributions of this method are, (a) demonstrating that an image can be mapped to a pose space using deep networks, and (b) the projection is pose sensitive and performs well on pose retrieval task. The previous works with which the above two methods are compared include Bag of visual words [85], Berkeley poselets [13] and Human pose estimation algorithms [103]. All the methods are quantitatively evaluated on a large dataset of images built from a number of standard benchmarks together with frames from Hollywood movies. This work along with 'Deep poselets' has been accepted to Journal of Image and Visual Computing.

Here we briefly give the outline of the various sections. The deep pose embedding model described in section 7.2 projects an image into lower dimensional pose-sensitive space. The properties of this pose sensitive space, the details of the projection function and training methodology are described in this section. For the projection function, we use three CNNs whose weights are shared. This requires presenting the network with an image triplet. In section 7.3, we describe how to handle the exponentially large triplet combinations. The representations obtained from the above two methods are then used to perform pose search as described in section 7.4. In the experimental section 7.5, we evaluate the deep pose embedding method and the pose search method by comparing them with relevant baselines.

This work is in continuation to our previous work [52] described in chapter 6, where the primary focus was deep poselets. Here, we extend [52] by proposing deep pose embedding model and make interesting connections between both the models.

7.2 Deep Pose Embedding

In this section, we describe the second way of representing the pose from an image. We describe the CNN projection function and the triplet ranking loss used for training.

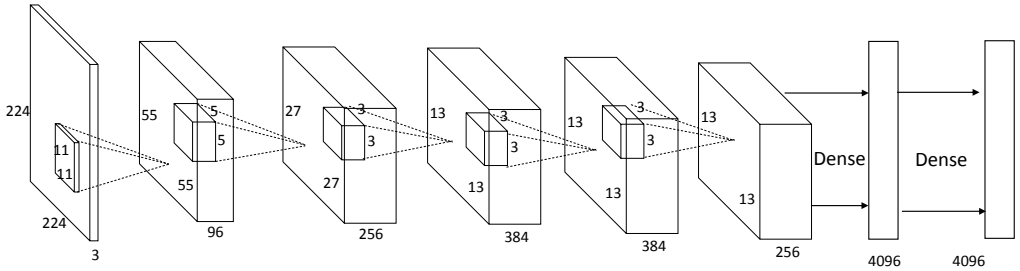


Figure 7.1: **CNN architecture:** This architecture is a minor variation of the CNN architecture proposed in [58]. The number of layers and the number of the parameters are depicted.

Given an image of a person in a particular pose, we project it into a low-dimensional pose-sensitive space. This low-dimensional space has the following structure: (a) The projections of all images with similar poses are near-by, and (b) The projections of images with dissimilar poses are far away.

For learning the projection function, image triplets (x_i, x_i^p, x_i^n) are given which consists of a reference image, an image with similar pose and an image with dissimilar pose respectively. The projection function $f : X \rightarrow Y$ parameterized by w, is learnt such that the L_2 distance d_i^p between the pair (y_i, y_i^p) (here $y = f(x)$) is less than the L_2 distance d_i^n between the pair (y_i, y_i^n) by a margin of 1. Formally, the parameters w are learnt by minimizing the following equation,

$$L = \frac{\alpha}{2}||w||^2 + \sum_{i \in I} \max(0, 1 - (d_i^n - d_i^p)) \tag{7.1}$$

where α is a hyper-parameter to control the amount of regularization and I is the set of indices of the training samples. The above equation is minimized using stochastic gradient descent where the gradient is given by,

$$\frac{\partial L}{\partial w} = \alpha * w + \sum_{i \in I} \frac{\partial g}{\partial w}. \tag{7.2}$$

Here $g = \max(0, 1 - (d_i^n - d_i^p))$ and its gradient is given by,

$$\frac{\partial g}{\partial w} = \begin{cases} 0, & \text{if } 1 - (d_i^n - d_i^p) \leq 0 \\ -\frac{\partial d_i^n}{\partial w} + \frac{\partial d_i^p}{\partial w}, & \text{otherwise.} \end{cases}$$

The gradient of the L_2 distance d between two points (y_1, y_2) is given by,

$$\frac{\partial d(y_1, y_2)}{\partial w} = \frac{1}{d(y_1, y_2)}(y_1 - y_2)^T \left[\frac{\partial y_1}{\partial w_k} - \frac{\partial y_2}{\partial w_k}\right] \tag{7.3}$$

For the projection function, we use convolutional neural networks (CNNs) which are highly non-linear and can handle the articulations of the poses. The architecture of our network is based on krizhevsky *et al.* [58] and is shown in figure 7.1. The network has three identical CNNs, both in terms of architecture and the parameters and is illustrated in figure 7.2. Each CNN has five convolutional layers followed by two fully connected layers. For the non-linearity, we use leaky relu unit [63] which is effective against saturation. The weights of all the layers are randomly initialized from the Gaussian distribution. For the convolutional layers, $(0, 0.01)$ are used as mean and standard deviation respectively. For the fully connected layers $(0, 0.005)$ are used as mean and standard deviation respectively. The size $(width \times height)$ of the first, second and fifth max pooling layers are 3×3 with a stride of 2. The third and fourth convolutional layers are not followed by any max-pooling layers. Both the first and second max-pooling layers are followed by cross map local response normalization with a size of 5×5 and its parameters are given in [58].

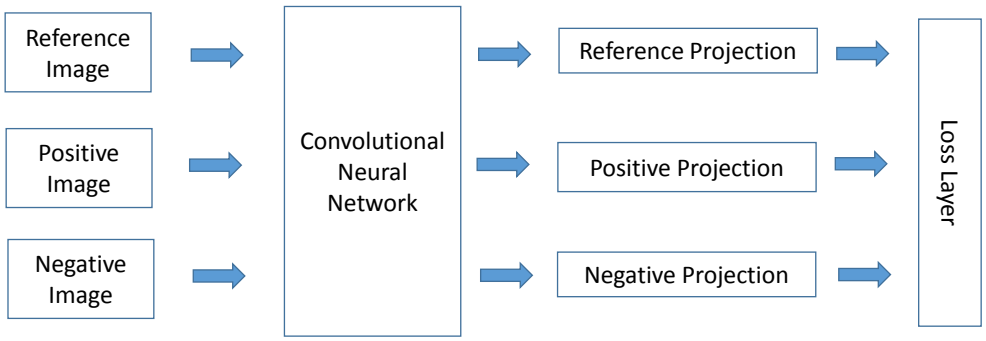


Figure 7.2: **Triplet architecture:** A training sample to this network contains a tuple of three images. The three images are reference image, a positive image which contains the similar pose as reference image, and a negative image which contains a dissimilar pose to the reference image. The images are forward propagated and passed to the loss function which measures how well the network separates reference-positive pair and the reference-negative pair. This architecture can be visualized as containing three networks, each for an image, with same CNN architecture and same set of parameters at any given time.

For training the network, a mini-batch of 128 image triplets X are presented to the network. The sampling strategies and augmentations are discussed in the next sections. For a given image triplet X_i, the three images are passed through the network to obtain feature maps Y_i of the final layer. These feature maps are then used to obtain the gradient (Eqn. 7.2) of the loss function defined in Eqn. 7.1. The weight vectors are then updated using the following equation,

$$w^t = w^{t-1} - \eta \frac{\partial L}{\partial w} + \beta w_m^{t-1} \tag{7.4}$$

$$w_m^t = w_m^{t-1} - \eta \frac{\partial L}{\partial w}. \tag{7.5}$$

$$\tag{7.6}$$

Here w_m is the standard momentum term, $\beta = 0.9$ is the rate of momentum and η is the learning rate. Note that the gradient of the L_2 regularization term in equation 7.1 is accounted for in equation 7.2. The network is trained using the popular deep learning library Theano [7, 9].

7.3 Training Samples

The challenge with models that use triplets is the data explosion. A data set of just ten thousand images can produce a training set of one trillion triplets. While training the CNNS, which requires thousands of data augmentations [58] (through minor translations, scaling and rotations), this problem is further compounded. Given the computational constraints, clearly it is not possible to train the network on all the triplets and their augmentations. Yet, it is not prudent to discard them from which the network can learn. To solve this problem, we propose a method which mines the difficult triplets and their augmentations. By presenting these difficult examples to the network and leaving the simpler examples out, both the computational efficiency and better utilization of data are achieved. The method works by first mining for difficult triplets and then difficult augmentations per triplets. Both these mining steps are described below.

7.3.1 Triplet Mining

Many real datasets follow power law and have significantly higher samples for certain classes. To correct this skew, we organize the data set into K clusters $\{C_1, ..., C_K\}$ where samples belonging to each cluster have similar poses. We obtain these clusters using K-means algorithm described in section 6.3.1. For the first few epochs (5 in our implementation), we randomly sample image triplets in the following way. First R images are randomly sampled from each cluster without replacement. For each sample, T positives and negatives are randomly sampled to form triplets. Thus the total number of triplets are RTK. At the end of these epochs the network would have reasonable estimate of parameters.

After the initial epochs, the data is mined to obtain difficult examples for training. As before, from each cluster C_i, R_i samples are randomly sampled. Each sample is then forward propagated to obtain the loss value. All the sample values whose loss is 0 are discarded. The remaining samples, which the network found difficult, are sent for training. Similar strategies have been used in other applications [82].

7.3.2 Augment Mining

Given a triplet T_i, it is augmented with minor translated, rotated and scaled versions of itself. Each image in the triplet is transformed by small random translation, rotation and scaling several times to obtain the augmentations T_i^j, where $j = 1...N$. In our implementation, we obtain 4 augmentations per triplet. These augmented triplets are then forward propagated to obtain the loss given in Eqn. 7.1. The M augmented triplets (5 in our implementation) with the non-zero loss are retained for training.

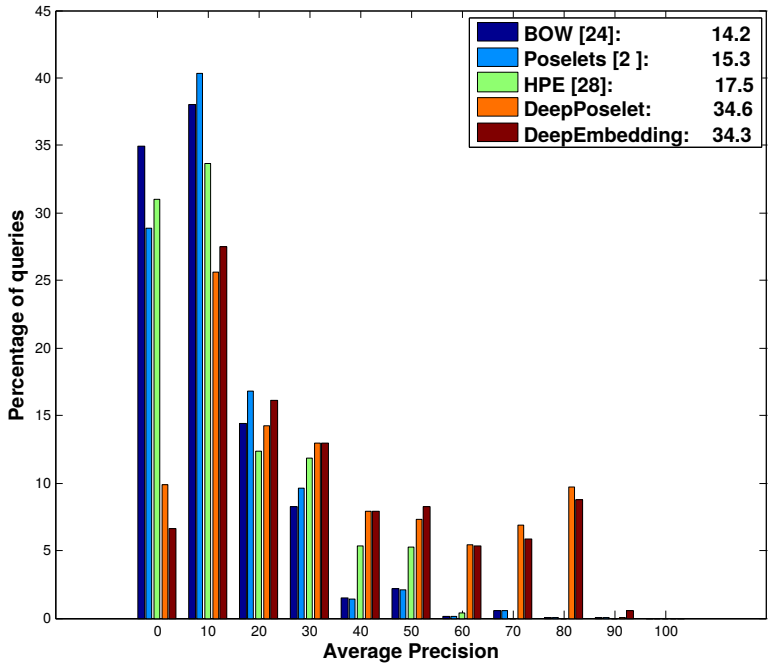


Figure 7.3: **Pose search performance:** The distribution of query performances by various retrieval methods are shown. Each bar in the graph shows the percentage of queries (Y-axis) having an average precision (X-axis). Thus the more the number of queries on the right side of the graph the better the method. This is also reflected by the mean of the distribution (mAP) of various methods given in the top right corner. It is clear that the proposed method significantly outperforms other methods.

7.4 Pose Representation and Pose Search

Given a query image, it is passed through the trained convolutional neural network and the output representation is noted. This representation is compared against all the samples in the dataset using Euclidean distance. The samples in the dataset are sorted by distance and presented to the user.

7.5 Experiments

In this section, we present the experimental evaluation of the deep pose embedding method and the pose search method. First the data used for both the tasks is described in detail. Then the experimental setup and results for the deep pose embedding method and pose search method are described. For the experiment we use same data used for deep poselets method. A very detailed description in given in section 6.5.1.

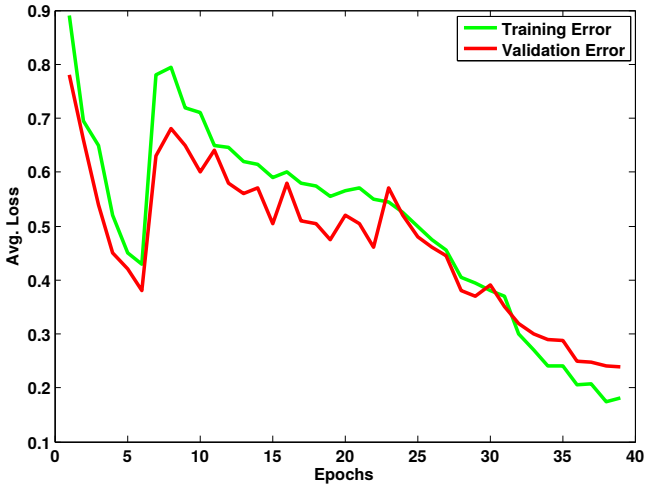


Figure 7.4: **Deep pose embedding network's performance:**

7.5.1 Deep Pose Embedding

Given the training and validation data, the performance of this method is measured using average values of the loss function. The training data is chosen as described in section 7.3. A total of 44 epochs have been generated to train the neural network. For the validation data, we use a total of $20,000$ triplets using the procedure described in section 7.3.1 for the initial set of training epochs. Note that for validation data, we neither discard triplets with zero loss nor do any augmentations. The figure 7.4 displays the performance of both training and validation data over the epochs. The minimum possible loss value is 0 and there is no upper bound on the loss function. Loss value of 0 indicates that the network is able to separate the positive and negative sample by a margin of at least 1. Loss value $(0, 1)$ indicates that the network is still able to separate reference-positive pair and the reference-negative pair but by a margin less than 1. A loss $[1, \inf)$ on a training sample indicates that the distance between the reference-positive pair is more than or equal to than the reference-negative pair.

The plot in figure 7.4 demonstrates that the learning is converging. It also shows how the loss function on validation data has similar error values as on the training data. This indicates that the network is able to generalize well. As described earlier, the first five epochs do not use any data augmentations. After the fifth epoch, augmentation and data mining applied which explains the sudden spike in the plot.

7.5.2 Pose Search

Given a query image, the feature representation is computed and its similarity score or distance is computed with all samples in the test data. These scores are then sorted to obtain a ranked list. The label for each sample in this list, which indicates if the sample has a similar pose as the query, is determined using the part angles as described in section 3.3.3. Using the ranked list and labels, average precision (AP) is calculated. Each sample in the test data is used as a query to retrieve the results, thus evaluating

Methods	#Dimension	mAP
Bag of Visual Words [85]	1000	14.2
Berkeley Poselets [13]	150	15.3
Human Pose Estimation [103]	8	17.5
CNN-HPE I [68]	8	23.8
CNN-HPE II [18]	8	**37.1**
Ours - Deep Poselets	122	32.9
+ Spatial Reasoning	122	**34.6**
Ours - Deep Pose Embedding	4096	34.3

Table 7.1: Pose search performance (mAP) and pose representation's dimensions of various methods.

the various retrieval methods on a total of 5440 queries, the size of test data. The pose search task is evaluated using mean average precision (mAP), which is the average of APs over all the queries.

Table 7.1 shows the mAPs of various methods over all the queries and the dimension of the pose representation. As is evident, the proposed deep pose embedding method, with a mAP 34.3%, significantly outperforms the traditional methods with the best of them at 17.5%. The table also shows that deep pose embedding method and deep poselet method proposed in the previous chapter have very similar performance. The new CNN based human pose estimation algorithms 'CNN-HPE I' and 'CNN-HPE II', which are currently the state-of-the-art on several datasets, do better than their traditional counterparts. The 'CNN-HPE II' algorithm mildly outperforms our algorithm by 2.5%. We have to note that these both algorithms have used sophisticated modelling while ours uses a standard and relatively small neural network. We strongly believe that our method will significantly benefit from initializations with a pre-trained model, increasing the depth of the network and improving the data mining strategies. The pose representations of HPE algorithms are obtained using the method described in section 4.5. For this experiment, we encoded the two arms obtaining $8D$ feature vector.

Figure 7.3 shows an interesting pattern where the AP distributions for the deep poselet method and deep pose embedding method are very similar. This observation throws up questions and we attempt to answer them here: (i) Do they have similar failure cases? If we consider all the queries which have an AP less than 10% as failures, the number of common failure queries between both the methods are about 70% of the total failure queries for either of the methods. This clearly suggests that both the methods have similar failure cases. (ii) What are their strengths and weaknesses? The deep poselet method is easily comprehensible. A failure can be easily understood and can typically be attributed to either a poselet misfiring or the query's pose not being covered by any of the poselets . The flip side is that there are several steps involved in training and building the feature vector. Further a unified method for locating the parts and reasoning the spatial consistency could have better performance. Consider the following analysis of pose retrieval using deep poselets. For both query and the retrieved image, the poselet labels and detection scores are noted. First, each poselet is classified into one of the three categories: (a) Positive-Positive, (b) Positive-Negative or (c) Negative-Negative where, for example, "Positive-Negative" label would mean one of query/retrieval is positive and other is nega-

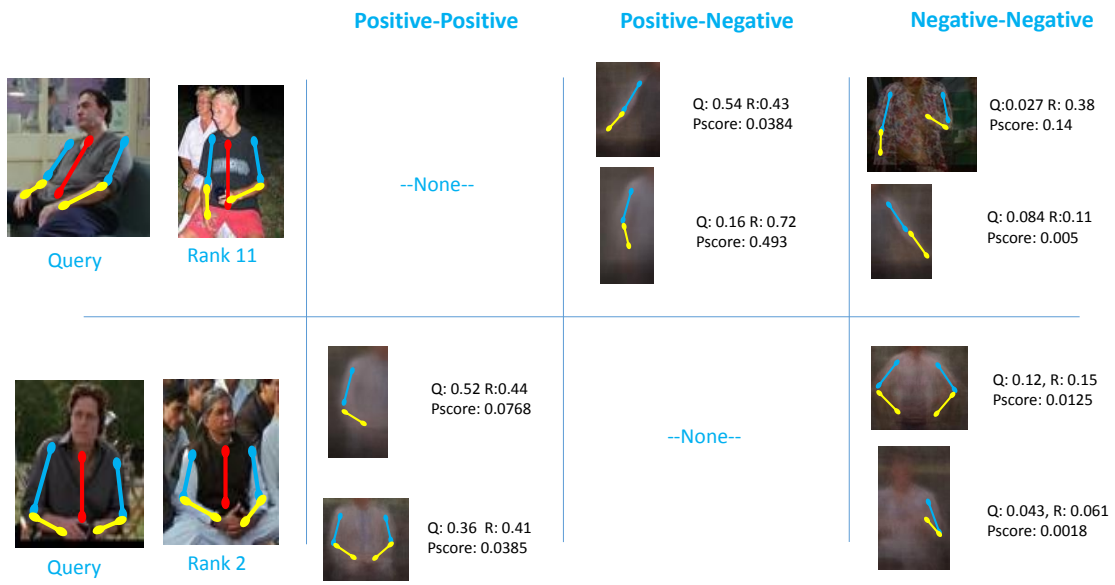


Figure 7.5: **Deep poselet analysis:** For the two query-retrieval pairs, the top and bottom poselets based on prominence scores are displayed for all three poselet categories. For the first pair (above), the retrieval is incorrect. This analysis clearly shows that the top poselet in positive-negative category has misfired for the retrieval image. Similarly the top poselet in negative-negative category also misfired. For the second pair (below), the retrieval is correct and all the prominence scores reflect it.

tive. In order to understand which of the poselet detections have affected the result most, the product of detection-score-sum and detection-score-absolute-difference is noted and is termed as prominence score. For Positive-Positive and Negative-Negative poselets, the prominence score should be small and for Positive-Negative poselets the prominence score should be large. Note that each poselet category are independently analyzed. The prominence scores of poselets belonging to the same category are appropriately sorted (ascending order for Positive-Negative poselets and descending order for other pairs). The poselets at the top of the sorted list are responsible for wrong retrieval and poselets at the bottom are responsible for correct retrieval. Figure 7.5 demonstrates this analysis on two query-retrieval pairs.

7.6 Summary

In this chapter, we successfully proposed a novel pose descriptor for image and video search using pose as a query modality. We proposed the deep pose embedding method to obtain the pose descriptors and perform the pose retrieval. We have shown how an image can be directly mapped to a lower dimensional pose-sensitive space. We then empirically demonstrated that pose retrieval using our method is on par with the state-of-the-art pose desciptors and is on par with our deep poselet approach (chapter 6).

Chapter 8

Conclusions

Human pose is an important information conveying action and gesture. It is also a base technology which is useful in solving other problems like cloth parsing [102, 49]. Retrieving images using human pose is a significant capability addition to content based information retrieval (CBIR) systems. In this thesis, we have empirically demonstrated that retrieving images using human pose is feasible. While building the real time pose retrieval system, we have solved several related problems. Our contributions are summarized below.

In order to demonstrate that human pose retrieval is feasible, we have built a real time pose retrieval system on 22 Hollywood movies. The first challenge we solved was the querying mechanism. Pose as a query modality does not intuitively fit into query-by-text paradigm. While query-by-exemplar is good alternative, it is too tedious for the user to find the image of a person with the desired pose. We proposed a query-by-stickmen and query-by-kinect mechanisms. Both these modalities are highly intuitive and very natural for human pose. In query-by-stickmen the user simply has to adjust an articulate stickmen and in query-by-kinect the user can strike a pose at kinect. The second challenge is the scalability over large collection of images and videos. We solved this problem by proposing three low dimensional human pose representations (described below) and indexed these representations using approximate nearest neighbor methods. Finally we built an attractive user interface which takes the query, gets a list of samples with similar pose from the server and presents it to the user. We empirically demonstrated that our system is working on a range of queries.

Pose descriptors are critical in any pose retrieval system. Our first descriptor is based on the output of HPE algorithms [5],[26],[80],[103]. Here we found that the HPE algorithms are unreliable and are affecting the performance of the retrieval system. We solved this problem by proposing a human pose evaluator which can automatically assess if the output of the algorithm is correct or not. We empirically demonstrated on four different HPE algorithms [5],[26],[80],[103] that the human pose evaluator can indeed predict if the algorithm is correct or not. Further, we have shown that using the output of the evaluator, the best pose estimate from among the outputs of various HPE algorithms empirically outperforms the individual algorithms. It also outperforms the individual HPE algorithms after their outputs

have been filtered using the pose evaluator. After the pose estimates are passed through the human pose evaluator stage, we built a very compact pose representation of $12D$.

Our second pose descriptor is based on poselets [13], a set of body part detectors. We refined the definition of poselets as body part detectors which are sensitive to human pose. We demonstrated that these poselets can be automatically detected from the data. Using CNNs [60] we demonstrated that these poselets significantly outperform poslets which are based on HOG. Given a new test image, we then ran all these poselets and max pooled them to obtain our second pose descriptor. We empirically demonstrated that the pose retrieval system with this pose descriptor is on par with the state-of-the-art pose descriptors.

For our third pose descriptor, we learnt a projection function which projects the image to a lower dimensional pose sensitive manifold. We used a triplet network which takes a tuple consisting of three images as input sample. The three images are reference image, an image with similar pose and an image with a dissimilar pose. The triplet network is then trained using a ranking loss which penalizes when the reference-dissimilar pair are close by or the reference-similar pair are far away or both. Here again we empirically demonstrated that the descriptor is on par with the state-of-the-art pose descriptors and is on par with the deep poselet descriptor.

Future work

While many elements of our work can benefit from more training data and better engineering, we believe that the following three works have scope for much greater exploration.

Automatic Algorithm Evaluator: In our work, we proposed the automatic evaluator for a particular vision algorithm, namely HPE algorithms, and for a class of machine learning models, namely undirected graphical models which generate marginals and max-marginals while inferencing. We believe that our work is a mere demonstration of the concept and in some form all the vision algorithms need an automatic evaluator of some kind. Thus the future work in this direction would be to extend these automatic evaluators to other vision algorithms. Given that deep neural network based algorithms have taken over computer vision, we believe an evaluator for deep neural network class machine learning models would be a good direction to work on.

Deep embedding: In our deep pose embedding work, we have used a CNN architecture which was optimized for object classification task where large variances and coarse-scale features are required. Pose is a very fine grained information and clearly optimizing the CNN architecture can be explored. Further we have used a very simple ranking loss. We believe much more interesting losses can be formulated which exploit the fact that the body joint locations are known for training data.

Improving retrieval using temporal information in videos: Temporal information can be used to improve (a) pose representation using human pose estimation algorithms and (b) shot retrieval. Human pose estimation algorithms have been shown [19] to improve their performance when temporal consistency has been accounted for. This in turn will improve the pose representation. For the shot retrieval,

we retrieve the shots with the best matching frame. Using temporal information shot retrieval can be improved by determining if the desired pose is held for long or if it is a pose in transition. Clearly former is prefered to latter.

Pose retrieval to Action retrieval:

As we mentioned several times in this thesis, pose, action and gesture are very related and even dependent events [53]. We believe our pose retrieval system can form basis for action or gesture retrieval systems.

Related Publications

- Nataraj Jammalamadaka, Andrew Zisserman, Marcin Eichner, Vittorio Ferrari, and C. V. Jawahar. Has my algorithm succeeded? an evaluator for human pose estimators. In European Conference on Computer Vision (ECCV), Florence, Italy, 2012.
- Nataraj Jammalamadaka, Andrew Zisserman, Marcin Eichner, Vittorio Ferrari, and C. V. Jawahar. Video retrieval by mimicking poses. In International conference on multimedia retrieval, HongKong, China, 2012.
- Nataraj Jammalamadaka, Andrew Zisserman and C.V. Jawahar. Human Pose Search using Deep Poselets. In International Conference on Automatic Face and Gesture Recognition (FG), Ljubljana, Slovenia, 2015.
- Nataraj Jammalamadaka, Andrew Zisserman and C. V. Jawahar, Human Pose Search using Deep Networks. Accepted to Journal of Image and Vision Computing, 2016

Bibliography

[1] G. Aggarwal, S. Biswas, P. Flynn, and K. Bowyer. Predicting performance of face recognition systems: An image characterization approach. In *Proc. CVPR workshops*, pages 52 –59, 2011.

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE PAMI*, 34(11):2189–2202, Nov 2012.

[3] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[4] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proc. CVPR*, June 2014.

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

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

[7] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[8] S. Belongie and J. Malik. Matching with shape contexts. In *CBAIVL00*, pages 20–26, 2000.

[9] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, june 2010. Oral Presentation.

[10] A. Blake, P. Kohli, and C. Rother. *Markov Random Fields for Vision and Image Processing*. MIT Press, 2011.

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

[12] M. Boshra and B. Bhanu. Predicting performance of object recognition. *IEEE PAMI*, 22(9):956 –969, 2000.

[13] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. *Proc. ICCV*, 2009.

[14] L. D. Bourdev, F. Yang, and R. Fergus. Deep poselets for human detection. *CoRR*, abs/1407.0717, 2014.

[15] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. Signature verification using a siamese time delay neural network. In *NIPS*, pages 737–744, 1993.

[16] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. pages 121–167, 1998.

[17] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[18] X. Chen and A. Yuille. Parsing occluded people by flexible compositions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] A. Cherian, J. Mairal, K. Alahari, and C. Schmid. Mixing body-part sequences for human pose estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[20] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. CVPR*, pages 539–546, 2005.

[21] N. Dalal and B. Triggs. Histogram of Oriented Gradients for Human Detection. In *Proc. CVPR*, volume 2, pages 886–893, 2005.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. CVPR*, 2009.

[23] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[24] G. Dorkó and C. Schmid. Object class recognition using discriminative local features. *IEEE PAMI*, 2004.

[25] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. In *NIPS*, 1996.

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

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

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

[29] M. Eichner and V. Ferrari. We are family: Joint pose estimation of multiple persons. In *Proc. ECCV*, pages 228–242, 2010.

[30] M. Eichner and V. Ferrari. Human pose co-estimation and applications. *IEEE PAMI*, 34(11):2282–2288, 2012.

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

[32] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *IJCV*, 99:190–214, 2012.

[33] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[34] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1), 2005.

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

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

[37] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 32(9):1627–1645, 2010.

[38] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006.

[39] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proc. CVPR*, 2008.

[40] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Pose search: Retrieving people using their pose. In *Proc. CVPR*, 2009.

[41] V. Ferrari, M. J. Marín-Jiménez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Proc. CVPR*, 2008.

[42] V. Ferrari, T. Tuytelaars, and L. J. V. Gool. Real-time affine region tracking and coplanar grouping. In *Proc. CVPR*, 2001.

[43] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, c-22(1):67–92, Jan 1973.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014.

[45] G. Gkioxari, P. Arbelaez, L. Bourdev, and J. Malik. Articulated pose estimation using discriminative armlet classifiers. In *Proc. CVPR*, 2013.

[46] G. Gkioxari, B. Hariharan, R. B. Girshick, and J. Malik. R-cnns for pose estimation and action detection. *CoRR*, abs/1406.5212, 2014.

[47] A. Gupta, Y. Verma, and C. V. Jawahar. Choosing linguistics over vision to describe images. 2012.

[48] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[49] N. Jammalamadaka, A. Minocha, D. Singh, and C. V. Jawahar. Parsing clothes in unrestricted images. In *Proc. BMVC.*, 2013.

[50] N. Jammalamadaka, A. Zisserman, M. Eichner, V. Ferrari, and C. V. Jawahar. Has my algorithm succeeded? an evaluator for human pose estimators. In *Proc. ECCV*, 2012.

[51] N. Jammalamadaka, A. Zisserman, M. Eichner, V. Ferrari, and C. V. Jawahar. Video retrieval by mimicking poses. In *ACM ICMR*, 2012.

[52] N. Jammalamadaka, A. Zisserman, and C. V. Jawahar. Human pose search using deep poselets. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2015.

[53] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *Proc. ICCV*, pages 3192–3199. IEEE, 2013.

[54] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

[55] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Proc. CVPR*, pages 90–96, 2004.

[56] R. Kindermann and J. L. Snell. *Markov random fields and their applications. Contemporary Mathematics, 1*. American Mathematical Society, Providence, R.I., 1980.

[57] A. Krizhevsky. *Cuda-Convnet: a fast C++/CUDA implementation of convolutional neural networks*.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

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

[60] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

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

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

[63] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing (WDLASL)*, 2013.

[64] O. Mac Aodha, G. J. Brostow, and M. Pollefeys. Segmenting video into classes of algorithm-suitability. In *Proc. CVPR*, 2010.

[65] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISSAPP)*, pages 331–340. INSTICC Press, 2009.

[66] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[67] M. Perdóch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Proc. CVPR*, 2009.

[68] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *IEEE International Conference on Computer Vision*, 2015.

[69] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

[70] L. Pishchulin, M. Andriluka, P. V. Gehler, and B. Schiele. Poselet conditioned pictorial structures. In *Proc. CVPR*, 2013.

[71] L. Pishchulin, A. Jain, M. Andriluka, T. Thormahlen, and B. Schiele. Articulated people detection and pose estimation: Reshaping the future. In *Proc. CVPR*, pages 3178 –3185.

[72] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, 1999.

[73] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, pages 61–74. MIT Press, 1999.

[74] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*. MIT Press, 2006.

[75] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[76] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In *Proc. ECCV*, 2002.

[77] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. *Proc. ACM SIGGRAPH*, 23(3):309–314, 2004.

[78] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3.

[79] B. Sapp and B. Taskar. MODEC: multimodal decomposable models for human pose estimation. In *Proc. CVPR*, 2013.

[80] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision ECCV 2010*, volume 6312 of *Lecture Notes in Computer Science*, pages 406–420. Springer Berlin Heidelberg, 2010.

[81] W. Scheirer, A. Bendale, and T. Boult. Predicting biometric facial recognition failure with similarity surfaces and support vector machines. In *Proc. CVPR*, pages 1 –8, 2008.

[82] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, 2015.

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

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

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

[86] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, volume 2, pages 1470–1477, 2003.

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

[88] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[89] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus. Learning invariance through imitation. In *Proc. CVPR*, pages 2729–2736, 2011.

[90] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. *CoRR*, abs/1406.2984, 2014.

[91] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014.

[92] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proc. CVPR*, pages 1653–1660, 2014.

[93] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proc. CVPR*, 2014.

[94] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. ICML*, 2004.

[95] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[96] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[97] Y. Verma, A. Gupta, P. Mannem, and C. V. Jawahar. Generating image descriptions using semantic similarities in the output space. In *Proc. CVPR workshops*, pages 288–293, 2013.

[98] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[99] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.

[100] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proc. CVPR*, pages 1386–1393, 2014.

[101] R. Wang and B. Bhanu. Learning models for predicting recognition performance. In *Proc. CVPR*, volume 2, pages 1613 –1618 Vol. 2, 2005.

[102] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. Parsing clothing in fashion photographs. In *Proc. CVPR*, 2012.

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

[104] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, Dec 2013.