Advancing Human Pose and Gesture Recognition

DPhil Thesis

Supervisor: Professor Andrew Zisserman

Tomas Pfister

Visual Geometry Group

Department of Engineering Science

University of Oxford

Wolfson College

April 2015

Tomas Pfister
Wolfson College

Doctor of Philosophy
April 2015

Advancing Human Pose and Gesture Recognition

Abstract

This thesis presents new methods in two closely related areas of computer vision: human pose estimation, and gesture recognition in videos.

In human pose estimation, we show that random forests can be used to estimate human pose in monocular videos. To this end, we propose a co-segmentation algorithm for segmenting humans out of videos, and an evaluator that predicts whether the estimated poses are correct or not. We further extend this pose estimator to new domains (with a transfer learning approach), and enhance its predictions by predicting the joint positions sequentially (rather than independently) in an image, and using temporal information in the videos (rather than predicting the poses from a single frame). Finally, we go beyond random forests, and show that convolutional neural networks can be used to estimate human pose even more accurately and efficiently. We propose two new convolutional neural network architectures, and show how optical flow can be employed in convolutional nets to further improve the predictions.

In gesture recognition, we explore the idea of using weak supervision to learn gestures. We show that we can learn sign language automatically from signed TV broadcasts with subtitles by letting algorithms 'watch' the TV broadcasts and 'match' the signs with the subtitles. We further show that if even a small amount of strong supervision is available (as there is for sign language, in the form of sign language video dictionaries), this strong supervision can be combined with weak supervision to learn even better models.

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Tomas Pfister, Wolfson College

Acknowledgements

I'm grateful to my supervisor, Andrew Zisserman, for his great guidance and inspiration. I have learnt an unimaginable amount during the past 4 years in his lab. Always looking forward to our weekly Friday morning meetings, from which most of the ideas in this thesis emerged.

Thanks to my parents, who supported my move to the UK for my undergrad, and for always being there when I needed help.

I'm also grateful to the many funding agencies that have supported the work in this thesis – particularly EPSRC who funded my first year and Osk. Huttunen Foundation who funded the majority of my remaining years.

I really appreciate all the help I have received from various lab members – particularly Andrea, Chai, Karen, Ken, Max, Minh, Omkar, Relja, Varun and Yusuf. And James, who's a co-author on all my papers, and has diligently stayed up with me to midnight or 1am for every paper submission deadline!

Finally, this thesis wouldn't exist without my wife Sophia. Many (if not most) of the ideas in this thesis originate from discussions with her. If a thesis could have a co-author, she should definitely be one!

Contents

Chapter 1

Introduction

1.1 Objective and Motivation

The objective of this thesis is to advance the state of the art in two closely related areas of computer vision: human pose estimation and gesture recognition.

In human pose estimation, the objective is to track the positions of human body parts in images and videos (as shown in Figure 1.1). In particular in this thesis, we focus on estimating human pose from RGB images without depth. While some commercial systems (such as Kinect) have recently shown some promise tackling this problem using depth sensors, human pose estimation from raw RGB remains very challenging.

In gesture recognition, the objective is to recognise *intentional human body movements* (most commonly expressed by hands and face), and classify them as one of many gestures in some predefined *gesture language*. This is, likewise, a

Figure 1.1: **Human pose estimation.** Here shown for upper-body joints.

very challenging task – particularly for complex gesture languages with 1,000s of gestures (such as sign languages).

These two areas are closely related: human pose estimation methods are often used to localise the body parts (particularly the hands and face) to aid gesture recognition (*e.g.* by simplifying the task to recognising gestures by matching the trajectories of the hands; or using hand positions to derive the *shape* of the hands).

1.2 Applications

Robust pose estimation has a wide range of applications – including tracking body parts in gaming, human-computer interaction, augmented & virtual reality, healthcare; and helping solve other challenging problems, such as human action recognition, activity analysis, automated surveillance, content-based image indexing and retrieval, markerless motion capture, and gesture recognition.

Likewise, gesture recognition has a plethora of applications. It is being very actively explored both in academia, and also in industry for various applications

– *e.g.* for video gesture control (GestureTek, Omek, Flutter), virtual reality control and 'telepresence' (Oculus Rift, Nimble, Qualcomm), gesture-based computer control for surgeons (GestSure), human-computer interaction (Leap Motion, eyeSight), rehabilitation of people after serious accidents (Roke), and sign language recognition (*e.g.* using the Leap Motion tracker – MotionSavvy).

Sign language recognition. One application this thesis will explore is sign language recognition. Here, the task is to automatically identify a sequence of sign language gestures and understand their meaning. The ultimate aim is a method that automatically translates sign language into text/speech, enabling deaf people to communicate with hearing people without a human interpreter. To date, the few commercial translation services in existence rely on human translators, who perform the translation either in person or online using a webcam. So, needless to say, a robust automated solution for this application could have a tremendously positive impact on the daily lives of deaf individuals throughout the world.

These sign languages ('spoken' by about 70 million deaf people [World Federation of the Deaf]) consist of 1,000s of gestures which convey information using multiple channels: *manual* (hand gestures: including hand shape, movement, location, hand orientation), *non-manual* (facial expressions, body posture, head pose, lip patterns), and *finger spelling* (in which words are 'spelt out' with gestures, letter by letter) [Stokoe, 2005]. Sign language is not an international language – *e.g.* there is a separate sign language for America, Britain and Sweden [Wheatley and Pabsch, 2010]. They evolved independently of the spoken languages around them, and possess the structural properties of other human

languages (although with a quite different grammar from *e.g.* English) [Stokoe, 2005].

One approach to tackling sign language recognition is to collect a large dataset of gestures in a given sign language (say British Sign Language) performed by many people, and hand-labelling the gestures. However, this is (i) quite expensive (as it requires a lot of labour-intensive manual labelling); (ii) has to be done separately for each sign language; (iii) needs to be done for a large number of signers (to deal with inter-signer variations); and (iv) needs to be regularly repeated (because sign languages are continually evolving). These challenges have motivated investigations into alternative forms of supervision – most prominently, from signed TV broadcasts.

Learning sign language by watching TV. The idea here is to automatically learn sign language by letting algorithms 'watch' sign language TV broadcasts with subtitles (with an overlaid signer translating to the deaf audience), and 'match' the signs with the subtitles – much like what a human would do when attempting to learn a (sign or spoken) language. Since sign language-interpreted broadcasts are shown for many hours every day (and in many different sign languages), this provides a near-infinite resource of training data.

In particular, this material can be exploited to learn *signs* corresponding to English (or Swedish, Chinese ...) *words* in the subtitles, effectively building a database of word-sign pairs for a large number of signs and signers (which can be used to train a sign language to text translator). The mechanism for learning

the sign generally exploits the supervision by, for a given English word, selecting a set of subtitles and videos that contain the word ('positive sequences') and a set of subtitles and videos that do not (the 'negative sequences'), and then looking for signs that are common in the positive and do not occur in the negative sequences. Although this may sound straightforward, in practice doing this is very challenging due to the weak and noisy nature of this supervision (as we will discuss below).

1.3 Challenges

1.3.1 Pose estimation

Despite a long history of research, human pose estimation remains a very challenging task in computer vision. It is very challenging (as demonstrated in Figure 1.2) due to the: (i) high variability of human visual appearance in images (due to viewing angle, lighting, different clothing, background); (ii) variability in human body shape (size of different body parts); (iii) occlusions (either by the human itself or by objects between the camera and the human); (iv) high dimensionality of the possible poses (think of the crazy poses of yoga masters!); (v) loss of 3D information in RGB images (leading to additional ambiguities); and (vi) motion blur due to the low frame rate of many cameras.

A few of these challenges are shown for signed TV broadcasts in Figure 1.3: self-occlusions of the person, self-shadowing, motion blur due to the speed of the gesturing, overlapping hands, multiple people behind each other, and an ever-

Figure 1.2: **Challenges in human pose estimation.** The examples here demonstrate the high dimensionality of human poses.

changing background (which can be very similar in colour to the foreground).

1.3.2 Gesture recognition

Gesture recognition is challenging because of both high inter-person and intra-person variability. For example, sign languages contain a lot of variations in: (i) speed (*e.g.* a gesture could be 0.2s long or 2s long, depending on how much in a 'hurry' the gesturer is); (ii) positional (*e.g.* some persons – or sometimes even the same person on different days – may perform the same gesture at chest-level, some in front of the face – yet they may have the same meaning); (iii) regional (*e.g.* the same 'word' can be a completely different-looking sign in different regions

(a) (b) (c) (d)

Figure 1.3: **Challenges in human pose estimation for sign language TV broadcasts.** (a) Similar foreground and background colours render the colour cue less informative; (b) motion blur removes much of the edges of the arm; (c) another face in the background; (d) proximity of the two hands makes the assignment to left and right hand ambiguous.

of the UK).

The main challenge, however, is that the salient information 'channel' varies a lot for different gestures/signs – for some signs the main 'channel' might be simply the start and end position relative to face of the right hand (*i.e.* speed, intermediate hand positions between start and end, and absolute position are irrelevant); in another, the channel might be the speed and the *trajectory* of both hands plus a particular facial expression (*i.e.* the intermediate hand positions – the start and end positions are irrelevant); or the salient channel could be a hand movement away from the body towards the camera (which wouldn't be discernible in RGB without depth).

Discerning the salient information is made even more difficult by *coarticulation* in gesturing, where the transition from the previous and to the next gestures affect the 'current' gesture (by essentially 'cutting away' parts of the information in the current gesture and making it look less like the 'prototype' of the gesture). Moreover, producing linguistically valid translations of complex ges-

Figure 1.4: **Challenges in learning sign language by watching TV.** The supervision is inherently weak (as the subtitles only define *roughly* where a sign occurs) and noisy (subtitled words aren't always signed).

ture languages (such as sign language) also requires dealing with grammar issues such as word order, modifiers, compounding (new concepts derived by combining words), placement (placing particular referents to a particular position in the signing space), and many more.

Challenges in learning sign language by watching TV. Above we briefly touched upon the idea of automatically learning sign language by watching signed TV broadcasts with subtitles. This is in fact very challenging due to the *weak* and *noisy* nature of this supervision, as illustrated in Figure 1.4. The supervision is weak because the subtitles are not temporally aligned with the signs – a sign (typically 8–13 frames long) could be anywhere in the overlapping subtitle video sequences (typically 400 frames). It is noisy because there the occurrence of a word in the subtitle does not always imply that the word is signed. Together, these points make this correspondence problem (trying to identify the temporal window of the sign) very difficult to solve.

In addition to the challenging supervision, previous research tackling sign language recognition using this approach [Buehler et al., 2009, Cooper and Bowden, 2009] has also been held back by the difficulty of obtaining a sufficient amount of training data with human pose annotated (which is used to simplify the matching bit of this correspondence problem). This is one of the motivators for our work on fast, accurate and automatic human pose estimation in this thesis.

1.4 Contributions and Thesis Outline

For human pose estimation (Part I), we make the following contributions:

1. **Chapter 4:** We show that random forests can be used to estimate human pose in monocular videos (in the spirit of Kinect, but here for RGB rather than RGB-D videos). In addition, we propose a co-segmentation algorithm for segmenting humans out of videos (which we use in our pose estimator), and an evaluator that predicts whether the estimated poses are likely to be correct or not.

2. **Chapter 5:** We further extend this random forest-based pose estimator to new domains (with a transfer learning approach), and enhance its predictions with new methods that predict human joints sequentially (rather than independently), and use the temporal information in the videos to 'propagate' pose prediction confidence temporally (rather than predicting the pose from a single frame).

3. **Chapter 6:** We show that convolutional neural networks can be used to estimate human pose even more accurately and efficiently than with a random forest. We propose two new convolutional network architecture types for human pose estimation, and show how optical flow can be employed in convolutional nets to further improve the predictions.

For gesture recognition (Part II), we make the following contributions:

1. **Chapter 7:** We explore the idea of using *weak* (instead of strong) supervision to learn gestures, and show that we can learn sign language automatically from signed TV broadcasts with subtitles. Furthermore, we show that correlations between the mouth and hand movements of signers can be used to significantly cut down the search space when automatically learning sign language gestures using this approach.

2. **Chapter 8:** We show that if even a small amount of *strong* supervision is available (as there is for sign language, in the form of sign language video dictionaries), this strong supervision can be combined with weak supervision to learn even better models.

1.5 Publications

This thesis has led to eight papers:

1. BMVC'12 [Pfister et al., 2012] (oral). This won the Best paper honourable mention award and Best video award. (Chapter 4)

2. IJCV'13 [Charles et al., 2013a] is an extension of the paper published at BMVC'12. (Chapter 4)

3. BMVC'13 [Pfister et al., 2013]. (Chapter 7)

4. BMVC'13 [Charles et al., 2013b] (oral). (Section 5.1)

5. BMVC'14 [Charles et al., 2014]. This won the Best poster award. (Sections 5.2 and 5.3)

6. ECCV'14 [Pfister et al., 2014a]. (Chapter 8)

7. ACCV'14 [Pfister et al., 2014b]. (Section 6.1)

8. ICCV'15 (under submission). (Sections 6.2 and 6.3)

Chapter 2

Literature Review

This literature review first discusses past work related to human pose estimation (Section 2.1), and then to gesture and sign language recognition (Section 2.2).

2.1 Human Pose Estimation

We first review past work on human pose estimation.

2.1.1 Overview

There is a vast array of literature regarding human pose estimation, due to many applications reliant on analysing people in images and video (gaming, human-computer interaction, security and gesture recognition).

Looking back at past work on human pose estimation, four distinct *eras* emerge:

1. Early methods (1980–)
2. Pictorial structure-based methods and poselets (mainly 2000–)
3. Random forests (2011–)
4. Deep convolutional neural networks (ConvNets) (2012–)

The next sections discuss the methods in each of these eras in detail.

These discussions are followed by detailed *case studies* of two particularly relevant works: (i) [Buehler et al., 2011]: pictorial structures for pose estimation; and (ii) [Charles et al., 2013a]: random forests for pose estimation. The first work is used for generating training annotations for a dataset in this thesis, and Chapter 4 uses the second work.

A comprehensive survey of pose estimation works up to 2011 is provided in [Moeslund, 2011].

2.1.2 Early methods

The earliest pose estimation methods addressed the problem with classic model-based approaches [Bregler and Malik, 1998, Forsyth and Fleck, 1997, Hogg, 1983, O'Rourke and Badler, 1980, Rohr, 1994]. One of the earliest approaches was the idea of 'body plan' [Forsyth and Fleck, 1997], which models object layout by defining what parts of an object can be grouped together (and how). [Mori and Malik, 2002] proposed a simple model for find a nearest neighbour based on shape context, and transferring joint locations. Other works included retrieving

poses from segmentations [Ren et al., 2005], and estimating pose directly from skin colour [Hua et al., 2005].

2.1.3 Pictorial structures and poselets

Pictorial Structures [Fischler and Elschlager, 1973] model the body parts of a human as a conditional random field (CRF). In most methods, the parts are parametrised by location (x, y), orientation θ and scale s, and correspond to the human limbs. Generally, the posterior of a configuration of parts is a function of a unary potential $u(\boldsymbol{I}|l_i)$ (which evaluates the local image evidence for a part given a position) and a pairwise potential $pw(l_i, l_j)$ (which is a prior for the relative positions of parts in the human kinematic chain). Together, these yield the optimisation target

$$p(\boldsymbol{L}|\boldsymbol{I}) \propto \exp(\sum_{(i,j)\in M} pw(l_i, l_j) + \sum_i u(\boldsymbol{I}|l_i)). \tag{2.1}$$

In most works, the prior M on the relative positions of the parts is a tree, which enables the model to be fitted to an image in time quadratically proportional to the number of parts [Fischler and Elschlager, 1973]. Furthermore, [Felzenszwalb and Huttenlocher, 2000, 2005] show that by restricting the pairwise potentials to a certain form (using distance transforms), the maximum a posteriori (MAP) of the above equation can be computed in linear time. Because of the fairly low complexity of inference of pictorial structures, they have been used in many applications [Ramanan, 2006, Ramanan et al., 2005, Sivic et al., 2006].

In more recent work, the focus has been on improving the appearance models used in pictorial structures for modelling the individual body parts [Alahari et al., 2013, Andriluka et al., 2012, Eichner and Ferrari, 2009, Eichner et al., 2012, Ferrari et al., 2008, Johnson and Everingham, 2009, Sapp et al., 2010]. Gradient-based cues have been used to design better part detectors, for example based on templates of discriminatively learnt HOG descriptors [Johnson and Everingham, 2009, Kumar et al., 2009]. [Eichner and Ferrari, 2009, Eichner et al., 2012] improved the colour cue by exploiting relations between the appearance of different parts of the body. [Sapp et al., 2011] model body joints rather than limbs, and also track joints across frames, using a set of tree-structured sub-models.

[Yang and Ramanan, 2011, 2013] and deformable part-based models. Building upon the pictorial structure framework, [Felzenszwalb et al., 2008, 2010] proposed deformable part-based models. [Yang and Ramanan, 2011, 2013] used a mixture of deformable parts in a tree structured model to efficiently model human pose. They learnt unary and pairwise potentials using the effective discriminative parts-based model from [Felzenszwalb et al., 2008] (based on structured SVMs). Unlike in traditional models, their parts corresponded to the *mid and end points* of each limb (vs the actual limbs in previous work). The parts were modelled as a mixture in order to capture the orientations of the limbs, and their method searches over multiple locations, scales and all part mixtures. This is a well-performing pose estimator, to which we compare our method in Chapter 4.

Poselets. Another recent stream of works worth mentioning is 'poselets' [Bourdev and Malik, 2009, Gkioxari et al., 2013, 2014]. They learn parts by forming tight patch *clusters* in both 3D pose and 2D image appearance, clustered with datasets that have 3D pose annotations. These are then used to train an SVM for each poselet (cluster), which is used to match poses in a sliding window fashion.

2.1.4 Random forests

A different approach to part-based models is to tackle the problem of pose estimation holistically, without decomposing the problem into smaller, conditionally independent pieces. Instead, the pose is estimated directly from the image (often without any explicitly specified spatial model). We next review recent work in this line using random forests. In the next section, this will be followed by a discussion of similar approaches using deep convolutional neural networks.

The innate versatility of random forests (RFs) [Amit and Geman, 1997, Breiman, 2001] makes them suitable for a variety of machine learning tasks [Criminisi et al., 2012], such as classification, regression and clustering. RFs are a collection of decision trees, whose split node tests are recursively trained (with supervised training data) to maximise the information gain at each node going down the tree. They are naturally multi-class and contain a structure which lends itself to parallelisation and multi-core implementations [Sharp, 2008]. Along with these properties, the ever increasing computing power and training data over recent years has spurred the interest in RFs and fern-based [Ozuysal et al., 2010] methods in computer vision literature. In particular, as RFs are fast to resolve at

inference time, many methods have explored using them for real-time applications of tracking [Lepetit and Fua, 2006, Santner et al., 2010], head pose estimation [Fanelli et al., 2011] and detecting facial feature points [Cootes et al., 2012, Dantone et al., 2012, Fanelli et al., 2012].

Human pose estimation from depth with Random Forests. For human pose estimation, notable success has been achieved with random forests using depth imagery. [Shotton et al., 2011, 2013] segmented a 3D depth map of a person into body parts and used the segmentation as an intermediate stage for computing body joint locations. A performance boost to this original method was proposed by [Girshick et al., 2011], who used regression forests and Hough voting. Further improvements were obtained by [Taylor et al., 2012] using an RF to form dense correspondences between depth image pixels and a 3D body model surface, enabling the use of a one-shot optimisation procedure for inferring the pose. Recently [Sun et al., 2012] have conditioned the RF on a global variable, such as torso orientation.

Human pose estimation from RGB with Random Forests. The success of these RF-based pose estimation methods depends upon the use of depth imagery. Depth images are colour and texture-invariant, which makes background subtraction much easier, and substantially reduces the variability in human appearance. The remaining variability due to body shape, pose and camera angle is accounted for by training with large quantities of data. In contrast, in RGB (without depth), achieving robustness to these variabilities is challenging. To this

end, in Chapter 4 we propose an upper body pose estimation method that creates a colour and texture-invariant representation from raw RGB, which enables the use of random forests for robustly estimating the pose in raw RGB videos. Another recent work tackling this problem is [Kazemi et al., 2013].

Exploiting spatial information: pose structure and independence of output variables. Existing random forest-based methods assume independence for each output variable, and ignore the output structure [Girshick et al., 2011, Shotton et al., 2011]. Past solutions to this have been of two kinds: post-processing methods, and implicit methods. Post-processing methods take the output of the Random Forests and fit models to them, such as Markov or Conditional Random Fields [Jancsary et al., 2012, Payet and Todorovic, 2010], or simply filter the output by checking global consistency of local detections [Yang and Patras, 2013]. Usually post-processing methods are rather slow due to the additional overhead. In contrast, implicit methods build constraints between output variables into the detection method directly during training [Kontschieder et al., 2013, Tu and Bai, 2010] by passing the output from a sequence of classifiers as an input into another classifier. In Chapter 5 we present a method for RFs that address these issues by combining the benefits of both types of approaches (post-processing and implicit).

2.1.5 Deep convolutional neural networks (ConvNets)

Many recent works have demonstrated the power of ConvNets in a wide variety of vision tasks – object classification and detection [Girshick et al., 2014, Krizhevsky et al., 2012, Oquab et al., 2014a,b, Sermanet et al., 2014, Zeiler and Fergus, 2014], face recognition [Taigman et al., 2014], text recognition [Alsharif and Pineau, 2014, Goodfellow et al., 2014, Jaderberg et al., 2014], video action recognition [Karpathy et al., 2014, Simonyan and Zisserman, 2014] and many more [Donahue et al., 2014, Osadchy et al., 2007, Razavian et al., 2014]. These networks comprise several layers of non-linear feature extractors and are therefore said to be 'deep' (in contrast to classic methods that are 'shallow').

Recent works have also explored the use of ConvNets for estimating the human pose. [Toshev and Szegedy, 2014] proposed to use a cascade of ConvNet regressors to improve precision over a single pose regressor network. [Jain et al., 2014a, Tompson et al., 2014] proposed a hybrid architecture combining ConvNets with a Markov Random Field-based spatial model. [Chen and Yuille, 2014] combine a parts-based model with ConvNets (by using a ConvNet to learn conditional probabilities for the presence of parts and their spatial relationship with image patches). Effectively, they use ConvNets to learn better appearance features in a DPM (replacing HOG); the resulting optimisation target is solved as usual with MAP. [Jain et al., 2014b] investigated the use of temporal information in videos using ConvNets (using optical flow as a feature). [Tompson et al., 2015] proposed adding a pose refinement model (based on a Siamese network with shared weights) upon a rougher pose estimator ConvNet.

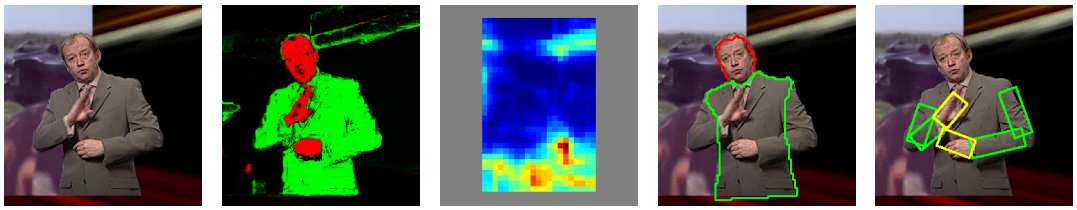
In Chapter 6, we present two computationally efficient novel network architectures that outperform previous work in video pose estimation.

2.1.6 Case study 1: [Buehler et al., 2011]

[Buehler et al., 2011] uses a generative model for both the foreground (signer) and background (the image area surrounding the human) to estimate the upper-body pose with pictorial structures. The foreground is generated by rendering colour models of the limbs and torso in back-to-front depth order (the 'painter's algorithm') so that occlusions are handled correctly. The background is likewise rendered with colour models (which are updated every frame to account for the changing background). The computational expenses of evaluating the renderings is reduced by sampling from a pictorial structure proposal distribution.

This previous state of the art method for tracking arms and hands in sign language TV broadcasts is powerful, but faces two challenges:

1. It requires manual labelling for each new input video. The method requires 64 manually labelled frames for each new input video, which is around three hours of manual user input per one hour of TV footage. Manually labelled data is required to: (i) build the head and torso model (20 shapes per video; examples shown in Figure 2.2); (ii) learn the part-specific colour distributions (5 segmentations shown in Figure 2.3); and (iii) create HOG templates (39 frames with the arm configuration manually specified, see Figure 2.1e).



(a) Input image (b) Colour term (c) Gradient term (d) Segmentation (e) Arm pose

Figure 2.1: **Overview of [Buehler et al., 2011].** Pose estimation for input image (a) is performed using likelihoods from colour (b) and Histogram of Oriented Gradients (HOG) (c). The colour term (b) shows the posterior probability for skin and torso assigned to red and green colour channels respectively. The HOG term (c) shows the the likelihood at all locations in the image (red indicates high likelihood) for a learnt lower-arm template (peaking here at the centre of the left lower arm). Using the colour term (b), the head and torso are segmented (d). The final arm pose (e) is estimated with a global cost function that uses the estimated torso and head shape, plus the appearance terms (from colour and HOG).

2. It is computationally expensive. The method is computationally expensive, which means it cannot be used for real-time pose estimation. On average, the method takes 100 seconds per frame on a 1.83 GHz machine.

We next give a more detailed description of the method (with figures borrowed from [Buehler, 2010]). We provide a detailed discussion of this work because this method is used for generating the ground truth for the pose estimation datasets used in this work (Chapter 3).

Overview. The generative model in this work explains every pixel in the image with a cost function which assigns a cost to a given configuration of the upper body. The pose estimation process is divided into two stages: (1) the shape of the head and torso, and the position of the shoulders, are estimated; (2) given the head and torso segmentations, the configuration of both arms and hands is

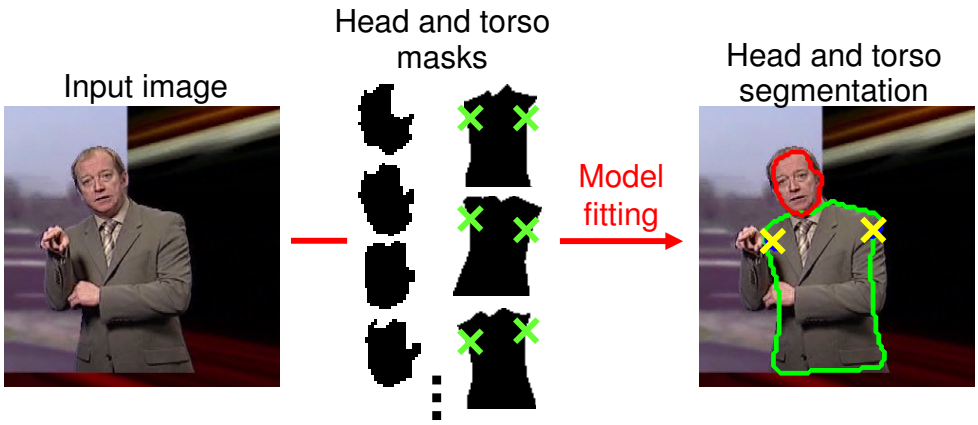


Figure 2.2: **Building the head and torso model for [Buehler et al., 2011] (Stage 1).** Manually labelled masks for the head and torso (for 20 frames) are fitted to the image using a pictorial structure model with two parts.

estimated with a cost function.

Stage 1: Estimating head and torso positions

The head and torso positions are estimated by segmenting the head and torso using multiple manually specified candidate shapes as templates, and fitting these templates to the image using a simple two part pictorial structure (PS) model (see Figure 2.2). The templates used for fitting are manually specified for each new input video.

Stage 2: Estimating arm and hand configuration

Global cost function. Given an image $\boldsymbol{I}$ that contains the upper body of the person and background, the target of this method is to find the hand configuration $\boldsymbol{L} = (b, \boldsymbol{l}_1, \boldsymbol{l}_2, ..., \boldsymbol{l}_n)$ which best explains the image, where $\{\boldsymbol{l}_i\}$ specifies the parts (limbs), b is a binary variable indicating the depth ordering of the two arms, and

Figure 2.3: **Manual segmentations required for learning part-specific colour distributions in [Buehler et al., 2011].** For each new input video, a manually specified segmentation is required (such as this one – into face, hands and torso).

$n = 6$ parts (the left and right upper arms, the lower arms and the hands).

The parts $\boldsymbol{l}_i = (s_i, \alpha_i)$ are specified by three parameters: (i) scale s_i (*i.e.* length of a part that models foreshortening); (ii) rotation α_i; and (iii) by the part to which it is connected (these connections follow the kinematic chain for the left and right arm respectively as shown in Figure 2.4).

The probability of a given configuration $\boldsymbol{L}$ conditioned on the image $\boldsymbol{I}$ is

$$p(\boldsymbol{L}|\boldsymbol{I}) \propto p(\boldsymbol{L}) \prod_{i=1}^{N} p(\boldsymbol{c}_i|\lambda_i) \prod_{j \in \{LL,LR\}} p(\boldsymbol{h}_j|\boldsymbol{l}_j) \tag{2.2}$$

where N is the number of pixels in the input image, $\boldsymbol{c}_i$ is the colour of pixel i, and $\boldsymbol{h}_j$ is the HOG descriptor computed for limb j.

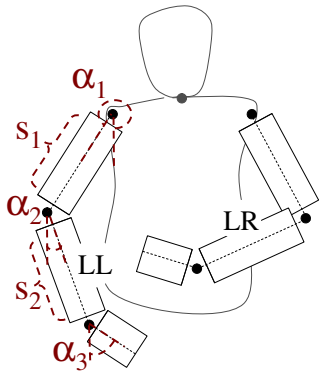


Figure 2.4: **Upper body model in [Buehler et al., 2011].** The pose is specified by 11 parameters – 5 for each arm and an additional binary parameter b indicating which arm is closer to the camera and hence visible in the case that the arms overlap.

In addition to the prior term $p(\boldsymbol{L})$ (described below), this cost function incorporates two appearance terms (that are learnt from manual annotation of a small number of training images) which model the agreement between the image $\boldsymbol{I}$ and configuration $\boldsymbol{L}$: colour and gradients (HOG). Both of these terms are learnt from a small number of manually labelled frames for each new input video (details below).

Colour term $p(\boldsymbol{c}_i|\lambda_i)$. This models the likelihood of pixel colours. Given the configuration $\boldsymbol{L}$, every pixel of the image is assigned a label which selects which part of the model is to explain that pixel (background, torso, arm, *etc.*). The labelling function is defined algorithmically by rendering the model (Figure 2.4) in back-to-front depth order (the 'painter's algorithm') such that occlusions are handled correctly.

HOG term $p(\boldsymbol{h}_j|\boldsymbol{l}_j)$. This models the likelihood of gradients. A HOG template for each part, scale and orientation is learnt from manually labelled images (for which the pose of the upper body is given). Separate templates are computed for every part, scale and orientation in order to capture illumination effects (and folds in clothing).

Prior term $p(\boldsymbol{L})$. This models the prior probability of configuration $\boldsymbol{L}$, placing plausible limits on the joint angles of the hands relative to the lower arms by enforcing the kinematic chain (in this method with manually specified constraints).

Computationally efficient solution for the cost function. The total number of possible arm configurations of the above method is $\approx 10^{13}$, which is infeasible to search by brute-force. To efficiently solve the cost function, [Buehler et al., 2011] proposed a sampling-based method where the arms are fitted sequentially. To further cut down the computational cost, this sampling-based approach is used only to identify frames where the pose can be estimated with high confidence (which the authors call 'distinctive frames'). The arm configurations in all non-distinctive frames are found by tracking over time (by adding a tracking term to the global cost function above), which is faster than using a sampling-based approach for each frame.

2.1.7 Case study 2: [Charles et al., 2013a]

We next review the random forest pose estimator from [Charles et al., 2013a]. We describe this random forest method here in detail, as it will be used in Chapter 4 in conjunction with a co-segmentation method to accurately estimate the pose in monocular RGB videos.

Overview. The method tackles pose estimation by treating it as a classification problem for each pixel in the input image. Each pixel in the image is classified into a discrete class label $l \in \{l_i\}$ (one for each body joint) in a sliding-window fashion.

The classification is performed by a random forest (an ensemble of T decision trees), where each tree t consists of split nodes which perform a true or false test on incoming pixels (the node tests are discussed below). As in any decision tree, at training and test time, pixels are recursively pushed down either the left or right branch depending upon the outcome of this test. When a pixel reaches a leaf at the bottom of the tree, a probability distribution $p_t(l|W_q, I)$ (learnt from training data as described below) assigns the pixel a probability for class label l (I is the input image, and W_q is the set of pixels in the window surrounding pixel q). The final conditional distribution $p(l|W_q, I)$ is obtained by taking an average across all trees in the forest.

Node tests. Inspired by [Shotton et al., 2008, 2011], this work uses very efficient test functions $f(.)$ at the nodes of the trees, which only compare pairs

of pixel values. A pixel q is represented by $\mathbf{x}_q = (x_q^1, x_q^2, x_q^3)$ where x_q^1, x_q^2, x_q^3 could be RGB values at pixel q; or skin, torso and background colour posterior values [Benfold and Reid, 2008]. The function f operates on a pair of pixels (a, b) from within the window W_q, and produces a scalar value which is compared to a learnt threshold value υ. Similar to [Shotton et al., 2008], these tests are restricted to take one of four forms: $f(a) = x_a^c$, or $f(a, b) = x_a^c - x_b^c$, or $f(a, b) = x_a^c + x_b^c$, or $f(a, b) = |x_a^c - x_b^c|$, where $c \in \{1, 2, 3\}$ indexes the colour channel.

Learning split node parameters. Each tree in the forest is trained recursively by randomly sampling a diverse set of points S_n (the 'dataset') for each node n from the training frames. The split function and threshold at each node are selected to split the data reaching that node as 'purely' as possible as measured using Gini impurity:

$$i(S_n) = 1 - \sum_l p(l|S_n)^2 \tag{2.3}$$

where $p(l|S_n)$ is a histogram of the dataset S_n over possible labels l at node n.

The parameters of split nodes are learnt by trying all possible test functions $f(.)$ and colour posterior types c for a randomly sampled offset pixel (a, b) inside the window W_q. This process is repeated k times (we use $k = 200$), and the set of parameters which maximises the drop in impurity is selected as the winning decision function. This process is repeated recursively for all nodes.

A node is declared a leaf node (*i.e.* not split further) when: (i) the maximum

depth limit D of the tree (where D is manually specified at training time) has been reached; or (ii) the node is 'pure' (*i.e.* all points reaching the node have the same class label). Finally, a per-leaf probability distribution $p_t(l|W_q)$ is stored at the leaf node (represented as a normalised histogram over the labels of all data points reaching the node).

Test-time pose estimation. At testing time, a location for the joint l is found by using the output $p(l|W_q, I)$ of the random forest, and estimating the density of joint proposals using a Parzen-window kernel density estimator with a Gaussian kernel. The position of maximum density is used as the joint estimate.

2.2 Gesture Recognition with Weak Supervision

This section reviews the existing methods for gesture recognition that are relevant to this thesis. Comprehensive reviews of gesture recognition in general are given in [Mitra and Acharya, 2007, Rautaray and Agrawal, 2015].

2.2.1 Overview

Gesture recognition has recently received significant attention thanks to Kinect, which showed that gesture recognition can be used as an effective user interface for games.

Effective gesture recognition requires some form of training data to learn the gestures from. The majority of approaches to gesture recognition so far have relied

on strongly supervised learning [Dreuw et al., 2006, Ong and Ranganath, 2005, Starner et al., 1998], where a large quantity of training data is ground truthed (typically to collect such a dataset, a person performs a sequence of gestures in laboratory conditions, followed by manual labelling of the gestures). This kind of approach is inherently expensive and does not scale to large, evolving gesture languages with high levels of variation. This has led to increased interest in two alternative approaches:

One-shot supervision. Several recent works have attempted to learn gestures at the other extreme – from a *single training example* per gesture (one-shot learning) [Guyon et al., 2013, Ke et al., 2007]. These methods generally use a combination of temporal segmentation (normally dynamic time warping) plus some spatiotemporal features (extracted either from the skeleton or depth frames) to find the closest matching sample in the training set. However, given the vast variability in how gestures are performed, and the variation in people and camera viewpoints, learning accurate, generalisable models with so little supervision is somewhat challenging, to say the least.

Weak supervision. Another avenue of work has explored learning gestures from practically infinite sources of data with weak supervision [Buehler et al., 2009, Cooper and Bowden, 2009, Kelly et al., 2011, Nayak et al., 2012], *e.g.* TV broadcasts with aligned subtitles (or similarly actions from movies with aligned transcripts [Bojanowski et al., 2013, Duchenne et al., 2009]). These works typically employ a multiple instance learning-based approach, where temporal se-

quences with a weak label are grouped together, and the sub-sequences that occur most frequently in the positively labelled 'bags' (and most infrequently in the negatively labelled bags) are selected. While these works have shown promise, they have run into some limitations of the weak supervision available for gestures today: it is so weak and noisy that it is very difficult to learn from it alone.

In this section, we first briefly discuss the traditional gesture recognition approaches that employ strong supervision. We then discuss in detail methods that exploit weak supervision, as these are particularly relevant to this thesis. Finally, we discuss approaches specific to recognising *sign language* gestures, which differ from most other gesture languages in that it uses*multiple modalities* (face, hand movement, gesture order *etc.*) to convey gestures.

2.2.2 Learning gestures from strong supervision

The majority of research on gesture recognition to date relies on manually annotated (strongly supervised) datasets. While this learning approach may not be scalable, many of the gesture matching methods discussed in this section can also be used in a weakly supervised setting.

Early work. In the earliest gesture recognition methods, heavy constraints were typically imposed, such as wearing motion sensors [Chunli et al., 2002] or using a uniform background and/or wearing coloured gloves. Hidden Markov Models (HMMs) [Chunli et al., 2002, Starner et al., 1998, Yamato et al., 1992] were particularly popular. [Starner and Pentland, 1997, Starner et al., 1998]

showcased a system for real-time sign language recognition from RGB (with hand tracking) using a wearable computer, which inspired many other papers tackling gesture recognition with HMMs [Campbell et al., 1996, Chunli et al., 2002, Liang and Ouhyoung, 1998, Vogler and Metaxas, 1998, 2004]. [Bowden et al., 2004, Kadir et al., 2004] experimented on colour glove data, and showed that it is possible to learn an accurate person-specific gesture recognition model from a very small amount of training data.

One-shot learning. In one-shot learning, initial work focused on nearest neighbour matching by shape [Ke et al., 2007]. [Farhadi et al., 2007] showed that one can learn American sign language gestures from an avatar dictionary (with a single example for each gesture), and generalise to realistic sign language videos using a transfer learning approach. Work on the ChaLearn one-shot gesture learning challenges [Guyon et al., 2012, 2013] recently spurred a burst of interest into learning gestures from one-shot learning. The best-performing methods [Fanello et al., 2013, Krishnan and Sarkar, 2013, Wan et al., 2013] generally use dynamic time warping to match to the nearest gesture in the training dataset, either based on pose estimation output, or a (spatial or spatiotemporal) descriptor (normally HOG or optical flow) of the raw RGB / depth frames.

Depth data and Kinect. Depth data, particularly from Kinect, has recently spurred increased interest in gesture recognition. The benefits of using data from Kinect are twofold: first, the depth data helps (somewhat) in detecting occluding limbs and recovering the 3D information in gestures; second, it comes

with (somewhat noisy) pose predictions.

Several methods using Kinect data have emerged (in addition to ChaLearn methods above), including: (i) [Ershaed et al., 2011] recognised isolated Arabic sign language signs; (ii) [Zafrulla et al., 2011] investigated the use of Kinect for sign language recognition in educational games for deaf children (and proposed a sign language-based game called 'CopyCat' that matches based on Kinect-provided joints); (iii) [Pugeault and Bowden, 2011] used Kinect for tracking the hand positions and doing real-time sign language fingerspelling recognition by matching hand shape features with a random forest; (iv) [Cooper et al., 2012] presented two new Kinect datasets for Greek Sign Language and German Sign Language and used them to learn gesture sub-units; and (v) [Chai et al., 2013, Lin et al., 2014] proposed a method to translate Chinese sign language to text in real-time (which received world-wide press recently). The method matches 3D hand trajectories from Kinect and combines these with a simple sign language language model. While impressive performance was reported (rank-1 83.51% for 239 words), it is unclear whether their method required training on labelled gesture examples for the test signer, which would make the method less useful in practice.

We note that this thesis focuses on learning gestures from weakly supervised data in TV broadcasts (which are pure RGB – no depth), so we do not use depth data in this thesis. However, we have an ongoing project where we record Kinect data in TV sign language studios which we discuss in more detail in the future work section of Chapter 9.

2.2.3 Learning gestures from weak supervision

In contrast to the strongly supervised approaches, in a weakly supervised learning, no manual annotation is performed – instead, existing weak supervision is used.

The most prominent example of using weak supervision for gesture recognition is subtitled sign language TV broadcasts. Here, the idea is that words are likely to occur in the subtitles sometime near the time that they are signed, so the subtitles can be used as weak supervision to learn the signs. However, the supervision is both weak and noisy: it is weak because of the bad temporal alignment between the subtitles and the signing, and is noisy because the sign might not occur when the subtitle does.

The most cited works using this approach are [Buehler et al., 2009] and [Cooper and Bowden, 2009], who both automatically extracted sign-video pairs from TV broadcasts. We next review these two methods:

[Buehler et al., 2009]. This work used 10.5 hours of signing sequences with subtitles, recorded from TV broadcasts. Given a subtitle word, the subtitle supervision is used to split the data into 'positive' sequences (in which a word is likely to occur) and 'negative' sequences (in which the word is known to be unlikely to occur). A distance function between two temporal windows is defined (based on hand position, hand shape and hand orientation, obtained from the pose estimator in Section 2.1.6), and is used within a multiple instance learning method to find the signs that occur most frequently within the positive sequences (but do not occur in the negative sequences). In later work, these automatically

extracted signs were used to train a signer-independent sign classifier [Buehler et al., 2010]. While the method performs well, it relies on performing a computationally expensive brute force search over all temporal windows. In Chapter 7, we address this by using mouth motion to cut down the search space.

[Cooper and Bowden, 2009]. This work used a temporally constrained adaptation of apriori data mining on hand and head positions (again obtained with the method in Section 2.1.6) to learn signs. Similar sections of each video block are mined, and the responses are used to show where a sign is likely to be located. This method is very similar in spirit to [Buehler et al., 2009] (given a word, using positive and negative sequences to find the most likely sign), but differs in three main aspects: (i) apriori mining is used instead of MIL; (ii) no appearance-based features (such as hand shape) are employed; and (iii) experiments are only conducted on a 30min long video of a single signer (vs 10.5h of video for 3 signers in [Buehler et al., 2008]).

Other works using weak supervision for learning gestures include:

[Farhadi and Forsyth, 2006]. This work considered the problem of aligning an American Sign Language sign with an English text subtitle, but under much stronger supervisory conditions than the above two approaches. This work used one 1 hour long sign language-interpreted children's film consisting of a series of blocks of video (where spoken narration, signing and subtitles occur), interspersed with video where there is neither dialogue nor narration. This means that the video blocks and subtitle blocks are already very closely aligned (unlike in TV

broadcasts, where the signing and subtitles are much more out of sync), which enabled the use of a much simpler method than the above two methods. After identifying hand positions with a skin detector they describe the frames with SIFT features centred on head and hands. Finally, HMMs are used with these features to estimate the start and end positions of each sign.

[Kelly et al., 2011]. This work built upon the ideas of [Buehler et al., 2009, Cooper and Bowden, 2009]. However, the data differs slightly. Instead of using TV broadcasts, this work recorded a dataset of two Irish sign language signers in a studio, and interpreted their signing into 844 text sentences. Similar to TV broadcasts, the supervision here is weak: the input is a set of video sequences and their 'subtitles' (from interpreters). The task was to learn the signs automatically (without temporal information w.r.t. *when* the sign occurs). However, in contrast to TV broadcasts, this data is both less weak and less noisy (because the temporal correspondence between the signed and 'subtitles' are tighter – more like that in [Farhadi and Forsyth, 2006]). Similar to [Buehler et al., 2009], they used this weak supervision to automatically extract signs using multiple instance learning. Given the extracted signs, they train an HMM based on geometric features computed from hand positions, which they use to recognise signs in continuous signing video (in a signer-dependent setting).

[Nayak et al., 2012]. This work used a Bayesian framework for extracting the common subsequences for a set of weakly labelled sign language sequences. The dataset they use is similar to that in [Kelly et al., 2011]: a set of videos with

sentence-level translations to English (here with 155 signs recorded for a single person). The automatically extracted examples are then used to localise signs in test sequences (again in a signer-dependent setting).

2.2.4 Sign language: beyond hand gestures

In this short section, we give a brief overview of work on other modalities (beyond the hand gesture-based works discussed above) that are involved in sign language recognition. A comprehensive review of sign language recognition methods is provided by [Ong and Ranganath, 2005].

Non-manual features for sign language recognition. In addition to manual features (hand movement), sign language also consists of non-manual features: lip patterns, facial expressions and body pose.

Lip patterns are used by most signers to mouth the word they are signing the same time they perform the manual gesture. These lip patterns resolve ambiguities in some signs that can be disambiguated solely by the lip shape (*e.g.* uncle and aunt in BSL). Lip reading is already an established field with multiple computer vision [Ong and Bowden, 2008, Zhou et al., 2011]. However, until the work in this thesis, no work had explored using lip patterns for sign language recognition (now, inspired by the work in this thesis,[Koller et al., 2014] and others have explored this further).

Facial expressions have seen the most rapid increase in use in sign language recognition [Aran et al., 2009, Vogler and Goldenstein, 2005]. They can be used

to modify any sign language word – they are essentially the tone of voice for sign language. [von Agris et al., 2008] showed that roughly 20% of the signs in their dataset can be recognised purely from facial expressions. [Nguyen and Ranganath, 2008] recognised four commonly occurring facial expressions (grammatical markers) in isolated American Sign Language signs, and in later work [Nguyen and Ranganath, 2010] extended their method to cope with continuous facial gestures affected by coarticulation.

In this thesis, the use of non-manual features is explored in Chapter 7, where lip patterns are used to narrow down the search space when learning signs from weak supervision.

Chapter 3

Datasets

In this chapter we describe the datasets used in this work and how they have been generated.

We first describe the main datasets used for pose estimation, and then describe the datasets used for learning gestures from weak supervision.

3.1 Pose Estimation Datasets

This section introduces three pose estimation datasets – BBC Pose, its extended version (Extended BBC Pose), and Poses in the Wild. These three datasets are used in Part I of this thesis. Table 3.1 shows a summary of these datasets.

	BBC Pose	Extended BBC Pose	Poses in the Wild
Total videos	20	92	30
Train videos	10	82 (10 same)	-
Val videos	5	5 (same)	-
Test videos	5	5 (same)	30
People	9	≈40	-
Frames	1.5M	7M	830
Train labels	Buehler et al.	Buehler et al. (10) + Chapter 4 (72)	-
Val labels	1,000 manual GT	1,000 manual GT (same)	-
Test labels	1,000 manual GT	1,000 manual GT (same)	830 manual GT

Table 3.1: **Summary of statistics for pose estimation datasets.**

3.1.1 BBC Pose dataset

This dataset consists of 20 TV broadcast videos overlaid with a person interpreting what is being spoken into sign language. The videos, each between 0.5h–1h in length (45K–90K frames), contain content from a variety of TV programmes. All videos have pose annotations from a semi-automatic pose estimator [Buehler et al., 2011]. Figure 3.1 shows example frames from one video, and Figure 3.2 shows an example frame from each of the videos in this dataset.

Semi-automatic generation of training labels. All frames of the videos have been automatically assigned joint locations (which we use as ground truth for training) using a semi-automatic and slow (but reliable) tracker by [Buehler et al., 2011] (described in more detail in Section 2.1.6). It is semi-automatic because it requires manual labelling of 64 frames per video (which is around three hours of manual user input per one hour of TV footage). It is slow because it is based on an expensive pictorial structure model (which requires hundreds of seconds computation per frame).

Figure 3.1: **Example frames from one video in the BBC Pose dataset.**

Split into training/validation/testing sets. The 20 videos are split into 3 disjoint sets: 10 videos for training, 5 for validation and 5 for testing (as shown in Figure 3.2). The dataset contains 9 signers. Of the 9 signers, the training and validation sets contain 5, and the testing set contains another 4. Splitting the data this way maintains enough diversity for training but also ensures fairness as the testing set contains completely different signers than the training and validation sets.

Sampling training/validation/testing data. The methods are tested and validated on frames sampled from each video. 200 frames containing a diverse range of poses are sampled by clustering the signers' poses (with k-means, $K = 100$) with data provided by Buehler *et al.*'s tracker, and uniformly sampling frames across clusters from each of the 5+5 validation/test videos (2,000 frames in total). This ensures the accuracy of joint estimates are not biased towards poses which occur more frequently, *e.g.* "resting" poses between signs. In Chapters 4 and 5, the methods are furthermore *trained* on sampled poses, which increases the diversity of poses in the training set.

Figure 3.2: **Visualisation of the BBC Pose dataset, showing one example frame per video.** Videos are split into training, validation and testing sets. Variation in terms of signer identity, clothing and background video content is ensured in the training set by using different videos and only duplicating signers if they are wearing different clothing. The testing set contains completely different signers than those present in the training or validation sets. A scale bar is provided in the top left hand corner image to compare pixel distance with signer size.

Ground truth labelling. For the validation and test videos, the 200 sampled frames (with diverse poses) from each video are manually annotated with joint locations.

Evaluation measure. In all pose estimation experiments we evaluate the performance by comparing estimated joints against frames with manual ground truth. An estimated joint is deemed correctly located if it is within a set distance of d pixels from a marked joint centre. Accuracy is measured as the percentage of correctly estimated joints. A scale superimposed on the top left frame in Figure 3.2 shows how pixel distance relates to signer size.

Pose visualisation. Figure 3.3 shows a scatter plot of stickmen [Tran and Forsyth, 2010], illustrating upper and lower arm placements for every frame in the training, validation and testing sets (poses are normalised to the mid-point between shoulders). Poses in testing frames cover a similar space of poses as in training frames.

3.1.2 Extended BBC Pose dataset

We also generate an *extended* version of the above BBC Pose dataset, with an order of magnitude more training data. This dataset contains all videos from the BBC Pose dataset plus 72 additional training videos. Combined with the original BBC TV dataset, the dataset contains 92 videos (82 training, 5 validation and 5 testing), *i.e.* around 7 million frames. The frames of the new 72 videos are

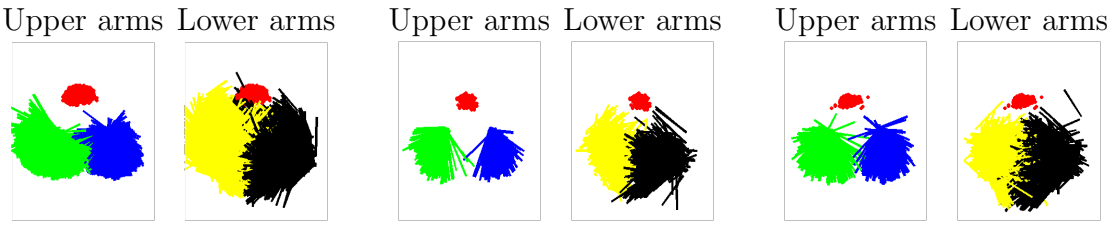


Figure 3.3: **Scatter plots of stickmen in the BBC Pose dataset**, showing plots of upper and lower arm placements for every frame in the training, validation and testing sets. Poses are normalised to the mid-point between shoulders. Head centre points are rendered as red dots; right/left *upper* arms are shown as green and blue lines respectively; and right/left *lower* arms are shown as yellow and black lines respectively. Poses are not scale-normalised, meaning scale and location variation is directly observable between sets.

automatically assigned joint locations (used as ground truth for training) with the tracker of Chapter 4. In practice, these 'ground truth' joint locations are slightly noisier than those in the original BBC Pose dataset (which were obtained using the slow, semi-automatic tracker of [Buehler et al., 2011]).

3.1.3 Poses in the Wild dataset

The Poses in the Wild dataset [Cherian et al., 2014] contains 30 sequences (total 830 frames) extracted from Hollywood movies. The frames are annotated with upper-body poses. As shown in Figure 3.4, it contains realistic poses in indoor and outdoor scenes, with background clutter, severe camera motion and occlusions. For training, we follow [Cherian et al., 2014] and use all the images annotated with upper-body parts (about 4.5K) in the FLIC dataset [Sapp and Taskar, 2013].

Figure 3.4: **Example frames from Poses in the Wild.**

3.2 Gesture and Sign Language Recognition Datasets

This section introduces the datasets used for gesture and sign language recognition – the Sign Extraction dataset, the Extended Sign Extraction dataset, two BSL dictionaries (BSL dictionary 1 and BSL dictionary 2), and ChaLearn. These are used in Part II of this thesis. Table 3.2 shows a summary of these datasets.

3.2.1 Sign Extraction dataset

The sign extraction dataset consists of 35 high-definition (HD) TV broadcast videos, with 17 different signers, and in total 30 hours of data. Each video contains between 40K and 85K frames of sign-interpreted video content from a variety of BBC TV programmes. All frames of the videos are automatically assigned segmentations, joint labels and mouthing scores using the methods described in Chapter 4 and Chapter 7. Samples frames of this dataset are shown

	Sign Extraction	Ext Sign Extraction	Dict 1	Dict 2	ChaLearn
Videos	35	130 (95 new)	3970	3409	955
Hours of video	30	155	2.5	3	23
Signers	17	≈50	1	6	27
Supervision	Weak (subtitles)	Weak (subtitles)	Strong	Strong	Strong
Manual GT	41 words	41 words (same)	N/A	N/A	N/A

Table 3.2: **Summary of statistics for gesture recognition datasets.** 'Ext' stands for Extended, 'Dict' for Dictionary, and 'GT' for ground truth.

in Figure 3.5.

Supervision (weak and noisy). We generate supervision for this dataset from the TV broadcast subtitles (which show what words are spoken in the TV broadcast). However, this supervision is both a weak and noisy signal for when (and whether) a sign occurs in the TV broadcast. It is weak as the subtitles are not temporally aligned with the signs – a sign (typically 8–15 frames long) could be anywhere in the overlapping subtitle video sequences (typically 400 frames). It is noisy as the occurrence of a word in the subtitle does not always imply that the word is signed (typically the word is signed only in 20–60% of the subtitle sequences). A description of the data generation is given below.

Manual ground truth. A set of 41 subtitle words (animal, antique, asian, bank, beacon, bear, beautiful, beef, bike, blood, buy, Chinese, chocolate, epigenome, fake, feel, gram, heart, heat, industry, jelly, jewish, kill, market, milk, pay, reindeer, rugby, school, science, sell, simple, snow, song, sound, target, vision, war, winter, work, year) is selected at random from the 1,000 most frequently occurring words, and for these the ground truth sign temporal windows are annotated. Table 3.3 shows the number of 'positive' subtitle sequences for each word, and

Figure 3.5: **Sample frames from the Sign Extraction dataset.**

the actual number of occurrences of the signs corresponding to the subtitle word.

Data generation. The subtitles are first stemmed, and stop words are removed. Then, given a word, the subtitles define a set of subtitle sequences in which the word occurs (8–40 sequences depending on how many times the word occurs), each around 15s (approx 400 frames) long. These subtitle sequences are used as the weak supervision.

3.2.2 Extended Sign Extraction dataset

We extend the Sign Extraction dataset with 135 hours of additional data, yielding a total of 155 hours of TV broadcasts. This dataset is otherwise like the Sign Extraction dataset (same weak supervision and data preprocessing), Upper

Sign	#pos	#GT
animal	12	9
antique	15	8
asian	11	9
bank	21	10
beacon	28	21
bear	14	10
beautiful	12	5
beef	15	8
bike	15	6
blood	15	4
buy	13	3
Chinese	33	11
chocolate	20	6
epigenome	20	13
fake	14	11
feel	12	8
gram	25	12
heart	14	5
heat	15	5
industry	19	9
jelly	11	1

Sign	#pos	#GT
jewish	27	14
kill	17	9
market	16	12
milk	10	6
pay	37	25
reindeer	15	9
rugby	11	7
school	13	5
science	18	10
sell	15	5
simple	12	10
snow	29	11
song	19	10
sound	26	5
target	23	4
vision	17	10
war	11	6
winter	23	12
work	22	14
year	32	8

Table 3.3: **Ground truth for the Sign Extraction and Extended Sign Extraction datasets.** #pos is the number of positive subtitle sequences for the word in the same row. #GT is the number of times the sign actually occurs in the positive subtitle sequences. For this test set, the ratio between ground truth occurrences and 'positive' subtitle sequences is 0.46 due to the noisy supervision.

body joint tracks are obtained automatically for this dataset using the method described in Chapter 4.

3.2.3 Two BSL dictionary datasets

These datasets are video dictionaries for British Sign Language, which we call 'Signstation' (BSL dictionary 1) [Bristol Centre for Deaf Studies, 2014] and 'Standard BSL dictionary' (BSL dictionary 2) [BSL, 2005]. Samples frames from the datasets are shown in Figure 3.6(a–b). The first contains 3,970 videos (total 2.5 hours), one for each word; and the second contains 3,409 videos (total 3 hours),

Figure 3.6: **Sample frames from three datasets with upper body pose estimates overlaid.** (a) BSL dictionary 1 (Signstation); (b) BSL dictionary 2 (Standard BSL dictionary); and (c) ChaLearn.

and covers 1,771 words (the majority of words signed in one or more regional variation). BSL dictionary 1 contains a single signer, whereas BSL dictionary 2 contains multiple signers and multiple regional variations. There is no overlap of signers between the two dictionaries. The two datasets contain different sets of gestures, and intersect (*i.e.* have common words) only for a subset of these.

Data generation. In order to effectively use this data in Chapter 8 (as training and testing material for the same set of signs), it is first necessary to find the 'pairs' of gestures that are the same in the two dictionaries. This is made difficult

by the fact that the dictionaries contain different regional variations of the same gestures (*i.e.*, we cannot simply assume gestures with the same English word label are the same). We therefore need to look for visual similarity as well as the same English word label. We automatically find a subset of words pairs of the same gesture performed the same way by computing a time-and-space aligned distance (see Section 8.2.2) from upper body joint positions (obtained with the method in Chapter 4) for all gesture pairs of the same word, selecting pairs with distance below a threshold (set from a small manually labelled set of pairs). This list of pairs is manually verified and any false matches (mainly due to incorrect pose estimates) are filtered away. This results in 500 signs in common between the two dictionaries.

3.2.4 ChaLearn gesture dataset

This dataset is the ChaLearn 2013 Multi-modal gesture dataset [Escalera et al., 2013], which contains 23 hours of Kinect data of 27 persons performing 20 Italian gestures. The data includes RGB, depth, foreground segmentations and Kinect skeletons. The data is split into training, validation and testing sets, with in total 955 videos each lasting 1–2min and containing 8–20 non-continuous gestures. The large variation in clothing across videos poses a challenging task for pose estimation methods. Samples frames from this dataset are shown in Figure 3.6(c).

Part I

Pose Estimation in Videos

Chapter 4

Pose Estimation with Random Forests

This chapter presents a pose estimator based on Random Forests that detects joint positions (of hands, arms, shoulders and head) in RGB videos of more than an hour in length, without any manual annotation.

We make contributions in three areas:

1. Foreground/background segmentation: We show that for sign language TV broadcasts introduced in Section 3, the overlaid person can be separated from the background TV broadcast using co-segmentation over all frames with a layered model (Section 4.1).

2. Random forest regressor trained from semi-automatically labelled data: Given the segmentation of the person, we show that the joint positions

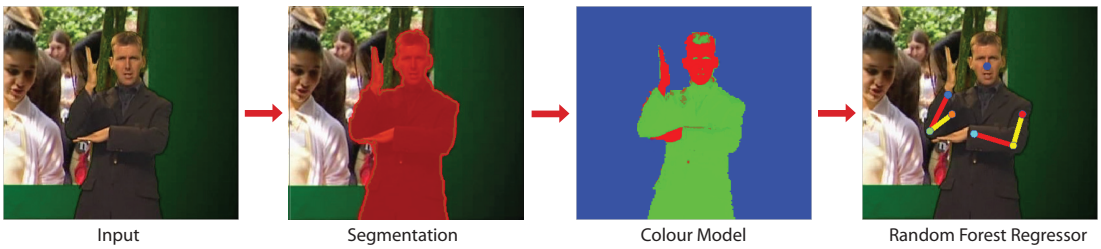


Figure 4.1: **Random forest pose estimator overview.** Arm and hand joint positions are predicted by first segmenting the signer using a layered foreground/background model, and then feeding the segmentation together with a colour model into a random forest regressor.

(shoulders, elbows, wrists) can be predicted per-frame using a random forest regressor, trained from an existing semi-automatic, but computationally expensive, tracker, with no manual annotation (Section 4.2). This is similar in spirit to the work by [Shotton et al., 2011] who also regress joint positions from (depth) images – however, our approach works for RGB frames.

3. Pose evaluator: We introduce an evaluator to assess whether the predicted joint positions are correct (Section 4.3).

The method is applied to the BBC Pose dataset described in Section 3.1.1 with changing background, challenging imaging conditions, and for different signers. This material is challenging to segment and determine human joint positions on for a number of reasons (shown in Figure 1.3) that include: self-occlusion of the signer, self-shadowing, motion blur due to the speed of the signing, and, in particular, the changing background (since the signer is superimposed over a moving video that frequently even contains other people.

This method outperforms the long term tracker [Buehler et al., 2011] (described

in detail Chapter 2.1.6), does not require the manual annotation of that work, and, after automatic initialisation, performs tracking in real-time. It also achieves superior joint localisation results to those of [Yang and Ramanan, 2011], which at the time the work was conducted was the state-of-the-art pose estimator.

Figure 4.1 shows an overview of the processing steps of this method. The input frame is first segmented using a co-segmentation method; this is then turned into a skin/torso/background *colour posterior* that is used for training a Random Forest Regressor. We next discuss each of these steps in more detail.

4.1 Co-segmentation Algorithm

The goal of the co-segmentation algorithm is to segment the overlaid signer from each frame of the broadcast. We exploit the fact that sign language broadcasts consist of an explicit layered model as illustrated in Figure 4.2. In the spirit of a generative model, *i.e.* one that generates the image by composition, we exploit these inherent layers to provide an accurate segmentation of the signer. We describe the three layers in the following paragraphs.

The static background layer (SBG) essentially consists of the framing (around the actual/original broadcast) that has been added by the studio. As can be seen in Figure 4.3, the static background is partially revealed and partially occluded in each frame depending on the position of the signer. In a similar manner to how a "clean plate" is constructed in film post-production, by looking through the whole video and combining the partially revealed static backgrounds one can

automatically, and almost fully, reconstruct the actual static background. This layer can then be exploited when segmenting the signer.

The dynamic background layer (DBG) consists of a fixed rectangle, where the original video is displayed, but is always partially covered by the signer and changes from one frame to another. Its colour information, for the region where it does not overlap a bounding box on the signer, is modelled separately and forms a background distribution for a subsequent segmentation of the signer.

Finally, the foreground layer (FG) consists of the moving signer. By assuming that the colour distribution of the signer remains constant we can build an accurate foreground colour model for the whole video.

4.1.1 Algorithm overview

The input to the co-segmentation algorithm is a signed TV broadcast video, and the output is a foreground segmentation, a quality score for the segmentation, the head position and a colour model for the skin and torso. These will be used in the random forest regressor.

The algorithm consists of two main steps:

1. Automatic initialisation (per image sequence). To exploit the inherent layered model we initialise the algorithm by first determining the "clean plate", the dynamic rectangle and the foreground colour model. The details of how this 'initialisation set' is obtained are given in Section 4.1.2.

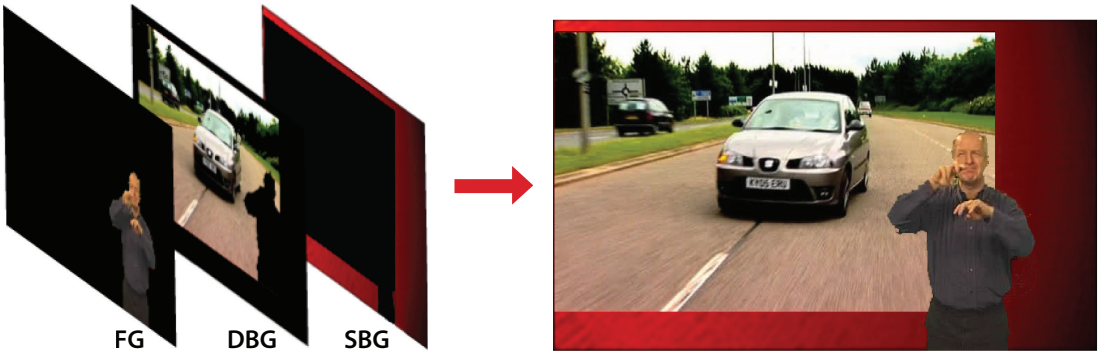


Figure 4.2: **Generative layered model of each frame.** The co-segmentation algorithm separates the signer from any signed TV broadcast by building a layered model consisting of a foreground (FG), dynamic background (DBG) and static background (SBG).

2. Segmentation with a layered model and area constraints (per frame). The initialisation set is then used to derive an accurate hard segmentation of the signer in each frame. The clean plate and an area constraint are used to refine an initial rough segmentation. The details of this method are given in Section 4.1.3.

4.1.2 Co-segmentation initialisation

In this section we describe how, given an input sign language video, we can obtain the layers and their layout that are common to the video sequence (in order to enable the subsequent per-frame segmentation). In particular, we wish to obtain the regions shown in Figure 4.3, as well as the foreground colour distribution. Our approach is to treat each frame as being generated from a number of layers, as depicted in Figure 4.2, and to thereby solve for the layers and layout. This problem differs from typical applications of generative layered models for video,

e.g. [Jojic and Frey, 2001, Kumar et al., 2008], since part of the background in the video is always moving so we have a dynamic rather than fixed layer. The creation of the layered model can be broken down into a step per layer:

Dynamic background. The aim in this step is to find the rectangle that contains the dynamic background, and furthermore divide it into a region where the signer may overlap, and another where the signer never reaches (see Figure 4.3c). The latter region will be used to define a per-frame background colour. To this end we find pixels that change intensity values for the majority of frames and compute their rectangular bounding box, as shown in Figure 4.3b. This also yields an area that is permanently static throughout the video (region A in the same figure) that we use as a permanent BG clamping region. Regions A and B in the same figure, which the signer never reaches, are defined relative to the position of the signer's face (the face detection method is described below).

Static background. The aim here is to find the static background, which can be viewed as consisting of a 'clean plate' (term explained above). Once we have this 'clean plate', we can then say with near-certainty whether a pixel belongs to the FG or BG. The clean plate is obtained by roughly segmenting a random set of frames into FG (signer) and BG using a graph cut algorithm. The regions used to obtain the FG and BG distributions are illustrated in Figure 4.3c. In particular, areas selected relative to the position of the signer's face (face detection method described below) are used to initialise the FG colour distribution. Given these segmentations, the clean plate is obtained as a median over the BG.

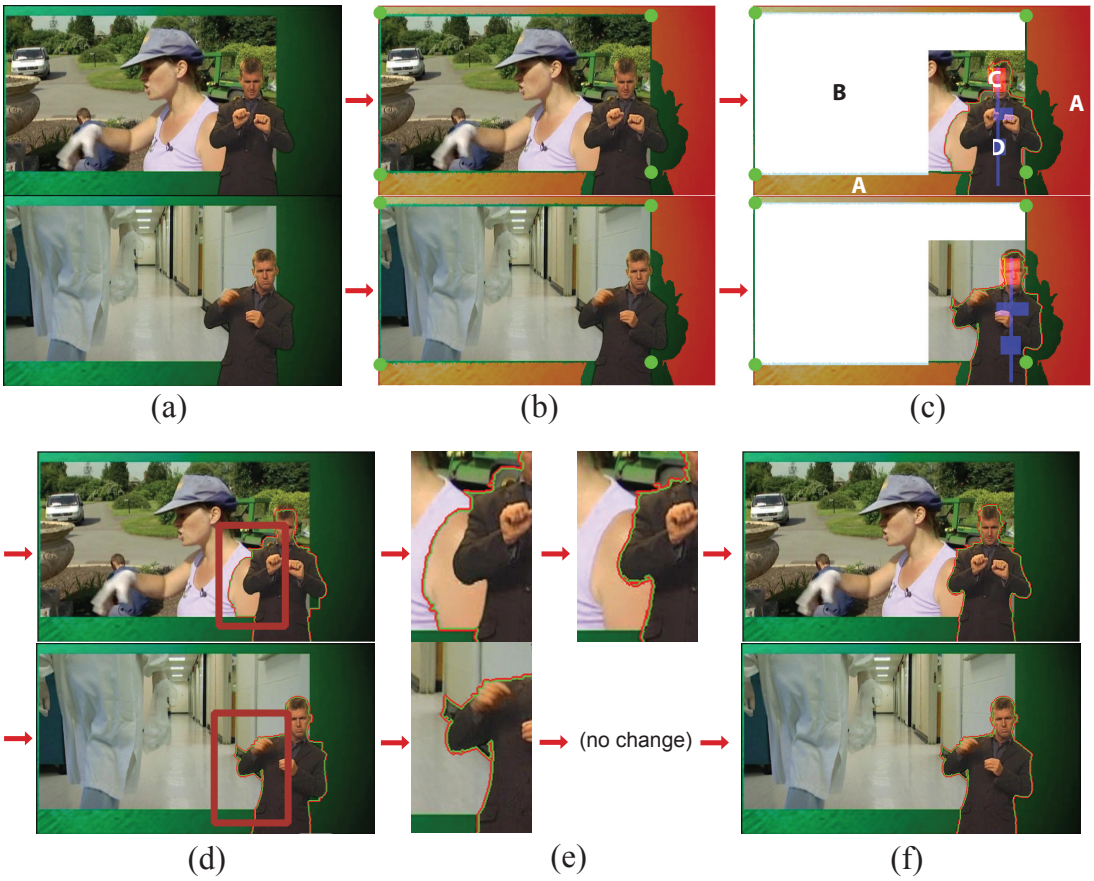


Figure 4.3: **Co-segmentation method.** (a) original frames; (b) dynamic layer (rectangle spanned by the green dots) and the permanently fixed background (in red) – the remaining green area behind the signer is the backdrop which is not part of the fixed background; (c) rough segmentation with clamping regions for running graph cut, where: A is the permanently fixed background; B is the clamping region for the dynamic background; C is part of the foreground colour model; and D is a hard foreground clamp (based on the position of the detected face). (d) initial GrabCut segmentation that uses colour distributions of A, B for background and C, D for foreground; (e) detail of the red rectangular region of (d) showing the segmentation refinement stage (see text); and (f) segmentation after clean plate and area size refinements.

Foreground colour model. Here the aim is to obtain the signer colour distribution (which is assumed approximately constant throughout the sequence). This removes the need for finding accurate FG colour models for individual frames. The colour distribution (which is represented by a histogram) is obtained from the rough FG segmentations (see Figure 4.3c, computation described above) using frames where the colour histograms of the FG and dynamic background differ the most. The high colour difference increases the likelihood that there is a high contrast between the FG and BG and thus that the segmentation is correct.

Face detection. Face detection is used for initialisation and frame-by-frame segmentation. Detection of both frontal and profile view faces is achieved by choosing between the face detector by [Zhu and Ramanan, 2012] (high recall for frontal faces) and a face detector based on upper body detection [Ferrari et al., 2008] (lower recall but detects profile views) according to their confidence values.

4.1.3 Per-frame segmentation with a layered model and area constraints

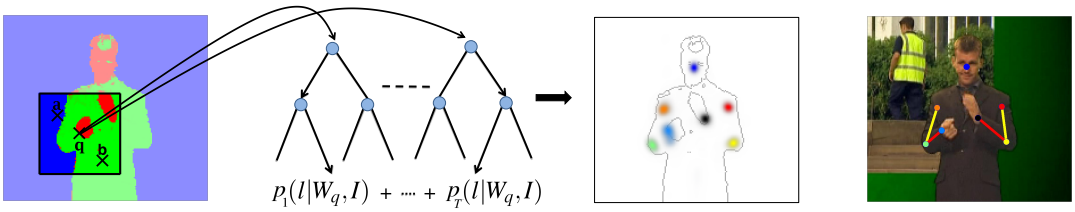
Having finished the initialisation step we now have a layered model that can be used to derive a segmentation of the signer. This layered model (the 'initialisation set') is used to: (i) improve the segmentation by comparing each pixel against the clean plate (to yield a near-certain segmentation label as the background is known); and (ii) shrink the foreground segmentation size if it is too big (to avoid catching *e.g.* skin regions in the background).

The segmentation uses GrabCut [Rother et al., 2004], with the FG colour model provided by the initialisation set and, as in [Ferrari et al., 2008], with the FG clamped in areas based on the face location (Figure 4.3c). The BG colour distribution is known from the dynamic background. The segmentation is refined twice: first by comparing pixels to the clean plate of the static background, and then by shrinking the foreground size if it is much bigger than the average size. The latter is done by adding a constant to the graph cut unary potentials of the dynamic background (this increases the likelihood that a larger part of the dynamic background is labelled as BG, hence reducing the size of the FG). This addresses a common failure case where the dynamic background contains a colour similar to the signer, which leads to the foreground region 'catching' part of the dynamic background and becoming too large. In contrast, the foreground is seldom too small thanks to good FG colour model estimates. Examples of fully refined segmentations are shown in Figure 4.3e.

The segmentation still fails in certain difficult cases, *e.g.* when the colours of the FG and BG are very similar or when the face detector fails. We fix this by computing a segmentation quality score as described in Section 4.3.

4.1.4 Colour posterior

At this stage we have a foreground segmentation that is rated by a segmentation quality score. However, additional layout information is also available from the the spatial position of the the skin and torso (*i.e.* non-skin) pixels. The posterior



(a) Color model (b) Random forest (c) PDF of joints (d) Estimated joints

Figure 4.4: **Estimating joint positions.** (a) input colour model image; (b) random forest classifies each pixel using a sliding window and learnt test functions; (c) probability density function of each joint location, shown in different colours per joint (more intense colour implies higher probability); (d) joint estimates, shown as small circles linked by a skeleton.

probability of the skin and torso pixels is obtained from a colour model. Computing the colour *posteriors* for skin and torso abstracts away from the original colour, of the clothes for example, which varies between signers and is not directly informative [Benfold and Reid, 2008, Jojic and Caspi, 2004].

In a similar manner to the construction of the initialisation set for the layers, the skin colour distribution is obtained from a patch of the face over several frames, and the torso colour distribution is obtained from a set of FG segmentations from which the colours of the face/skin are automatically removed. These colour distributions are then used to obtain a pixel-wise posterior for the skin and torso in each frame.

4.1.5 Technical details

Here we provide the additional details for the segmentation method. The dynamic background is determined using a subset of 300 uniformly sampled frames for each video. Earth Mover's Distance (EMD) is used to compare colour histograms for

extracting the foreground colour model and for generating colour posteriors (to remove skin regions from the FG segmentations). Faces are detected in the right half of the image for computational efficiency. The maximum foreground segmentation size is set to a standard deviation above the median segmentation size over all frames in a video.

4.1.6 Related work for co-segmentation

Co-segmentation methods [Hochbaum and Singh, 2009, Joulin et al., 2010, Rother et al., 2006] consider sets of images where the appearance of foreground and/or background share some similarities, and exploit these similarities to obtain accurate foreground-background segmentations. [Rother et al., 2006] originally introduced the problem of co-segmenting image pairs. Their approach was to minimise an energy function with an additional histogram matching term that forces foreground histograms of images to be similar. [Chai et al., 2011] proposed co-segmentation algorithms that work on each image category separately, and embed class-discriminative information into the co-segmentation process. In our case our co-segmentation algorithm automatically separates signers from any signed TV broadcast by building a layered model [Jojic and Frey, 2001, Kumar et al., 2008, Szeliski et al., 2000].

4.2 Random Forest Pose Estimator

In this section we describe how the foreground segmentation (from co-segmentation) and the skin and torso colour posterior (from the colour model) are fed into a Random Forest for training (and testing) and pose estimator.

We cast the task of localising upper body arm joints and head position as a multi-class classification problem, classifying each image pixel into one of 8 categories $l \in \{$head centre, left/right wrist, left/right elbow, left/right shoulder, other$\}$ using a random forest classifier in a sliding-window fashion. We also refer to the 'head centre' as a joint (see Figure 4.4d). As shown in Figure 4.4a, the input to the random forest comes from the colour model image after co-segmentation. The joints are localised on a per-frame basis.

The random forest classifier uses simple features to make classification extremely computationally efficient. Classification to a discrete class label $l \in \{l_i\}$, for each pixel q across the image, is performed in a sliding-window fashion. The method follows that of [Charles et al., 2013a], described in Section 2.1.7.

At testing time, a location for the joint l is found by using the output of the random forest and estimating the density of joint proposals using a Parzen-window kernel density estimator with a Gaussian kernel. The position of maximum density is used as the joint estimate.

4.3 Pose Evaluator

At this point our joint predictor outputs joint estimates for each frame of the video. However, the predictions are provided 'as is', without an indication of whether they are correct or not. Therefore, in the spirit of [Jammalamadaka et al., 2012] we train an evaluator that indicates whether a pose is correct or not. We accomplish this by analysing the failure cases and, accordingly, developing scores for predicting when the failures occur. At testing time frames classified as failures are discarded.

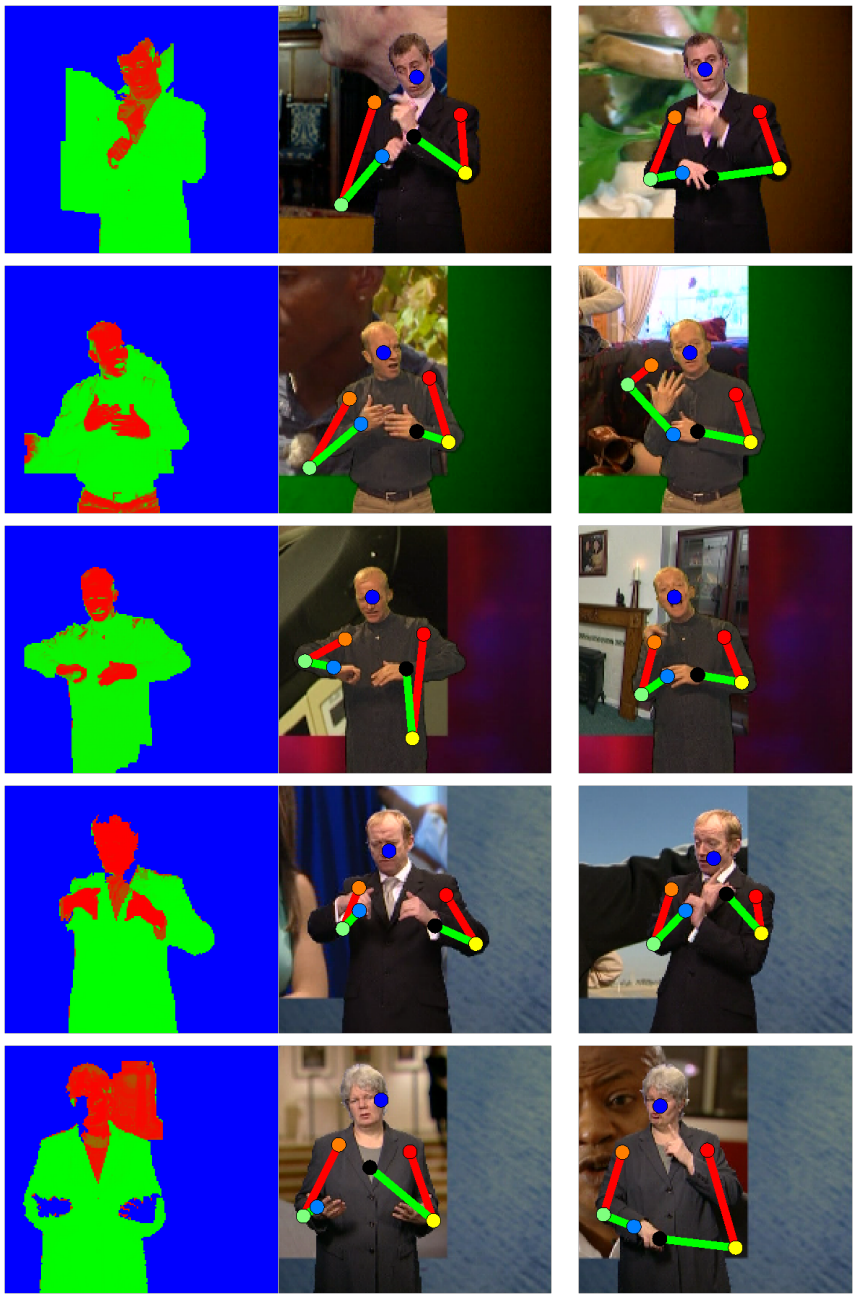
Figure 4.5 shows the main causes of failure: frames where the segmentation is faulty ($\approx 80\%$ of errors), and where the left and right hand are confused ($\approx 5\%$ of errors). The approach here will be to develop separate methods for detecting each of these failures. An SVM is trained to predict failed frames using the output of these methods as a feature vector. The classifier yields a simple lightweight evaluator that predicts whether the pose is correct or incorrect.

The features for the classifier are discussed in Sections 4.3.1 and 4.3.2, and details on the SVM that combines the features are given in Section 4.3.3.

4.3.1 Feature 1: Segmentation scores

The segmentations occasionally either over-segment or under-segment the foreground, due to a similar foreground and background or face detection failures. This in turn results in incorrect pose estimates.

One way to detect failures is to compare the segmentations to ground truth



(a) (b)

Figure 4.5: **Typical pose estimation errors.** (a) frames with segmentation failures, with the failed segmentation (left) and failed pose estimate (middle). (b) frames where the left and right hands are confused. Poses estimates are illustrated with a colour coded skeleton.

(a) (b)

Figure 4.6: **Segmentation score for evaluator.** (a) the silhouette (red boxes) rendered based on estimated joint positions. (b) the segmentation (black), rendered silhouette (yellow) and their overlap (red) which is used as a segmentation score.

segmentation masks – however, these are only available for a limited number of test frames. Instead, we exploit our joint estimates to render a partial silhouette (Figure 4.6a). This is done by rendering a rectangular binary mask for each limb given joint locations. Rectangles covering the head and arms are added according to the joint positions, and a rectangle covering the torso is added based on the shoulder positions. The partial silhouette can then be compared to the segmentation from the co-segmentation algorithm as shown in Figure 4.6b, resulting in scores such as those in Figure 4.7.

Several segmentation scores are computed based on the output of this rendering. First, we compute a standard overlap score $o = \frac{T \bigcap A}{T \bigcup A}$ for comparing the two silhouettes, where T is rendered partial silhouette and A is the mask generated by the co-segmentation algorithm (see Figure 4.8). Second, a Chamfer distance between the silhouettes is also computed, yielding a measure of the similarity of the shapes of the silhouettes. Third, statistics based on the size of the segmenta-

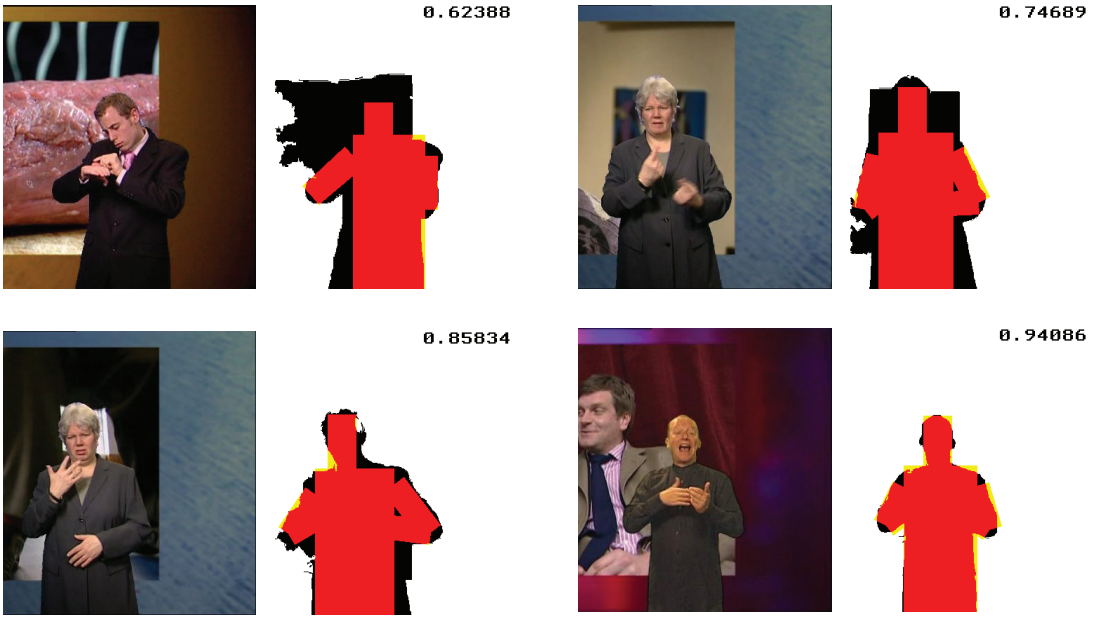


Figure 4.7: **Examples of frames with different segmentation overlap scores.** The masks show the segmentation (black), rendered silhouette (yellow) and their intersection (red).

tion are computed. These are absolute mask size $\|A\|$, difference between mask size and median mask size over all frames $\|M\|$: $\Delta = \frac{\|A\|-\|M\|}{\|M\|}$, Δ re-computed with temporally local medians, and differences between different Δ's.

These scores form the first part of the feature vector for the evaluator classifier.

4.3.2 Feature 2: Mixed hands

Another common error case is when the left and right hand are confused with each other, *i.e.* the left hand is connected to the right elbow and/or vice versa. In order to catch these failures, we train a classifier with local Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] features to detect correct and incorrect assignments. The tracking output from [Buehler et al., 2011] is used as manual

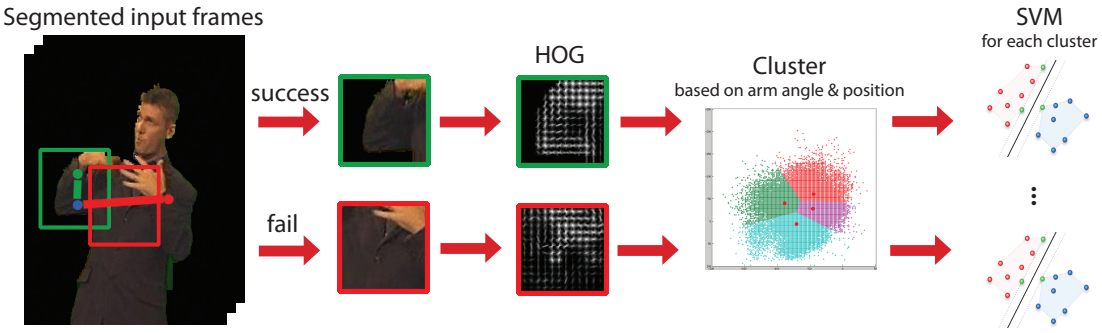


Figure 4.8: **Training the detector for mixed hand predictions.** The evaluator is trained on HOG feature vectors placed in the middle of the correct and incorrect positions of the lower arm. Feature vectors are clustered into separate SVMs based on the hand-elbow angle and hand position.

ground truth. The examples are clustered with K-means according to the hand-elbow angle and hand position into 15 clusters. One SVM is trained for each cluster as shown in Figure 4.8 (similar to poselets [Bourdev and Malik, 2009]). The HOG is computed in the middle of the lower arm. At test time, as shown in Figure 4.9, predicted joints are assigned to the nearest cluster centroid based on hand-elbow angle and hand position. The SVM for this cluster is evaluated and the output score forms the second part of the feature vector for the evaluator classifier.

4.3.3 Evaluator

The above two features (segmentation scores and mixed hands) are then used to train an evaluator, which classifies the body pose estimate of each frame as either success or failure. To this end we train an SVM with a Chi-squared kernel based on the the above two feature sets (9 scores for segmentation – 1 overlap score, 1 Chamfer score and 7 size statistics; and 1 score from the mixed hand

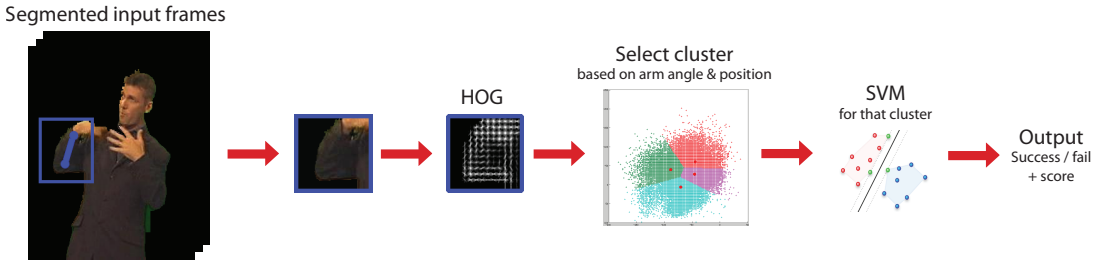


Figure 4.9: **Testing the hand mixup detector.** The SVM trained on a cluster whose centroid best represents the predicted joints is chosen to evaluate the HOG feature vector placed in the middle of the hand and elbow positions. This SVM outputs a failure score which the evaluator exploits as a feature for predicting whether the pose estimate is successful or failed.

classifier). An increase in accuracy was observed after adding each feature. The joint tracking output from [Buehler et al., 2011] is used to automatically label the training set. This yields a simple lightweight evaluator (with a feature vector of dimension 10) that predicts whether the pose is correct or incorrect.

4.4 Experiments

In this section, the performance of the co-segmentation algorithm, joint position estimator and pose evaluator is first assessed (Sections 4.4.1–4.4.3), and then the computation time of the methods is discussed (Section 4.4.4).

The experiments are performed using the BBC Pose train/validation/test dataset and evaluation measures described in Chapter 3.1.1.

4.4.1 Co-segmentation

The co-segmentation algorithm is evaluated in two experiments. The first experiment uses ground truth segmentation masks to evaluate the quality of segmentations. The second experiment uses silhouettes rendered based on ground truth joint locations as shown in Figure 4.6.

Experiment 1: Overlap of foreground segmentation with ground truth

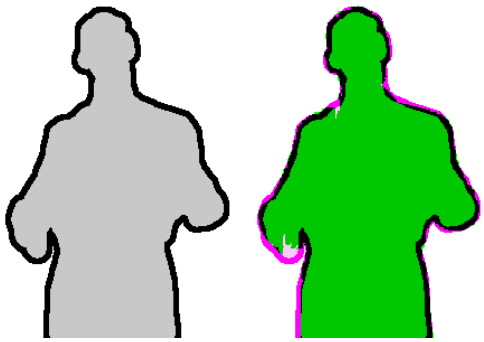
In this experiment the segmentation masks are compared against manual foreground segmentation ground truth. This ground truth consists of manually labelled foreground segmentation trimaps for 20 frames for each of the five test signers (100 frames in total). The frames are sampled uniformly from different pose clusters (as described in Chapter 3). The overlap score from Section 4.3.1 is evaluated separately for each test signer. The mean overlap scores and standard deviations are given in Figure 4.10. The results show that the segmentations are quite tight, but with some variation across different test videos (e.g. the segmentations for Signer 2 and 3 are somewhat poorer).

Experiment 2: Overlap of foreground segmentation with a larger number of automatically rendered silhouettes

In this experiment an overlap score is computed by rendering rectangles at the manual ground truth joint positions as shown in Figure 4.6. This is done using the frames in the test and validation sets that have manual ground truth joint

Signer	Overlap score	Standard deviation
1	0.962	0.049
2	0.959	0.072
3	0.947	0.043
4	0.960	0.025
5	0.965	0.043
Mean	**0.959**	**0.047**

(a)



(b) (c)

Figure 4.10: **Co-segmentation evaluation using overlap score.** (a) overlap scores for each test signer; (b) example of the ground truth trimap (white is background, grey is foreground and black is unknown); (c) segmentation (green) evaluated against the ground truth (magenta & black).

Data subset	Avg overlap score	Standard deviation
Test set	0.8628	0.0503
Validation set	0.8542	0.0637

Table 4.1: **Co-segmentation evaluation using overlap of segmentation and rendered silhouette.** Average overlap scores are shown for the validation and test sets.

locations (see Section 3), and is used for evaluating the quality of segmentations for the evaluator.

Table 4.1 shows the resulting segmentation overlap scores. A perfect overlap is not expected since the rendered rectangles are only approximations to the true ground truth segmentation. However, as demonstrated in Figure 4.7, the overlap score still gives a useful indication of whether the segmentation is good or not.

Figure 4.11 shows the cumulative distribution function of the overlap scores over the test and validation sets. It can be observed that the majority of scores are in the range 0.85 – 0.95, with no scores below 0.4 or above 0.95, and a small proportion of scores between 0.6 and 0.8. This demonstrates that the

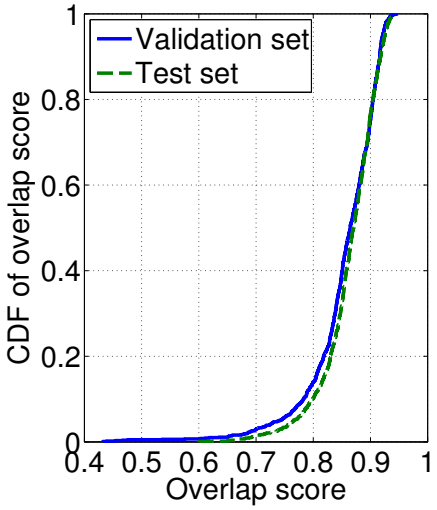


Figure 4.11: **Cumulative distribution function of segmentation overlap scores.** The majority of scores are in the range 0.85 – 0.95 (high overlap).

segmentation quality score used for the evaluator is fairly accurate.

4.4.2 Random forest pose estimator

The pose estimation method is evaluated in three experiments:

1. Frame representation: This explores alternative inputs for the forest and demonstrates the effectiveness of using a segmented colour posterior image (obtained through co-segmentation) over using other simple representations.

2. Increasing training data: This part evaluates the performance of the random forest as the amount of training data is increased.

3. Random forest vs. state-of-the-art: This part pitches our joint estimation method against Buehler *et al.*'s tracker, and the pose estimation method

of [Yang and Ramanan, 2011] which uses a mixture of parts.

The experiments in this section were conducted by James Charles.

Experiment 1: Frame representation

This section investigates four alternative frame representations: (i) a raw colour pixel representation in LAB (LAB); (ii) colour posterior on the whole image (CP); (iii) signer silhouette (S); and (iv) segmented colour posterior (Seg+CP), produced through co-segmentation (examples showing each type are shown in Figure 4.12a).

The forests are evaluated using 5-fold cross validation on videos of 5 different signers, where the random forests are trained on 4 videos and evaluated on a 5th "held-out" video.

Figure 4.12 shows average joint estimation accuracy for the forests as the threshold on allowed distance from manual ground truth is increased. It can be noticed that using LAB does not generalise well, and performs the worst. On the other hand, SEG+CP maintains best performance. Removing the background content and using SEG+CP allows the forest to learn a more refined appearance of body joints and boost detection accuracy by reducing the influence of noise. However, the method is left at the mercy of the background removal procedure.

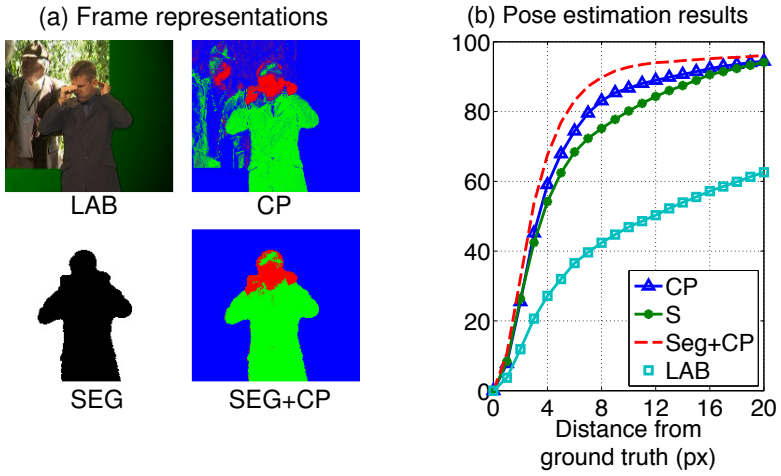


Figure 4.12: **Comparison of different input frame representations.** (a) Example frames showing different methods for representing a frame. (b) Average accuracy of forests as allowed distance from ground truth is increased. Using SEG+CP proves best.

Experiment 2: Increasing the amount of training data

This experiment tests the intuition that more training data will improve generalisation of the forest and hence increase the accuracy of joint estimates.

Protocol. Multiple forests with the SEG+CP frame representation are trained for samples of 2, 4, 6, 8 and 10 videos from the set of 10 training videos. Finally we also train a forest with 15 videos using the testing and validation sets combined, which which we are not able to tune parameters due to a limited number of available videos so they are fixed at the optimal parameters found when training with 10 videos. All forests are tested on 1,000 ground truth frames from videos in the testing set.

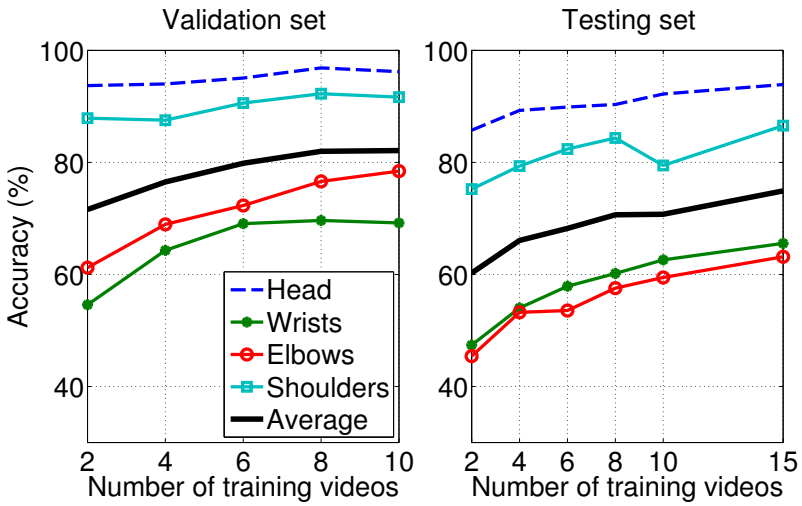


Figure 4.13: **Forest performance with increasing amount of training data.** Results on validation and testing sets are shown.

Results. Figure 4.13a shows the average accuracy achieved by forests on the validation set. For all joint types we observe a general increase in accuracy as more training data is added. The same trend is observed when applying these forests to unseen signers in the testing set as shown in Figure 4.13b. Of particular interest is the drop in accuracy of the shoulder joints when going from 8 to 10 videos. We believe this is due to a particular video having noisy segmentations on the signer's left shoulder. It can also be noticed that elbows have higher accuracy than wrists in the validation set, but *vice versa* on the testing set. This is due to more segmentation errors at elbow locations in the testing videos.

Experiment 3: Random forest vs. state-of-the-art

In this experiment the random forest is compared to Buehler *et al.*'s tracker (described in Section 2.1.6 and the deformable part based model by [Yang and

Ramanan, 2011] (described in Section 2.1.3.

Protocol. The forest is trained on the full 15 video training set. Parameters are optimised on the validation set: a tree depth of 32, window size of 71 and 8 trees is used.

The model by [Yang and Ramanan, 2011] is trained for two different types of video input: (i) The original RGB input; and (ii) an RGB input with the background content removed by setting it to black. For both types of input the full 15 video dataset is used for training. From each training video, 100 diverse training frames were sampled, totalling 1,500 frames. Model parameters were set the same as those used for upper body pose estimation in [Yang and Ramanan, 2011]. Negative training images not containing people were taken from the INRIA dataset.

Results. Figure 4.14 shows accuracy as the allowed distance from ground truth is increased. For all joints but the head, the forest consistently performs better than Buehler *et al.*'s tracker. For the wrists and shoulders, erroneous joint predictions by the forest are further from the ground truth than erroneous predictions from Buehler *et al.*'s tracker once joint predictions are at least ≈ 10 pixels from ground truth. This indicates that it is likely to be easier for a pose evaluator to detect errors made by the forest. Interestingly, the model by [Yang and Ramanan, 2011] achieves best performance when using the original RGB video input (input 1) over using a background removed version (input 2). We suggest that this is due to a poor representation of negative image patches in input 2 when using

negative training images from the INRIA dataset. Overall, Yang & Ramanan's model is the least accurate over all joint types.

Qualitative results for the forest on an example 5 frames from the testing set is shown in Figure 4.15.

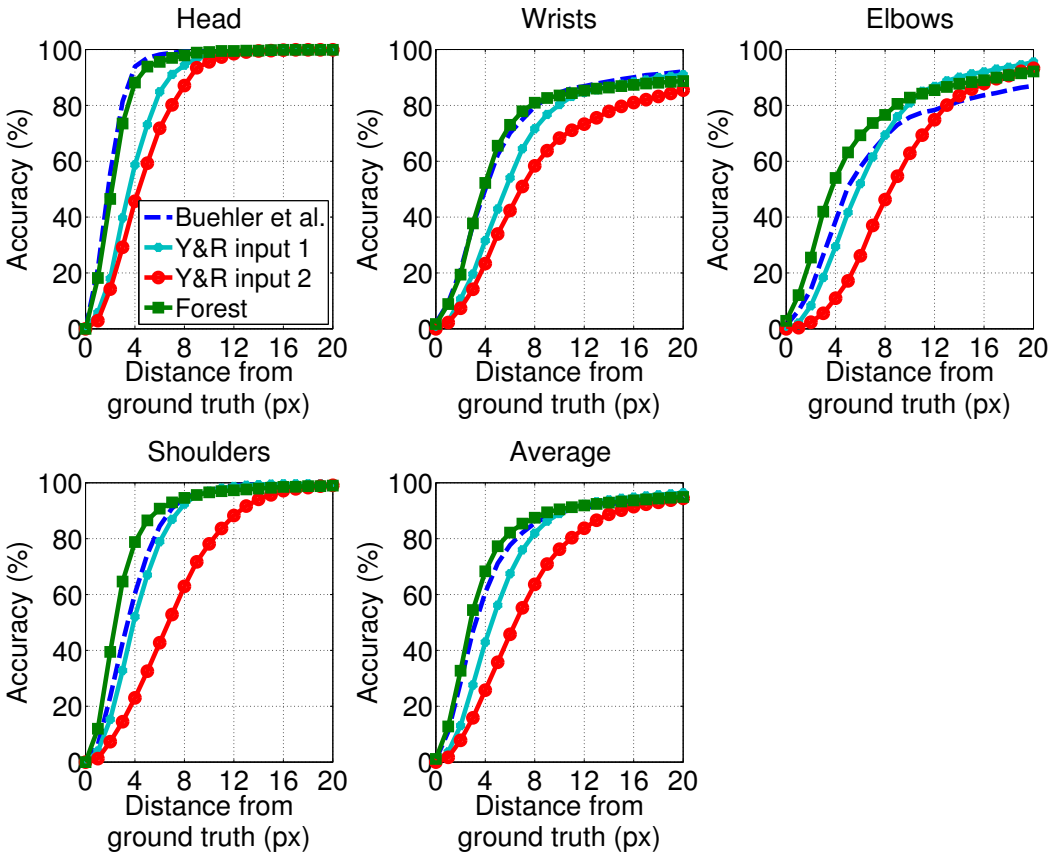


Figure 4.14: **Comparison of joint tracking accuracy of random forest against [Buehler et al., 2011] and [Yang and Ramanan, 2011].** Plots show accuracy per joint type (averaged over left and right body parts) as allowed distance from manual ground truth is increased.

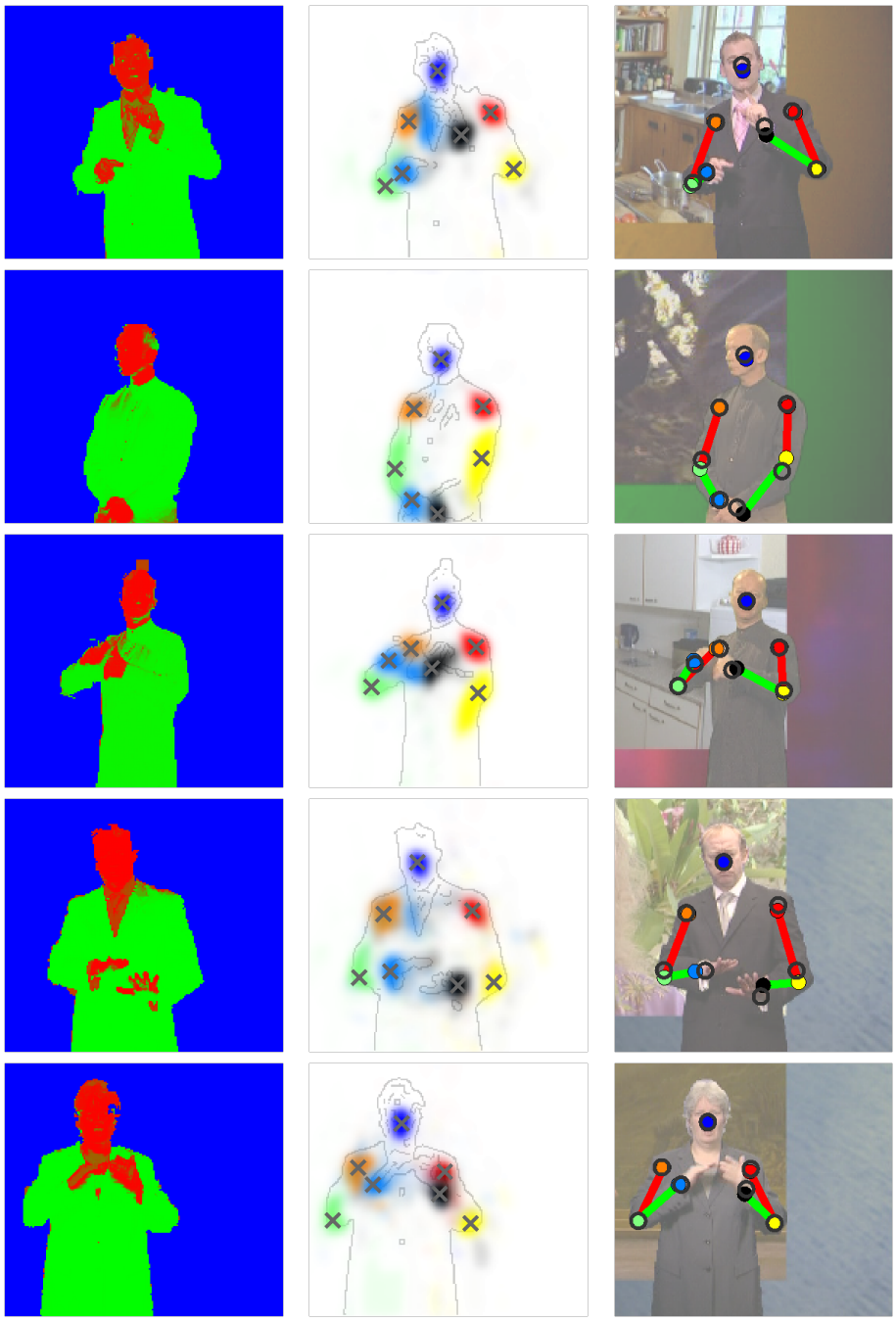
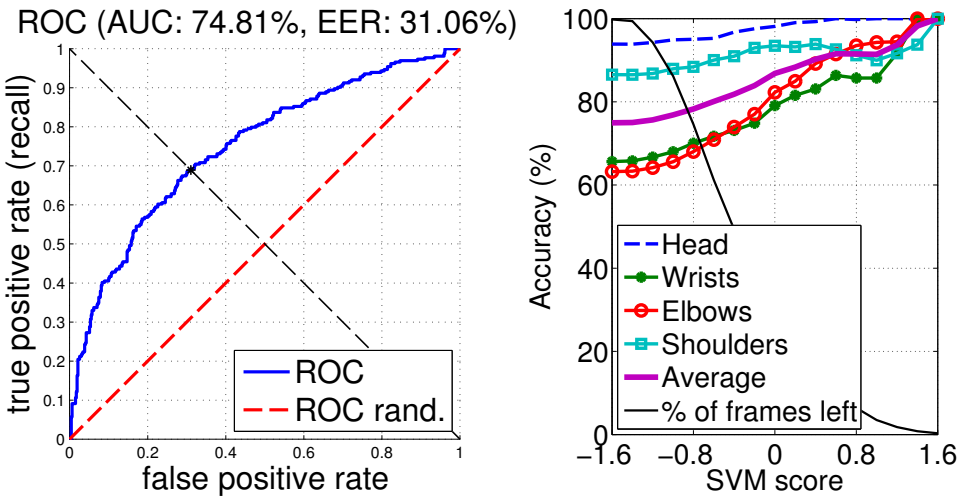


Figure 4.15: **Qualitative pose estimation results. Left** shows colour model images, from which we obtain probability densities of joint locations shown on top of the colour model edge image in **centre**. Different colours are used per joint (higher intensity colour implies higher probability). Maximum probability per joint is shown as grey crosses. **Right** shows a comparison of estimated joints (filled in circles linked by a skeleton are) overlaid on faded original frame, with ground truth joint locations (open circles).



(a) (b)

Figure 4.16: **Pose evaluator classification performance.** (a) ROC curve of the evaluator. (b) Change in accuracy as a function of the percentage of frames left after discarding frames that the evaluator detects as failures. For (b) the accuracy threshold is set as 5 pixels from manual ground truth.

4.4.3 Pose evaluator

The pose evaluator is assessed here on the ability to label joint predictions per frame as either success or fail. The quality of joint predictions on success frames is also used as a measure of the evaluator's performance.

Protocol. The evaluator is trained on the validation set and tested on the test set shown in Figure 3.2. For training, the joint tracking output from [Buehler et al., 2011] is used to automatically label poses for a set of training frames as success or fail (based on whether they agree with the Random Forest tracking output or not). For testing, the 1,000 frames with manual ground truth are used.

Results: Choice of operating point. Figure 4.16a shows the ROC curve of the evaluator when varying the operating point (effectively changing the threshold of the SVM classifier's decision function). This operating point determines the sensitivity at which the evaluator discards frames. The optimal operating point occurs at a point on the curve which best trades off false positives against true positives, which is the point closest to the top left hand corner of the plot. To gain further insight into the effect of the operating point choice on joint estimates, we plot this value against joint prediction accuracy in Figure 4.16b. This illustrates the correlation between the operating point and percentage of frames that the evaluator marks as successes (*i.e.* not failures). One can observe that when keeping the top 10% frames, a 90% average accuracy could be attained. More frames can be kept at the cost of loss in average accuracy. The bump at 0.8 suggests that at a particular SVM score, the pose evaluator begins to remove some frames which may not contain a higher degree of error compared to frames removed with a higher SVM score threshold. In general, however, there is a positive correlation between the SVM score and pose estimation accuracy.

Results: Joint localisation. Figure 4.17 demonstrates the improvement in joint localisation obtained by discarding frames that the evaluator classifies as failed. This yields an 8.5% increase in average accuracy (from 74.9% to 83.4%) at a maximum distance of 5 pixels from ground truth, with 40.4% of the test frames remaining. One can observe a particularly significant improvement in wrist and elbow localisation accuracy. This is due to a majority of hand mixup frames being correctly identified and filtered away. The improvements in other joints are

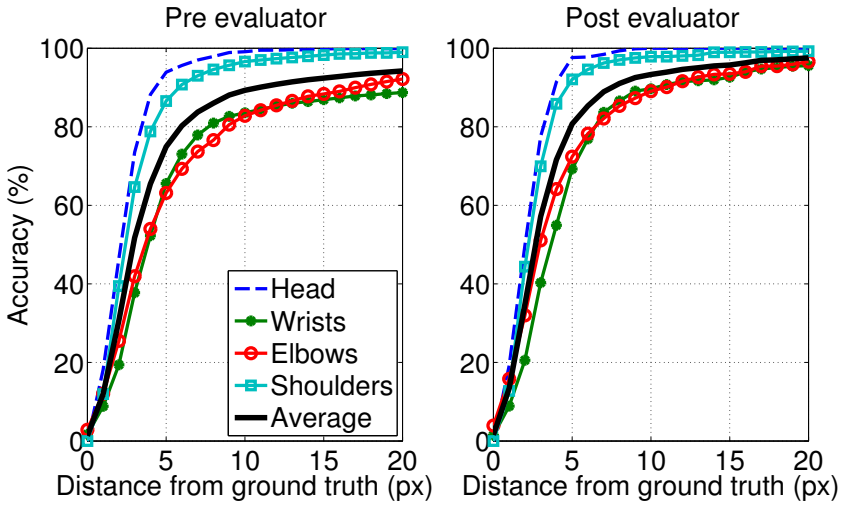
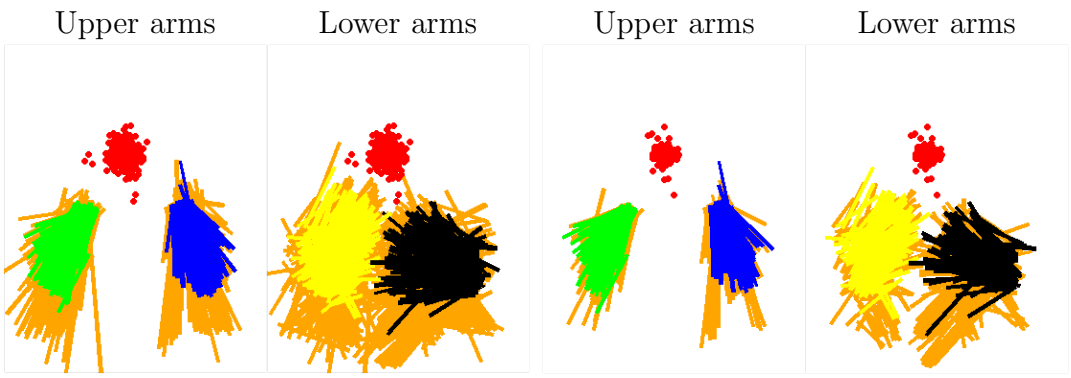


Figure 4.17: **Pose estimation with/without evaluator.** Average accuracy of per-joint estimates are shown without (left) and with (right) evaluator when the operating point of the pose evaluator is set to the optimum in Figure 4.16a.

due to the evaluator filtering away frames where joints are assigned incorrectly due to segmentation errors.

Results: Pose visualisation. A scatter plot of stickmen for the forest joint predictions are plotted on all test frames in Figure 4.18a. Sticks are marked as orange if the elbow or wrist joints are more than 5 pixels from ground truth. One observes erroneous joint predictions tend to exaggerate the length of upper arms. Typically wrist joint errors occur when the wrists are further away from the torso centre. Figure 4.18b shows the same plot as in Figure 4.18a but only on testing frames marked as successful by the evaluator. Notice the evaluator has successfully removed errors on the elbows and wrists while still retaining the majority of the correct poses.



(a) **Pre evaluator** (b) **Post evaluator**

Figure 4.18: **Pose visualisation with/without evaluator.** (a) shows scatter plots of stickmen for pose estimates from forest on all training data. (b) shows scatter plot of pose estimates from forest on training data marked as containing good poses by the evaluator. Elbow and wrist joints greater than 5px from ground truth are indicated by orange sticks.

4.4.4 Computation time

The following computation times are on a 2.4GHz Intel Quad Core I7 CPU with a 320×202 pixel image. The computation time for one frame is 0.14s for the co-segmentation algorithm, 0.1s for the random forest regressor and 0.1s for the evaluator, totalling 0.21s ($\approx$ 5fps). Face detection [Zhu and Ramanan, 2012] takes about 0.3s/frame for a quad-core processor. The per-frame initialisation timings of the co-segmentation algorithm are 6ms for finding the dynamic background layer and static background, 3ms for obtaining a clean plate and 5ms for finding the image sequence-wide foreground colour model, totalling 14ms (approx. 24min for a 100K frames). In comparison, Buehler *et al.*'s method runs at 100 seconds per frame on a 1.83 GHz CPU, which is two orders of magnitude slower. Each tree for our multi-signer random forests trained with 15 videos takes 20 hours to train.

4.5 Co-segmentation with Two Data Sources

The pose estimation method described in this chapter relies on accurate foreground segmentations of the signer, which (even with the described co-segmentation algorithm) are challenging to determine due to the colours of the foreground and background often being similar. In this section, we briefly describe an improvement for this co-segmentation method that we implemented post submission of the original version of this work. In particular, we show that for *new* TV broadcasts (those in Extended BBC Pose dataset, not in the original BBC Pose), the segmentation method described in this chapter can be improved upon by recording and using an additional source of freely available data.

The key observation is that the exact same TV programmes are broadcast *with* and *without* an overlaid sign language interpreter, as shown in Figure 4.19. If the two videos can be perfectly aligned, and any noise can be removed, then this provides a very strong cue for segmenting the foreground.

A caveat is that there are no pose estimation labels available from the semi-automatic tracker of [Buehler et al., 2011] (which we used to train the pose estimator in this chapter) for these new videos, which means that these new videos cannot be used directly for training a pose estimator. However, the pose estimator from this chapter can be applied to these new videos to estimate poses, resulting in more videos with (somewhat noisy) pose estimates (which are, qualitatively judging, significantly better due to better segmentations). This is the method used to generate segmentations for the Extended BBC Pose dataset, Sign Extraction dataset and Extended Sign Extraction dataset.

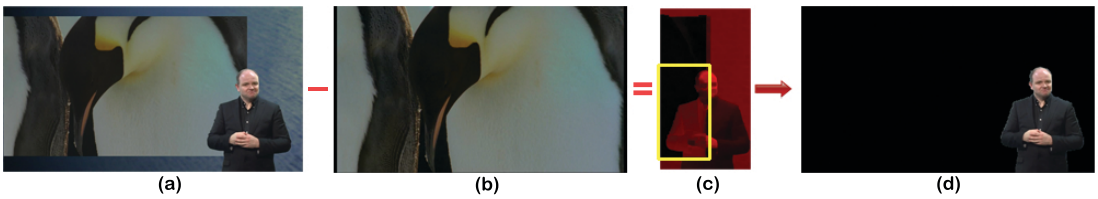


Figure 4.19: **Co-segmentation using two complementary data sources.** The signer-overlaid frame (a) and original frame (b) are subtracted after alignment, resulting in a difference image (c) that is used as a foreground clamp (in yellow) and to generate GrabCut constraints. (d) shows the segmentation output.

We next briefly describe how these two data sources are used to obtain foreground segmentations. Segmenting with the two data sources is not straightforward as the two videos differ greatly in broadcast quality (high definition vs standard definition), which results in many spurious edges if one simply computes an aligned difference image. We tackle this by finding edges in the original video and filtering these away from the difference image. The difference image then undergoes a set of image processing operations that produce a clean foreground 'clamp region' shown in yellow in Figure 4.19(c). This per-frame clamp region is used: (i) as a foreground clamping region in GrabCut [Rother et al., 2004], (ii) for building an accurate video-wide global foreground colour model, and (iii) for partially replacing the colour posterior unary in frames with similar foreground and background colours. A secondary GrabCut unary is also computed based on the background colour model of the video without an overlaid signer.

These improvements yield near-perfect segmentations similar to Figure 4.19(d) for signed TV broadcasts. We evaluate these results qualitatively, as the manual ground truth segmentation masks used in Section 4.4.1 are for old videos for which we do not have two sources of data (the original non-signed TV broadcast

is not available).

4.6 Conclusion

In this chapter we have presented a fully automatic arm and hand tracker that detects joint positions over continuous sign language video sequences of more than an hour in length. The method attains superior performance to a state-of-the-art long term tracker [Buehler et al., 2011], but does not require the manual annotation and, after automatic initialisation, performs tracking in real-time on people that have not been seen during training. Moreover, the method associates the joint estimates with a failure prediction score, enabling incorrect poses to be filtered away.

The contributions have more general applicability, beyond the BBC TV broadcasts: (i) the co-segmentation method could be easily generalised to other similarly laid out TV broadcasts, *e.g.* the majority of EU countries broadcast their signed TV broadcasts in a format suitable for this method; and (ii) joint positions can be predicted by a random forest from RGB in general once the person is segmented from the background (as in the Kinect line of research [Shotton et al., 2008], but here for RGB frames without depth).

Chapter 5

Enhancing the Random Forest Pose Estimator

This chapter presents three enhancements to the random forest pose estimator in Chapter 4:

1. Domain adaptation for short-sleeved signer videos: The pose estimator in Chapter 4 only functions on videos with long-sleeved signers, as the data it is trained on does not contain any videos with short-sleeved signers. We present a domain adaptation method to transfer the tracker from videos with long-sleeved signers to videos with short-sleeved signers (Section 5.1).

2. Sequential pose estimation: The pose estimator in Chapter 4 predicts all joints in one go. We present a method that predicts the joints sequentially, and show significant gains in performance (Section 5.2).

3. Temporal information for pose estimation: The pose estimator in Chapter 4 also predicts joints from a single frame. We show how temporal information can be used (in the form of optical flow) to improve the predictions (Section 5.3).

5.1 Domain Adaptation for Pose Estimation

Chapter 4 showed that we can train a robust real-time tracking system capable of generalising to new signers, but the method required that the signers were wearing long sleeves.

In this section we develop an upper body pose estimator that extends the set of videos that can be tracked. In particular, for short-sleeved signers (for which training annotation is *not* available), we show how we can *synthesise* training data by overlaying bare arms onto videos of long-sleeved signers (for which training annotation is available).

To this end, we will describe methods for:

1. Synthesising short-sleeved training data: We show how to use side-information about a signer's appearance in clothing (such as a specific sleeve length) in the target domain to produce semi-synthetic training data (Section 5.1.1, 'Stage 1').

2. Personalising the synthesised training data: We show that, given the semi-synthetic training data, one can re-synthesise it so it becomes signer-specific. This refined semi-synthetic data can then be used to train a *personalised* pose estimator that is tuned to a particular person's arm shape and sleeve length (Section 5.1.2, 'Stage 2').

Both steps contribute a significant boost to pose estimation performance, as shown in Figure 5.1(e).

Related work. Using synthetic images for training pose estimators has been successful in the past [Agarwal and Triggs, 2006, Everingham and Zisserman, 2005, Shakhnarovich et al., 2003, Shotton et al., 2011, Sun et al., 2012]. Of particular note is the work by [Shotton et al., 2011] where huge quantities of depth images were synthesised for training a full upper body pose detector. In our case, we adopt a similar approach to generate large amounts of data, but instead combine real data from signers wearing long sleeves and overlay synthetic sleeve information.

5.1.1 Stage 1: Synthesising training data

Our aim is to train a random forest pose estimator similar to Chapter 4, for use in videos where signers are wearing short sleeves. We show how to transfer knowledge learnt from the data containing long-sleeved signers and create thousands of semi-synthetic training images for signers wearing short sleeves, which we can use to train a random forest pose estimator tuned for a particular sleeve length.

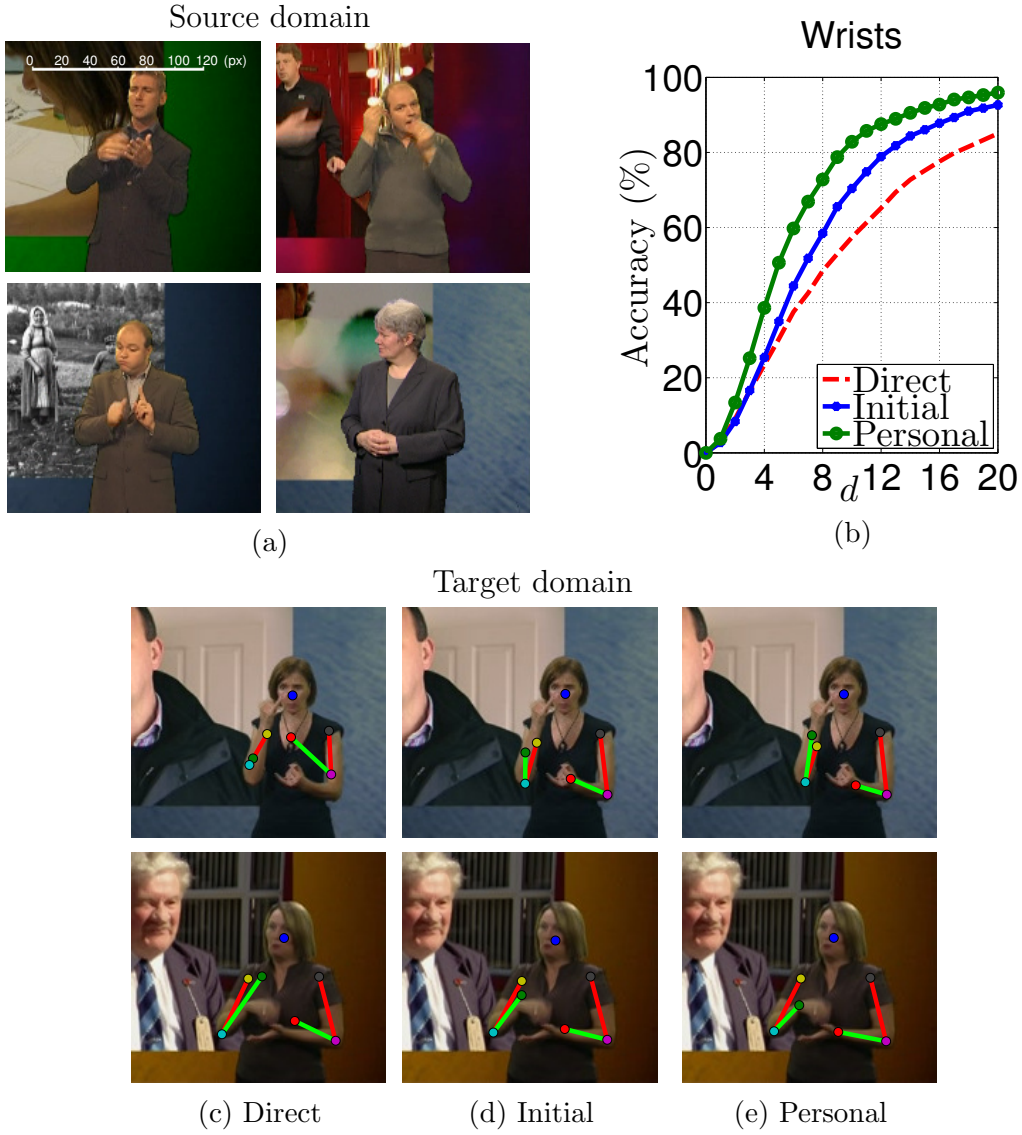


Figure 5.1: **Training an upper body pose estimator using knowledge from the source domain.** In this example, the transfer is from a source domain (a) where signers wear long sleeves, to a target domain (c)–(e) where signers wear short sleeves. The red and green lines on the signer show the output pose. The target domain results show: (c) the detector trained directly from the source domain; (d) trained from initial semi-synthetic images constructed using the source domain; and (e) from personal semi-synthetic images constructed from both source and target domain. (b) shows accuracy of (c)–(e) in detecting the wrist joint in the target domain. Accuracy is percentage of predicted joints within a distance d pixels from ground truth (scale bar shown in top left of (a)). The accuracy almost doubles at 8 pixels.

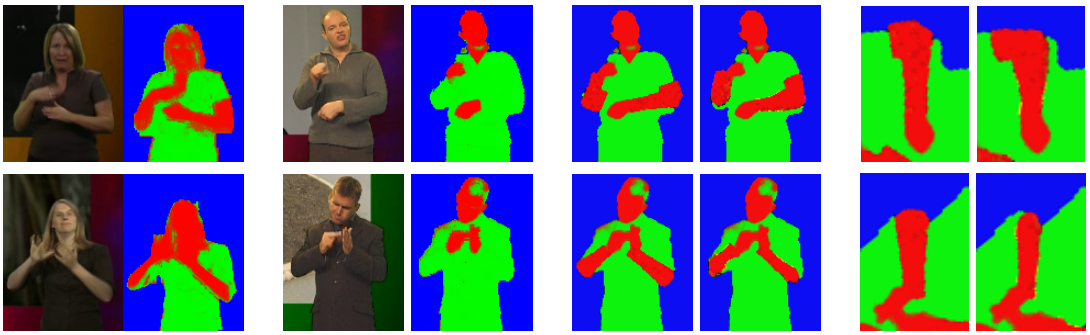
Similarly to Chapter 4, the input into our pose estimator is a colour posterior (CP) image (illustrated in Figure 5.2(a)), which highlights skin, torso and background regions which are important for tracking. To generate semi-synthetic data of signers wearing short sleeves, we take CP images of signers wearing long sleeves and augment them with bare arm information. We choose to synthesise CP images instead of using the original RGB image because it contains less information (less variation due to texture and lighting effects) and is therefore easier to model and synthesise accurately.

Semi-synthetic data for tracking a target signer wearing short sleeves is generated in four steps:

1. Obtain colour posterior (CP) images and body joint locations of signers wearing long sleeves: The body joint locations are obtained using the Random Forest pose estimator described in Chapter 4.

2. Measure the sleeve length of the target signer: Sleeve length of a target signer is provided as an input value between 0 and 1, which is the normalised length of the sleeve between the signers shoulder and wrist. For example, a sleeve length of 0.5 would represent fully sleeved upper arm and bare lower arm. For a target signer, sleeve length is measured manually on a single image.

3. Form an arm-skin template for upper and lower arms based on measured sleeve length: The appearance of bare arms in a CP image is synthesised using four rectangular templates describing upper/lower, left/right



(a) (b) (c) (d) (e) (f)

Figure 5.2: **Stages in synthesising training data.** Rows depict different sleeve length (sleeves in the top row are shorter than in the bottom row). (a) Raw RGB image and CP image counterpart of *short*-sleeved signers. (b) Example *long*-sleeved signers and CP images. (c) Synthetically produced CP images of short sleeves using CP images from (b) and of initial arm templates. (d) Personalised synthetic CP images using learnt arm templates. For closer comparison, rotated left arms of the synthetic images in (c) and (d) are shown in (e) and (f) respectively.

bare arms as shown in Figure 5.3. The shapes used are crude tapered rectangles from elbow to wrist for lower arms, and shoulder to elbow for upper arms, as shown in Figure 5.3.

4. Synthesise bare skin colour on arms using joint locations and arm templates: In this last step, the arm templates are positioned according to the provided joint locations and are used as stencils for placing skin colour values in the correct arm shape, as is shown in examples in Figure 5.2(c). Foreshortening of the arm part is handled naturally by anisotropic scaling of the template in the shoulder/elbow-to-elbow/wrist direction for upper/lower arm parts.

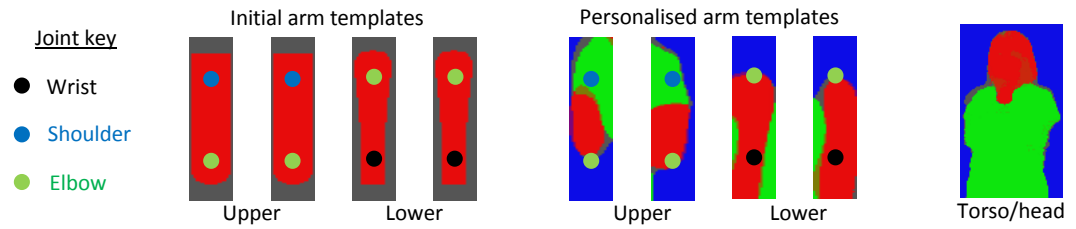
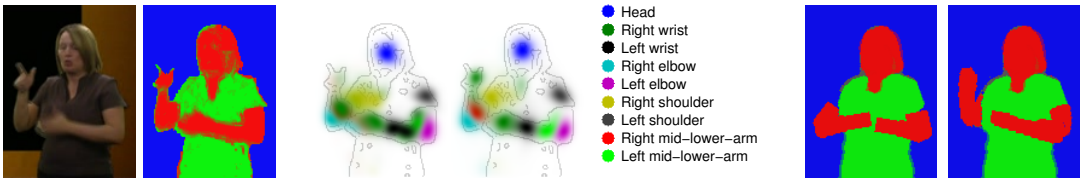


Figure 5.3: **Initial and personalised arm templates**. The personalised arm and torso/head templates shown here are for the signer in Figure 5.4(a).



(a) (b) (c) initial (d) personal (e) (f) initial (g) personal

Figure 5.4: **Illustration of sample and verify procedure.** (a) Target signer and (b) the corresponding CP image. (c) Shows body joint confidence map from initial forest, (d) shows the confidence map from the personalised forest, and (e) gives the body joint colour key. (f) and (g) show the most likely whole-image templates using initial and personalised arm templates respectively (computed by sampling joint locations from (c) and (d) respectively). The personalised model produces better joint samples (here for the right wrist).

5.1.2 Stage 2: Personalising the synthesised training data

In this section, we show that the synthetic training data from Stage 1 can be personalised (as shown in Figure 5.2). Figure 5.3 shows examples of personalised arm templates learnt from the signer in Figure 5.4(a).

This method continually alternates between updating the arm templates and re-training a random forest joint detector, until no change occurs in the arm templates.

The initial semi-synthetic CP images with a sleeve length matching a target signer from Stage 1 are refined in four steps:

1. Train a random forest pose estimator on semi-synthetic images and apply to frames of the target signer. We train a random forest body pose estimator (as in Chapter 4), but instead use the semi-synthetic images from Stage 1 for training. This forest is applied to a set of CP images containing the target short sleeved signer, as shown in Figure 5.4(a)-(b). The random forest provides a confidence map per joint per pixel (shown in Figure 5.4(c)).

2. Refine the pose estimates using a sample and verify approach: This confidence map could be used to infer body joint locations by selecting the points with maximum confidence per joint independently. However, the most confident location is not always the correct location, and the independence assumption is incorrect. We address these problems by adopting a sample and verify approach similar to [Buehler et al., 2011] and [Charles and Everingham, 2011]. Joint locations are sampled from the confidence map and scored with a verification function that considers the whole image content. This gives a chance to locations with weak confidence, and considers all joints dependently. The sampling process is repeated and the best scoring sample is selected.

3. Learn personalised arm templates from refined pose estimates: Given the improved joint predictions from the sample and verify approach, personalised arm templates are learnt. The joint predictions are used to extract rectangular windows of upper and lower arm parts, resulting in four sets of training windows for upper/lower and left/right arm parts. These training windows are then used to learn colour distributions for the personalised arm templates.

4. Re-train the pose estimator with personalised arm templates synthesised on top of source (long-sleeved) videos: Finally, the pose estimator is retrained on the synthesised CP images using the personalised arm templates. This pose estimator can then be applied to the target short-sleeved signer video.

These steps are repeated until no change occurs in the arm templates (typically 3 iterations for our data).

Figure 5.4(c) and (d) demonstrate the improvement in joint confidence maps produced using the initial and personalised random forests.

5.1.3 Experiments

In this section we test our method on signers wearing clothes with varying sleeve length.

Short and long-sleeved videos. Experiments are conducted on 5 additional short-sleeved videos (similar to the videos in the BBC Pose dataset but with shorter sleeves). Sleeve length varies between 1 and 0.2. For long-sleeved training videos, frames from the 10 training videos of BBC Pose (Section 3.1.1) are used.

Testing data. Each of the 5 short sleeve videos is split into two sections. The first 60% is used for training and the last 40% is used for testing. The performance of our system is evaluated on 200 diverse frames selected from each testing section

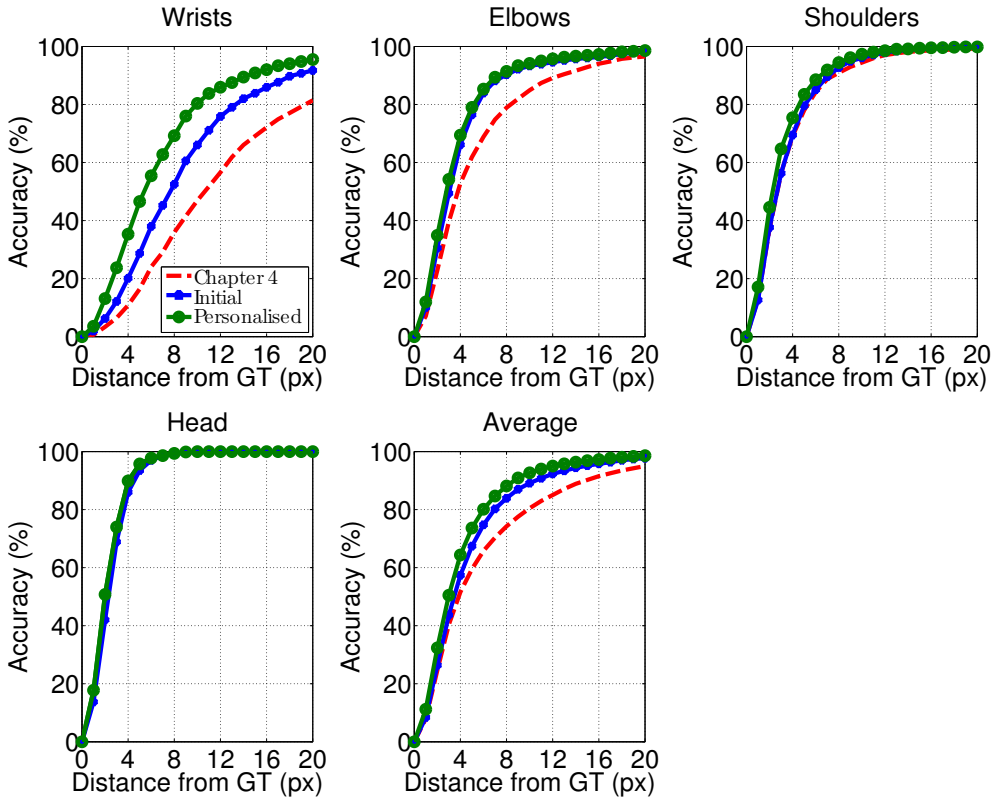


Figure 5.5: **Pose estimation performance for short-sleeved test videos**, showing a comparison of Chapter 4 (trained on long-sleeved training data) to the initial and personalised short sleeve pose estimator. Per-joint average accuracy against allowed pixel distance from ground truth is shown.

by clustering poses (1000 frames in total). Testing frames are manually annotated with joint locations.

Parameters. Tree depth of 64 and 8 trees in the forest was optimal. Stage 1 forests are trained on 2,000 CP images; Stage 2 forests on 4,000 CP images. We found two iterations of the refinement sufficient, with little improvement thereafter.

Method	Head	Wrists	Elbows	Shoulders	Lower Mid-arms	Average
Chapter 4	94.3	16.4	58.1	75.2	20.8	48.3
Initial without balancing	91.2	23.7	76.8	72.4	42.7	58.1
Initial with balancing	93.5	28.7	76.8	76.8	49.9	62.0
Personalised	**95.8**	**46.6**	**80.7**	**81.3**	**59.6**	**70.2**

Table 5.1: **Table showing pose estimation performance for short-sleeved test videos.** Average accuracy of per joint estimates is shown, comparing the pose estimator from Chapter 4 (trained on long-sleeved training data) to the initial and personalised short sleeve pose estimator. A joint is deemed correct if at most 5 pixels from manual ground truth. Personalising the training shows significant improvement.

Comparison to method in Chapter 4. Figure 5.5 shows a comparison between forests trained on long sleeve videos (Chapter 4), the *initial* forest trained on semi-synthetic images (with initial arm templates), and the updated forest using *personalised* semi-synthetic images. All methods work equally well for head and shoulders, but a large improvement is observed for wrists and elbows.

Table 5.1 shows accuracy when a joint is considered correct at only 5 pixels from ground truth (scale bar shown in top left of Figure 5.1(a) for comparison). Using forests trained on personalised semi-synthetics produces best results for all joint classes.

5.2 Pose Estimation with Sequential Forests

The pose estimators presented so far are all sliding window classifiers, which means that they only utilise context within a small sliding window per frame for making their predictions. This means they are unable to capture any global structure of the upper body and ignore dependencies between body parts, which results in some invalid upper body poses, and confusions between left and right

hand predictions. In this section, we show that we can mitigate these problems by implicitly encoding an upper body kinematic chain with a *sequential* forest.

We describe a pose estimation model with the following benefits:

1. Sequential prediction: The joints are estimated sequentially, taking account of the human kinematic chain. This means that we don't have to make the simplifying assumption of most previous Random Forest methods – that the joints are estimated independently.

2. Computational efficiency with mixture of experts: By combining multiple Random Forest classifiers and regressors (as a mixture of experts), we show that the learning problem is tractable and that more context can be taken into account.

The resulting method is computationally efficient, and can overcome a number of the errors (*e.g.* confusing left/right hands) made by pose estimators that infer their locations independently. We show that we improve upon the method in Chapter 4 on two datasets: the BBC TV Signing dataset (Section 3.1.1) and the ChaLearn Gesture Recognition dataset (Section 3.2.4).

5.2.1 Sequential pose estimation

Figure 5.6 illustrates the sequential detection. We encode the detection sequence so that each joint depends on the location of the previous joint in the kinematic chain: the head is detected first, followed by shoulders, elbows, and wrists, sep-

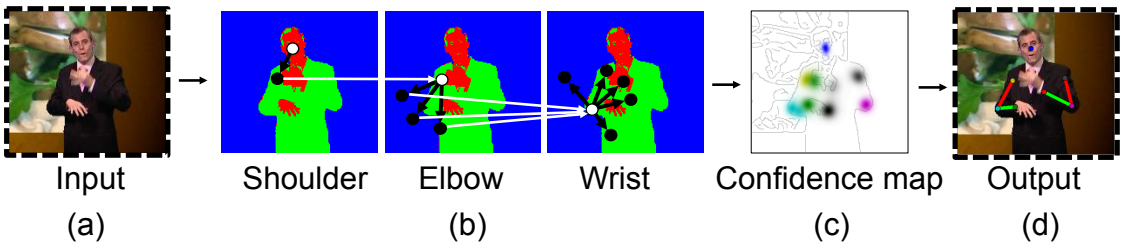


Figure 5.6: **Process diagram of sequential upper body pose estimation.** (a) Input RGB image. (b) Given the head position, joints are detected using a separate mixture of experts per joint, in order of shoulder, elbow then wrists. Knowledge of previous experts' output joint prediction (white arrows) is passed on when estimating the next joint in sequence. Each expert is responsible for a region of the image relative to the previous joint in sequence. Experts are positioned with fixed learnt offset vectors (positions shown as black dots with black arrows indicating offsets). (c) Joint detections produce a body joint confidence map (different colour per joint) shown superimposed on an edge image, and (d) the points of maximum confidence are selected (final pose output illustrated with a skeleton).

arately for left and right arms. Beginning with an RGB frame, the frame is first encoded into the colour posterior feature representation described in Chapter 4. For each joint, a separate mixture of experts (random forests) votes for the next joint location (votes shown as white lines in figure). Each expert (shown as black dots in figure) is responsible for a particular region of the image dependent upon the location of the previous joint in the sequence (positioned according to fixed learnt offset vectors, shown as black arrows). The output from this process consists of a confidence map over pixels for each joint, illustrated in Figure 5.6(c) with each joint in a different colour, with higher-intensity colours indicating higher confidence.

Two types of forests: classification and regression forests. Our sequential pose estimation method uses two types of Random Forests (RFs), one of each

type, for each expert: (1) classification forests, which measure the likelihood that an image region (determined by the learnt offset vector w.r.t. the previous joint) contains useful context for predicting the joint of interest, termed the region's 'utility score'; and (2) regression forests, which identify the joint's precise position within the regions, weighted by the utility scores from the classification forest. In contrast to the sliding window part detector RFs of such as those in Chapter 4 or [Shotton et al., 2011, Yang and Ramanan, 2011], our expert joint detectors are constrained to local image regions assigned by the classification forest. This considerably simplifies the learning task and yields a much more computationally efficient joint detector.

The sequential forest consists of four main components: (i) a global cost function, (ii) offset learning (between connected body parts), (iii) classification forests (for computing region utility scores), and (iv) regression forests (voting for joint positions in the regions, weighed by their region utility scores). We describe each of these in detail.

Global cost function. The sequential detection model is encoded as two separate 'chains' of detections, one for each arm. Each detection chain maximises the output from a mixture of 'experts' (random forest regressors), each of which votes separately for the most likely position of a joint within a region. These experts' votes are weighted according to the region's 'utility score', provided by the random forest classifier. Given an input image I, upper body joint detection reduces to optimising the cost function:

$$l_i = \arg\max_{l_i} \sum_{k=1}^{K} w_{ik} p(l_i|W_{ik}), \tag{5.1}$$

where $\{l_1, ..., l_4\}$ are 2D joint locations for head, shoulder, elbow and wrist for one of the arms; K is the number of experts (different for each body joint; we use 1 expert for shoulders, 3 for elbows and 5 for wrists); W_{ik} is the region for the k^{th} expert for joint i; w_{ik} are the region utility scores from the RF classifier, measuring how useful the context in image region W_{ik} is for predicting the position of the joint; and $p(l_i|W_{ik})$ is the likelihood of joint i being located at position l_i (obtained from the RF regressor). The locations of region centres are determined by previously detected joints and a learnt offset vector δ_{ik} (see below for how these are obtained) as $l_{i-1} + \delta_{ik}$.

Learning expert offset vectors. The human kinematic chain sets clear physical constraints for where one joint (*e.g.* the elbow) can be located given another joint (*e.g.* the shoulder). These 'offsets' essentially represent the most common positions of a joint relative to the previous joint in the sequence. We learn them from training data by clustering the relative offset of joint i from joint $(i-1)$ into K clusters using k-means (where K is defined above). The centroid of the clusters are used as the offsets δ_{ik}, and the variances are later used to add robustness to detections.

Classification forests. The offset vectors determine regions of the image that normally contain useful context for predicting the position of a given joint. How-

ever, for a given pose, not all pre-learnt regions are equally useful. We therefore employ a classification forest to obtain a 'utility score' for each of the regions. This utility score is used in the global cost function to weight the output from the expert body joint regressor (the regression forest). A separate classification forests, for each joint and expert, classifies an image region centred at W_{ik} as either *contained* or *not-contained* (indicating whether joint i is contained in the region or not). The average of all class probabilities across all trees is used to form the utility score w_{ik}. Parameters of test functions are learnt by minimising a measure of Gini impurity.

Regression forests. Given all pre-learnt regions and the position of the previous joint in the sequence, the task of the regression forests is to predict where in each region the joint is most likely to be. A separate regression forest, for each joint and expert, votes for pixel locations of joint i based on boolean tests performed on an image region centred at W_{ik}. All votes across all trees and all forests have equal weight. Parameters of test functions are learnt by recursively splitting a random selection of training regions (all of which contain the joint i) into two sets, based on minimising the sum of joint position variance within the two regions. The average joint position across regions falling at a leaf node is used as our voting vector. The aggregated voting vectors form the likelihood function $p(l_i|W_{ik})$.

5.2.2 Experiments

Three experiments are conducted using the standard long-sleeved BBC TV dataset (described in Section 3.1.1). Experiments 1 and 2 evaluate the quality of the structured output of the sequential detection system, and Experiment 3 evaluates the accuracy of pose estimates. Experiment 4 evaluates our method on the ChaLearn gesture dataset (Section 3.2.4) using RGB frames, and compares against Kinect skeletal output.

BBC TV experiments

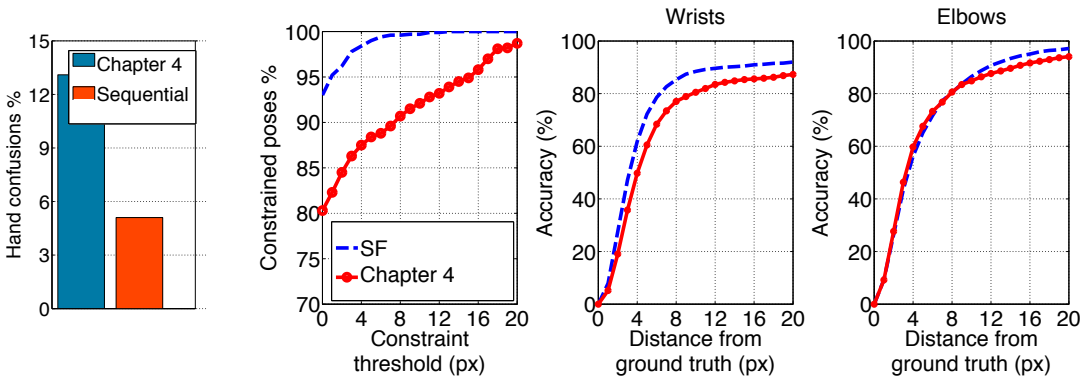
Parameter optimisation. Optimal parameters for the sequential detection forests are found on validation videos, and set as follows. Classification forests: 8 trees per forest grown to a depth of 20 and using a square window size of 51. Regression forests: 8 trees per forest grown to a different depth per joint type, 20 for wrists, 10 for elbows and 20 for shoulders; a square window size of 31 pixels is used for all joint types. After fixing the parameters, the forests are retrained on a pooled set of all training and validation videos.

Experiment 1: Constrained pose output. In this experiment we measure the proportion of output poses that are 'constrained' (essentially the proportion that is 'nearly correct'). We define a pose as 'correctly constrained' if the distance between connected joints (head to shoulder, shoulder to elbow and elbow to wrist) is less than the maximum ground truth projected limb length plus a threshold distance. Figure 5.7(b) shows the percentage of constrained pose outputs against

a constraint threshold (in pixels). Notably, 20% of pose estimates from Chapter 4 have invalid limb lengths (at constraint threshold zero), whereas the sequential forest (SF) only has 7%, with SF completely eradicating unconstrained pose outputs at a constraint threshold of 13 pixels.

Experiment 2: Hand confusions. This experiment measures the proportion of frames in which the two hands of the humans are confused (*i.e.*, right hand is detected as left, and vice versa) – a very common error in previous methods. Hand confusions are detected by subjectively analysing output on the test set for frames where a wrist estimate is further than 5 pixels from ground truth. Manual labelling of hand confusion is used, as automatic procedures are not reliable where the frames contain background segmentation errors or errors in the colour labelled image feature. As shown in Figure 5.7(a), SF reduces hand confusion errors by 61%, which is a considerable improvement.

Experiment 3: Pose estimation accuracy. This experiment measures pose estimation accuracy as a function of distance from ground truth. Figure 5.7(c) and (d) show results for wrists and elbows. Improved accuracy is observed using SF over Chapter 4 for both wrists and elbows, with a particularly marked increase in accuracy for the wrists. SF does not perform as well when (1) background subtraction fails (*e.g.* arms are cut off), or (2) hands are in front of the face or clothing contains skin-coloured regions.



(a) (b) (c) (d)

Figure 5.7: **Sequential forest performance on the BBC TV dataset.** (a) Bar chart showing percentage of hand confusions (lower is better); (b) number of constrained (near-correct) poses across all values of the constraint threshold (higher is better); (c) & (d) accuracy of average wrist and elbows joints respectively (averaged of left and right arms) against allowed pixel distance from ground truth (higher is better) (at 5 pixels).

ChaLearn experiments

Training & testing data. Each tree in both the sequential and classification forests sample 7,220 diverse frames from the training and validation videos combined. Diverse training frames are found by clustering poses and sampling uniformly from clusters. Testing data is formed by sampling 3,200 diverse frames from all testing videos. Kinect skeletal output is used as ground truth.

Parameter optimisation. We rescale the ChaLearn videos to be of same size as those in the BBC TV dataset, and use the parameters found optimal in the BBC TV experiments.

Baselines. The upper body tracker from Chapter 4 is used as a baseline (trained using the same parameters as for the BBC TV dataset). We train with the same

training frames used by our method.

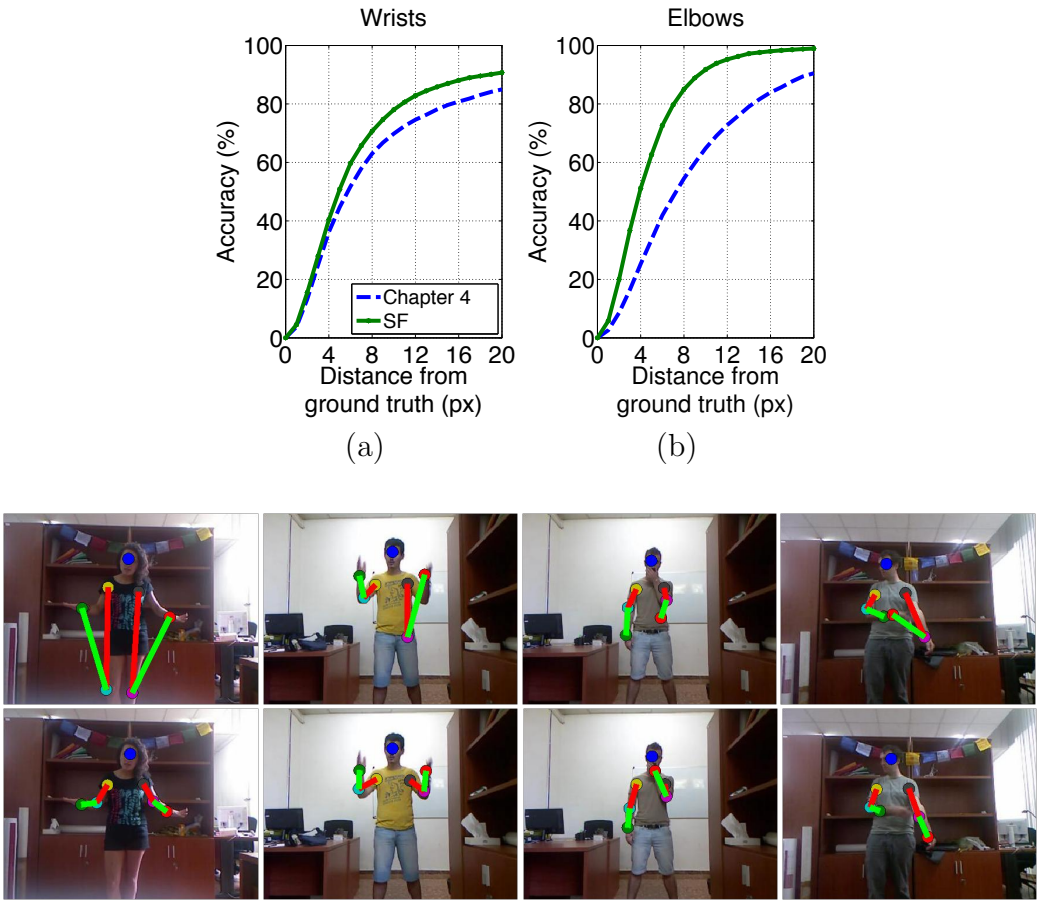
Experiment 4: Pose estimation accuracy. Figure 5.8(a) and (b) compares the method from Chapter 4 to SF. A significant improvement is seen in the wrists, with a stellar improvement for the elbows when using SF. The pose estimator in Chapter 4 does not generalise well to persons wearing trousers and sleeves of different length (due to confusions caused by legs and arm skin regions). The constrained output of SF helps overcome these problems.

Figure 5.8(c) shows example pose estimates from Chapter 4 and SF in top and bottom rows respectively.

Computation time. Using a 2.2GHz Intel Quad Core I7 CPU, computation time on a 320x202 pixel image is 0.4s for the sequential forest (272 trees in total). The sequential forest is implemented in Matlab with each tree only taking 1.5ms to evaluate the whole image due to the reduced search space. A classification tree takes 2.5 hours to train and a regression tree takes 4.5 hours.

5.2.3 Discussion

Most previous methods (such as that in Chapter 4) assume independence for each output variable, and ignore the output structure [Shotton et al., 2011]. Past solutions to this have been of two kinds: post-processing methods, and implicit methods. Post-processing methods take the output of the Random Forests and fit models to them, such as Markov or Conditional Random Fields [Jancsary



(a) (b)

(c)

Figure 5.8: **Sequential forest performance on the ChaLearn dataset.** (a) & (b) show average accuracy for wrists and elbows (against allowed distance from ground truth (Kinect provided skeleton) in pixels). Sequential forests (SF) show a significant improvement for wrist detections over Chapter 4, and nearly double the accuracy for the elbows at 5 pixels from ground truth. (c) Example pose estimates on ChaLearn for the forest in Chapter 4 (top row) and SF (bottom row).

et al., 2012], or simply filter the output by checking global consistency of local detections [Yang and Patras, 2013]. Usually post-processing methods are rather slow due to the additional overhead. In contrast, implicit methods build constraints between output variables into the detection method directly during training [Kontschieder et al., 2013] by passing the output from a sequence of

classifiers as an input into another classifier.

The method presented in this section addresses all of these issues by combining the benefits of both types of approaches (post-processing and implicit). First, unlike methods which aim to learn context, we take advantage of the kinematic constraints of the human body and explicitly build in context which we know is of importance, such as elbow location when detecting the wrist. A pose prior is implicitly built in by using a sequence of locally trained Random Forest experts for each body parts (incorporating both classification and regression forests). This sequential detection method is capable of learning strong dependencies between connected body parts while keeping the search window small and the learning problem tractable. Second, our method removes the need for a sliding window part detector. This allows many more trees to be used, boosting accuracy while still maintaining efficiency. Third, our method's locally trained Random Forests deal with less of the feature space compared to its sliding window counterparts, which makes learning easier and, as we show, leads to improved accuracy.

5.3 Temporal Information for Pose Estimation

The pose estimators presented so far operate on a frame-by-frame basis, and ignore the temporal information available in videos. However, in reality, strong dependencies exist between temporally close video frames. In this section, we show that optical flow can be used to temporally propagate predictions, and that this significantly improves the robustness of the pose estimates.

In particular, we show that the pose estimation confidence maps (such as those in Figures 4.15 and 5.6) produced at a frame t can be reinforced with temporal context from nearby frames. Additional confidence maps are produced for neighbouring frames, and are then aligned with frame t by warping them backwards or forwards using tracks from dense optical flow. This is illustrated in Figure 5.6(c) for confidences produced within n frames either side of frame t. These confidences represent a strong set of 'expert opinions' for frame t, from which joint positions can be more precisely estimated than when only using one confidence map. Finally, body joint locations are estimated at frame t by choosing positions of maximum confidence from a composite map produced by combining warped confidences (examples shown in Figure 5.6(d)).

5.3.1 Reinforcing pose estimates with optical flow

In more detail, the pose estimates are reinforced in three steps: (1) the confidences from nearby frames are aligned to the current frame using dense optical flow; (2) these confidences are then integrated into a composite confidence map; and (3) the final upper body pose estimate for a frame is then simply the positions of maximum confidence from the composite map. Below we discuss the details of the first two steps.

1. Warping confidence maps with optical flow. For a given frame t, pixel-wise tracks are computed from neighbouring frames $(t - n)$ to $(t + n)$ to frame t using dense optical flow [Weinzaepfel et al., 2013]. Tracks are used to

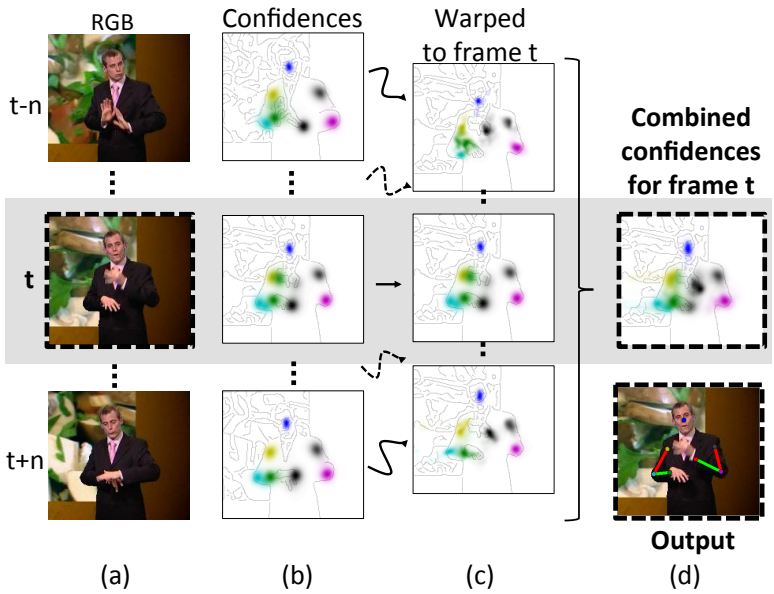
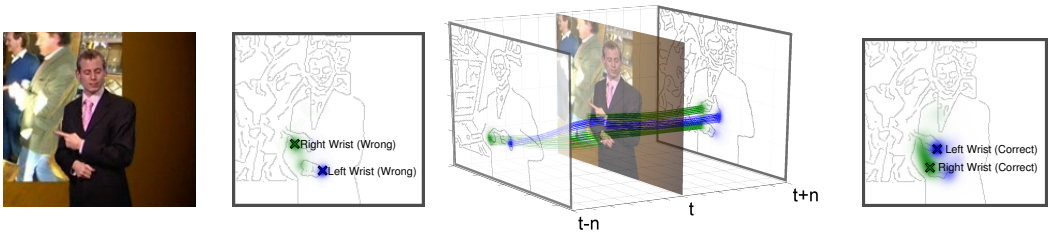


Figure 5.9: **Process diagram of temporal upper body pose estimation on RGB input frame t.** Detection reinforcement with optical flow for a person's right arm is shown running from left to right. (a) Given the head position, joints are detected using the sequential forest from Section 5.2. (b) Joint detections produce a body joint confidence map (different colour per joint) shown superimposed on an edge image. (c) Confidences at frame t are reinforced by combining warped confidences (using optical flow) from n frames either side of t. (d) Body joint estimates are chosen as points of maximum confidence from the combined confidence map, with final pose output illustrated with a skeleton on RGB frame t.

warp confidence values within a neighbouring map to align them to frame t by shifting confidences along the tracks. Given the horizontal and vertical dense optical flow maps vx and vy and a confidence map I, the warped confidence map W is:

$$W_{i,j} = \text{interp}_{i,j}(I, i - vx_{i,j}, j - vy_{i,j}) \tag{5.2}$$

where $\text{interp}_{i,j}$ returns the linearly interpolated value of I at coordinate $(i - vx_{i,j}, j - vy_{i,j})$.



(a) Frame t (b) Confidence at t (c) Warping (d) Composite at t

Figure 5.10: **Warping neighbouring confidence maps for improving pose estimates.** (a) RGB input at frame t. (b) Confidence map at frame t for left (blue) and right (green) hands (crosses show incorrect modes of confidence). (c) Confidence maps from frames $(t-n)$ and $(t+n)$ warped to frame t using tracks from optical flow (green & blue lines). (d) Composite map with corrected modes.

This process is repeated for confidences from all neighbouring frames. The final confidence map is an average of all these warped maps. Example tracks and the warping of wrist confidence values are shown in Figure 5.10(c).

2. Forming the composite confidence map. Aligned neighbouring confidence maps are integrated into a composite confidence map by taking the pixel-wise average (shown in Figures 5.9(d) and 5.10(d)). The composite map alleviates misdetections caused by failures in our feature representation. For example, Figure 5.11(a–d) shows examples where the image representation (b) loses relevant information w.r.t. the wrist location, with no hope of recovery using the confidences in (c). However, the composite confidence map (d) contains confidence in the correct regions even in these extreme circumstances. Figure 5.10(a-d) demonstrate the advantages of this method under the challenging scenario where the arms cross.

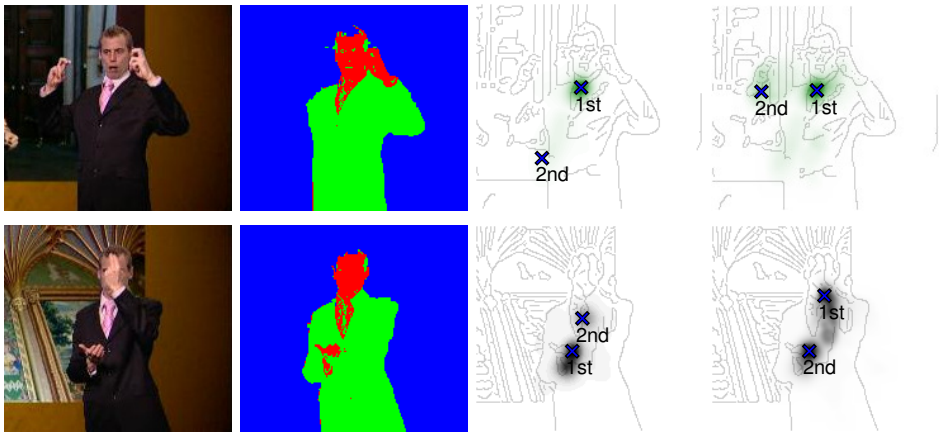


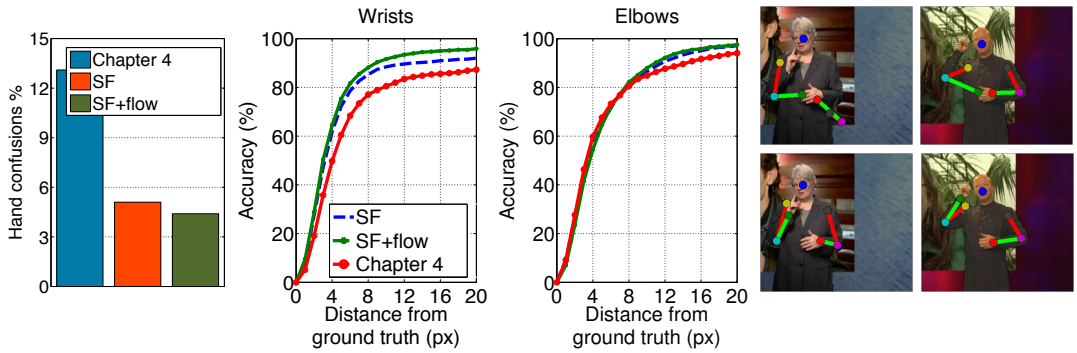
Figure 5.11: **The improvement from using optical flow.** (a) Example input images and (b) corresponding colour features, with the top row showing an error in segmentation (right arm missed), and bottom row showing a left hand in front of the face. (c) shows sequential forest (SF) confidences (superimposed on an edge image) for the left and right wrists for the two rows, with the first two modes of confidence (shown as crosses) both in wrong wrist locations. (d) demonstrates improved confidence output from using optical flow: in the top row the 2nd best mode of confidence now correctly locates the right wrist, and in the bottom row the 1st mode of confidence locates the left wrist.

5.3.2 Experiments

This section evaluates the improvement from using temporal information on the BBC Pose dataset of long-sleeved signers. Two experiments are conducted: the first measures the number of hand confusions, the second pose estimation accuracy.

Parameters. We use a neighbourhood of $n = 15$ frames to compute composite confidence maps.

Experiment 1: Hand confusions. Similar to Section 5.2, this experiment measures the proportion of frames in which the two hands of the humans are



(a) (b) (c) (d)

Figure 5.12: **Performance of sequential forests + optical flow on BBC TV dataset.** (a) Bar chart showing percentage of hand confusions (lower is better) – SF+flow does best; (b) & (c) accuracy of average wrist and elbows joints respectively (averaged of left and right arms) against allowed pixel distance from ground truth (higher is better) – SF+flow increases accuracy by 15% for wrists compared to Chapter 4 (at 5 pixels); (d) shows SF+flow (bottom row) correcting incorrect pose estimates from Chapter 4 (top row).

confused. Figure 5.12(a) shows the percentage of hand confusions. Sequential forests combined with optical flow (SF+flow) makes further improvements over SF alone, reducing hand swap errors by 66%.

Experiment 2: Pose estimation accuracy. Figure 5.12(b) shows pose estimation accuracy as a function of distance from ground truth. SF+flow gives an additional boost of 15% for the wrists, which is a significant improvement.

Computation time. Using a 2.2GHz Intel Quad Core I7 CPU, the computation time on a 320x202 pixel image is 1.2s for optical flow.

5.4 Conclusion

We have demonstrated three enhancements for Random Forests: (i) tracking persons with short-sleeves (by transferring from existing training material to a new domain and automatically personalising the tracker); (ii) predicting joints sequentially rather than independently (by implicitly encoding an upper body kinematic chain using a mixture of Random Forest experts); and (iii) using temporal information in the form of optical flow to improve the robustness of predictions (by spatially aligning pose confidence maps through time). All these enhancements are shown to improve pose estimation performance (with particularly significant improvements from predicting joints sequentially).

While the effects of these enhancements have been shown for Random Forests, they are generally applicable to other pose estimators. In the domain adaptation work (for short-sleeved signer videos), the synthesised training data could be used to train any other pose estimator (*e.g.* a deep convolutional neural network pose estimator). Similarly, the idea of cascaded pose estimation is not limited to RFs (and has recently also been investigated *e.g.* in the deep net literature [Toshev and Szegedy, 2014]). This cascade could be greedy (as in our sequential forest method) or globally optimised (as *e.g.* in [Yang and Ramanan, 2011, 2013]). Finally, using optical flow to warp the joint prediction confidences from past and future frames could be used with any pose estimator that predicts a confidence map of joint locations (as we show for CNNs in Chapter 6).

One could also imagine generalising the domain adaptation work to other kinds of differences (such as v-shaped necklines, long hair *etc.*) using a similar approach

as presented in this chapter. Another approach would be to fully *personalise* the video pose estimator for a particular person, allowing it to exploit person specifics (such as tattoos, hair length, jewellery *etc.*). We explore this idea in newer work (not included in this thesis) that has been submitted to BMVC'15.

Chapter 6

Pose Estimation with ConvNets

Building upon the recent success with deep neural networks, this chapter tackles the problem of pose estimation in videos with deep convolutional networks (ConvNets).

In Chapters 4 and 5 we showed that we can train a fast and reliable pose estimator using a random forest. However, the random forest relies on a hand-tuned foreground segmentation algorithm for preprocessing the videos, without which it performs poorly. In particular, without manual tuning of parameters, this segmentation method fails on certain videos with unusual body shapes, unusual absolute positions of persons in the video, or similar foreground and background colours. Furthermore, the segmentation algorithm requires extensive computationally expensive preprocessing (to build the layered model), reducing the speed of the method to near-realtime.

In this chapter we show how, with ConvNets, we can accurately predict the

pose without the need for foreground segmentation, and in real-time (150fps on a single GPU).

To this end, we present and compare three new deep nets for estimating human pose from videos:

1. 'Coordinate' network. This network regresses the coordinates positions of joints directly. This network exploits temporal information from multiple frames, leading to better performance than predicting from a single frame. (Section 6.1)

2. 'Heatmap' network. This network regresses a 'heatmap' of the joint positions. This allows visualisation of predictions and multi-modal output, yielding significantly improved performance over the coordinate network. (Section 6.2)

3. 'Optical flow heatmap' network. This network regresses a 'heatmap' of the joint positions and further exploits temporal information from multiple frames with optical flow. We show that optical flow from videos can be used to constrain pose estimates also in the context of ConvNets, thereby significantly improving performance. As in Chapter 5, optical flow is used to *warp* pose predictions from neighbouring frames. We show that a *parametric pooling* of these warped heatmaps can be learnt, effectively learning how to weigh the expert opinions for different neighbouring frames. (Section 6.3)[1]

[1]The latest version of this section (with updated results) can be found in a recent paper 'Flowing ConvNets for Human Pose Estimation in Videos' available at `http://tomas.pfister.fi`

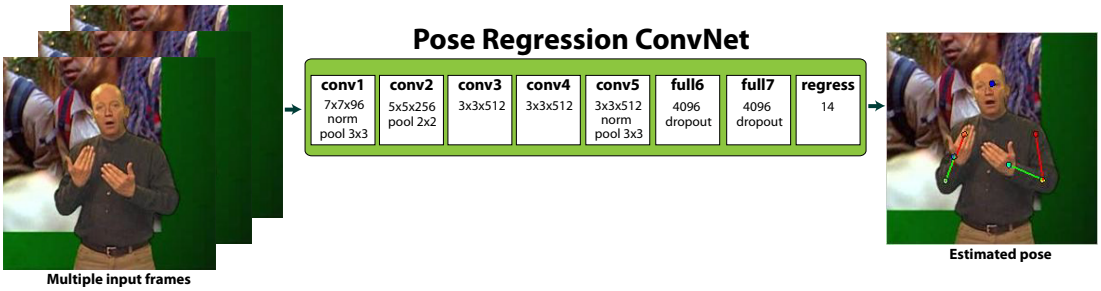


Figure 6.1: **Coordinate Network (CoordinateNet).** Given a set of input frames, the convolutional network regresses the positions of the head, shoulder, elbows and wrists.

In Section 6.4 (experiments), we show that these networks outperform the other methods presented in this thesis by a large margin.

6.1 Pose Estimation with a Deep Coordinate Network

In this section we treat the task of estimating the pose as a regression problem, where the regression targets are the (x, y) *coordinates* of the joints.

We show that this method implicitly learns constraints about the human kinematic chain, resulting in significantly better constrained pose estimates (*i.e.*, significantly smoother pose tracks with fewer serious prediction errors) than in previous work.

As the regressor we use a convolutional network. As shown in Figure 6.1, the input to the network is a set of RGB video frames, and the outputs of the last layer are the (x, y) coordinates of the upper-body joints. We base our network architecture on that of [Sermanet et al., 2014] which achieved excellent results on

ImageNet Challenge 2013 object classification and localisation tasks. Figure 6.1 shows the network architecture: five convolutional layers followed by three fully connected layers. A selection of convolutional layers are followed by pooling and local response normalisation layers, and the fully connected layers are regularised by dropout [Krizhevsky et al., 2012]. All hidden weight layers use a rectification activation functions (ReLUs).

6.1.1 CoordinateNet

Our network is trained for regressing the location of the human upper-body joints. Instead of the softmax loss layer, found in the image classification ConvNets [Krizhevsky et al., 2012], we employ an l_2 loss layer, which penalises the l_2 distance between the pose predictions and ground truth.

We denote $(\mathbf{X}, \mathbf{y})$ as a training example, where $\mathbf{y}$ stands for the coordinates of the k joints in the image $\mathbf{X}$. Given training data $N = \{\mathbf{X}, \mathbf{y}\}$ and a ConvNet regressor ϕ, the training objective becomes the task of estimating the the network weights λ:

$$\arg\min_{\lambda} \sum_{(\mathbf{X},\mathbf{y})\in N} \sum_{i=1}^{k} \|\mathbf{y}_i - \phi(\mathbf{X}, \lambda)\|^2 \tag{6.1}$$

An example visualisation of the loss is shown in Figure 6.2.

Multiple input frames. To exploit the temporal information available in videos, our network is trained on multiple video frames. This is in contrast to CNN pose estimators in previous work, which typically operate on a single

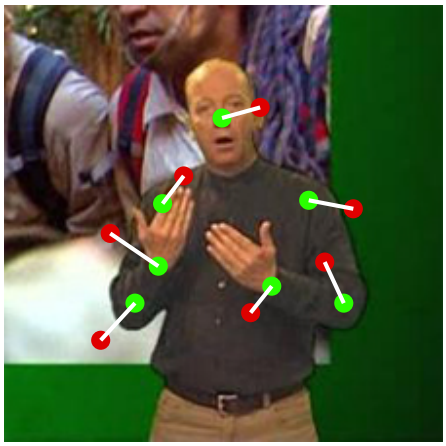


Figure 6.2: **Loss function for the CoordinateNet.** At each iteration of training, the squared l_2 loss between training labels (green) and current predictions (red) is computed. The loss is the sum of the squared l_2 distances between training labels and predictions (white lines), summed over all body joints.

frame. This is done by inserting multiple frames (or their difference images) into the data layer colour channels. So for example, a network with three input frames contains 9 colour channels in its data layer.

Video-specific learning: per-video mean. In contrast to RGB image pose estimation, videos generally contain several frames of the same person. Further, in our scenario, the person stands against a partially static background. We exploit this for an additional preprocessing step in our learning. Without this preprocessing step, we noticed that the network would overfit to the static background behind the person. To alleviate this overfitting, we compute the mean image μ_V over 2,000 sampled frames for each video V in our training and testing datasets and (before cropping the input image) subtract the video-specific mean from each input image: $x = x - \mu_V$ for frame x of video V. As shown in Figure 6.3, this removes the video-specific static background (and, as will be shown

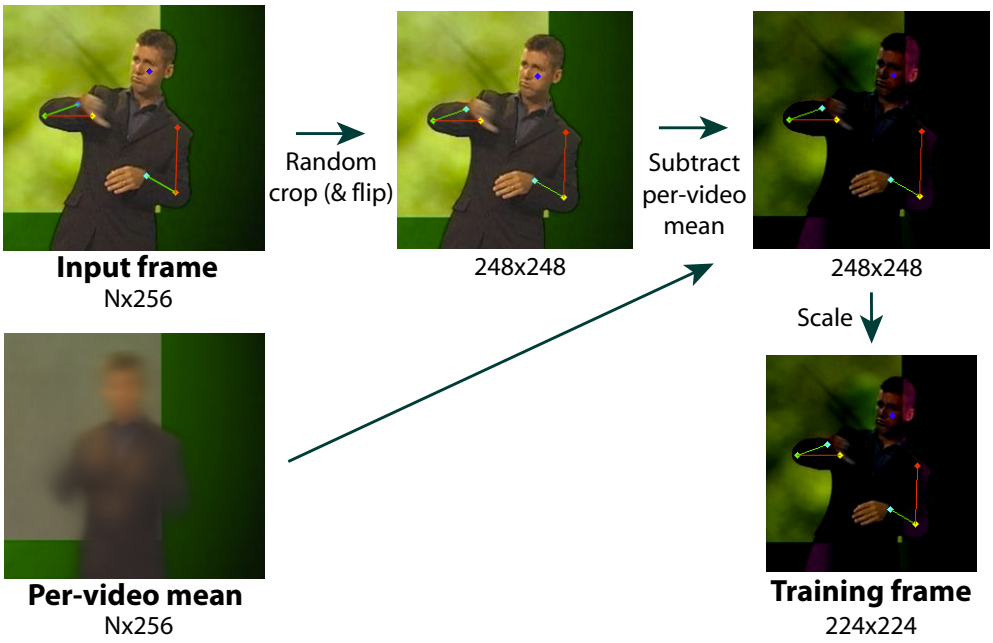


Figure 6.3: **Per-video mean and training augmentation.** At training time, the training data is augmented with random crops and flips. A per-video mean is computed from a subset of frames to provide some invariance to different background colours. The per-video mean is obtained once per video, and can be computed on-the-fly in online pose estimation scenarios.

in the evaluation section, yields an input representation for the ConvNet that generalises much better across different videos). This approach differs from static image ConvNets, which generally compute a mean image over the full dataset, and subtract the same mean from each input frame.

6.1.2 Implementation details

Training. The network weights are learnt using mini-batch stochastic gradient descent with momentum set to 0.9. Each iteration samples 256 training frames randomly across the training videos and uses them as a mini-batch. The input frames are rescaled to height 256. A 248×248 sub-image (of the $N \times 256$ input

image) is randomly cropped, randomly horizontally flipped and RGB jittered, and resized to 224×224. When training the ConvNet from scratch, the learning rate is set to 10^{-2}, and decreased to 10^{-3} at 80K iterations, to 10^{-4} after 90K iterations and stopped at 110K iterations. In the experiments in which we pretrain the weights on ImageNet ILSVRC-2012, learning rates are similarly decreased at 50K and 60K, and training is stopped at 70K iterations.

Testing. At test time, we crop the centre 248×248 of the input image, resize to 224×224 and feed forward through the network to obtain human joint location predictions. Test augmentation (*e.g.* computing the mean/median of predictions for 10 random image crops and flips, as done in classification ConvNet works) did not yield improved results over using the centre crop only.

Training time. Training was performed on a single NVIDIA GTX Titan GPU using a modified version of the Caffe framework [Jia, 2013]. Training the network from scratch took 3 days.

Size normalisation. Since the absolute image coordinates of the people vary across videos, we first normalise the training set with regards to a bounding box. The bounding boxes are estimated using a face detector: the estimated face bounding boxes are scaled by a fixed scaler (learnt from the training data such that joints in all training frames are contained within the bounding boxes). In the image domain, we crop out the bounding box, and rescale it to a fixed height. In the human joint domain, we rescale accordingly, and in addition re-normalise the

labels to the range $[0, 1]$. We found hyperparameter optimisation difficult without $[0, 1]$-normalised joints – in particular, the last fully-connected (regression) layer would require a different learning rate from other layers in order to converge.

Rescaling of input for multi-frame net. In practice, for the training of the multi-frame net to converge, input RGB values needed to be rescaled by the number of input frames to preserve the dynamic range of the hyperparameters.

6.2 Pose Estimation with a Deep Heatmap Network

This section shows that changing the regression target from the (x, y) *coordinates* of the joints to a *heatmap* of joint positions leads to significant boosts in performance. A description of this new network, and the reasons for this boost in performance, is provided below.

6.2.1 HeatmapNet

Our network (shown in Figure 6.4) is trained for regressing the location of the human joint positions. However, in contrast to Section 6.1 where we regressed the joint (x, y) positions directly, here we regress a *heatmap* of the joint positions, separately for each joint. This heatmap (the output of last convolutional layer, conv7) is a fixed-size $i \times j \times k$-dimensional cube (here $64 \times 64 \times 7$ for $k = 7$ upper-body joints). At training time, as the ground truth label, we synthesise a

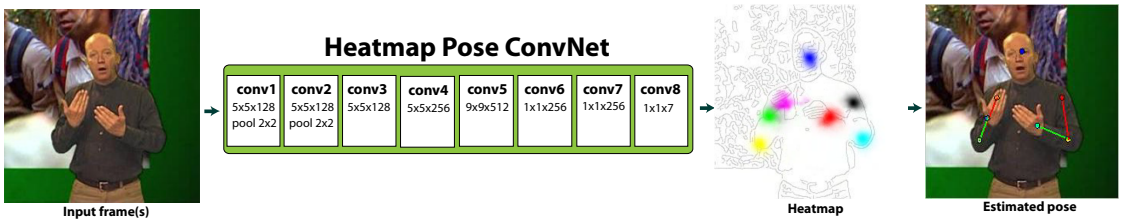


Figure 6.4: **Heatmap ConvNet architecture.**

heatmap for each joint separately by placing a Gaussian with fixed variance at the ground truth joint position (see Figure 6.5). We then use l_2 loss, which penalises the l_2 distance between the predicted heatmap and the synthesised ground truth heatmap.

We denote $(\mathbf{X}, \mathbf{y})$ as a training example, where $\mathbf{y}$ stands for the coordinates of the k joints in the image x. Given training data $N = \{\mathbf{X}, \mathbf{y}\}$ and the ConvNet regressor ϕ (output from conv7), the training objective becomes the task of estimating the network weights λ:

$$\arg\min_{\lambda} \sum_{(\mathbf{X},\mathbf{y})\in N} \sum_{i,j,k} \|G_{i,j,k}(\mathbf{y}_k) - \phi_{i,j,k}(\mathbf{X}, \lambda)\|^2 \tag{6.2}$$

where $G_{i,j,k}(\mathbf{y}_i) = \frac{1}{2\pi\sigma^2} e^{-[(y_k^1 - i)^2 + (y_k^2 - j)^2]/2\sigma^2}$ is a Gaussian centred at joint y_k with fixed σ.

We discuss implementation details (such as size normalisation of input data) in Sect 6.2.2

Discussion. The benefits of regressing a heatmap rather than (x, y) coordinates are twofold: first, one can understand failures and visualise the 'thinking process'

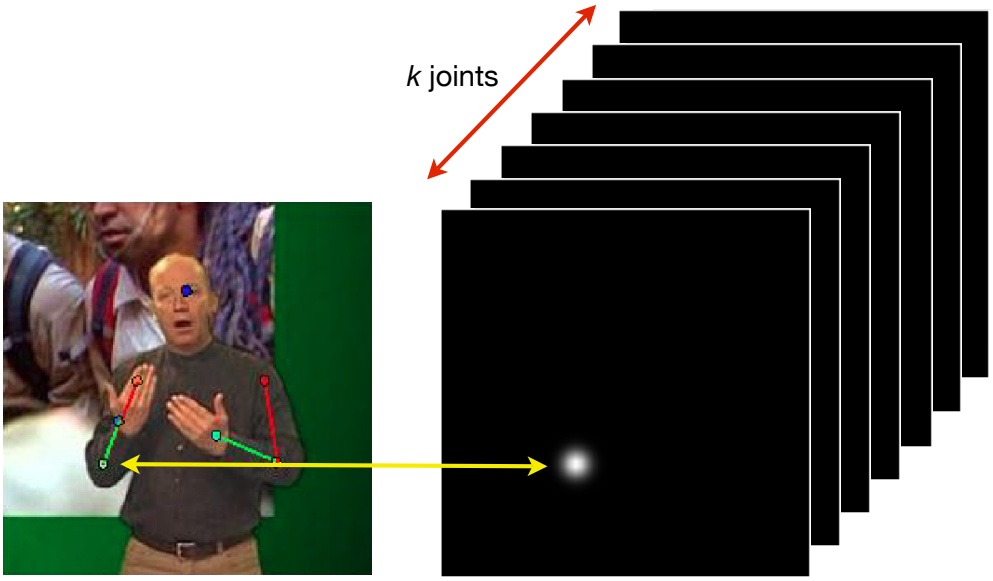


Figure 6.5: **Regression target for learning the Heatmap ConvNet.** The learning target for the convolutional network is (for each k joints) a heatmap with a synthesised Gaussian with a fixed variance centred at the ground truth joint position. The loss is the l_2 loss between this target and the output of the last convolutional layer.

of the network (see Figs 6.6 and 6.7); second, since by design, the output of the network can be multi-modal, *i.e.* allowed to have confidence at multiple spatial locations, learning becomes easier: early on in training (as shown in Figure 6.6), multiple locations may fire for a given joint (here the right wrist); the incorrect ones are then slowly suppressed as training proceeds. In contrast, if the output were only the wrist (x, y) coordinate, the net would only have a lower loss if it gets its prediction right (even if it was 'growing confidence' in the correct position). This is a highly non-linear and more difficult to learn mapping.

Architecture. The network architecture is shown in Figure 6.4, and sample activations for the layers are shown in Figure 6.7. To maximise the spatial resolution of the heatmap we make two important design choices: (i) minimal pooling

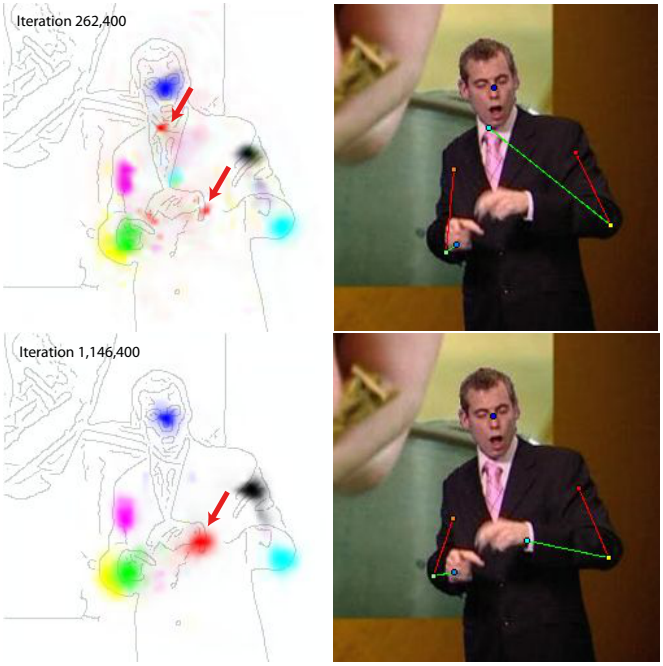


Figure 6.6: **Multiple peaks are possible with the Spatial Heatmap ConvNet.** Early on in training (top), multiple locations may fire for a given joint. These are then suppressed as training proceeds (bottom). The arrows identify two modes for the wrist; one correct, one erroneous. As the training proceeds the erroneous one is diminished.

is used (only two 2×2 max-pooling layers), and (ii) all strides are unity (so that the resolution is not reduced). All layers are followed by ReLUs except conv9 (the pooling layer). Our network architecture differs from both AlexNet and recent pose nets [Tompson et al., 2015]. In contrast to AlexNet [Krizhevsky et al., 2012], our network is fully convolutional (no fully-connected layers) with the fully-connected layers of [Krizhevsky et al., 2012] replaced by 1×1 convolutions. In contrast to both AlexNet and [Tompson et al., 2015], our network is deeper, does not use local contrast normalisation (as we did not find this beneficial), and utilises less max-pooling.

6.2.2 Implementation details

The implementation details are the same as in Section 6.1.2 except:

Training. The network weights are learnt using mini-batch stochastic gradient descent with momentum set to 0.95. Each iteration samples 64 training frames randomly across the training videos and uses them as a mini-batch. All layers are followed by ReLUs except conv8 (the pooling layer); no dropout is used. A 248×248 sub-image (of the $N \times 256$ input image) is randomly cropped, randomly horizontally flipped and RGB jittered, and resized to 256×256. The variance of the Gaussian is set to $\sigma = 1.5$ with an output heatmap size of 64×64. The learning rate is set to 10^{-4}, and decreased to 10^{-5} at 80K iterations, to 10^{-6} after 100K iterations and stopped at 120K iterations.

Training time. Training was performed on four NVIDIA GTX Titan GPUs using a modified version of the Caffe framework [Jia, 2013] with multi-GPU support. Training the network from scratch took 3 days.

6.3 Heatmap Network with Optical Flow

With the above heatmap network, we can use the same idea as in Chapter 5 in the context of deep nets: use optical flow to *warp* pose predictions from neighbouring frames. These warped heatmaps represent a strong set of 'expert opinions' (with corresponding confidences) for frame t, from which joint positions can be more

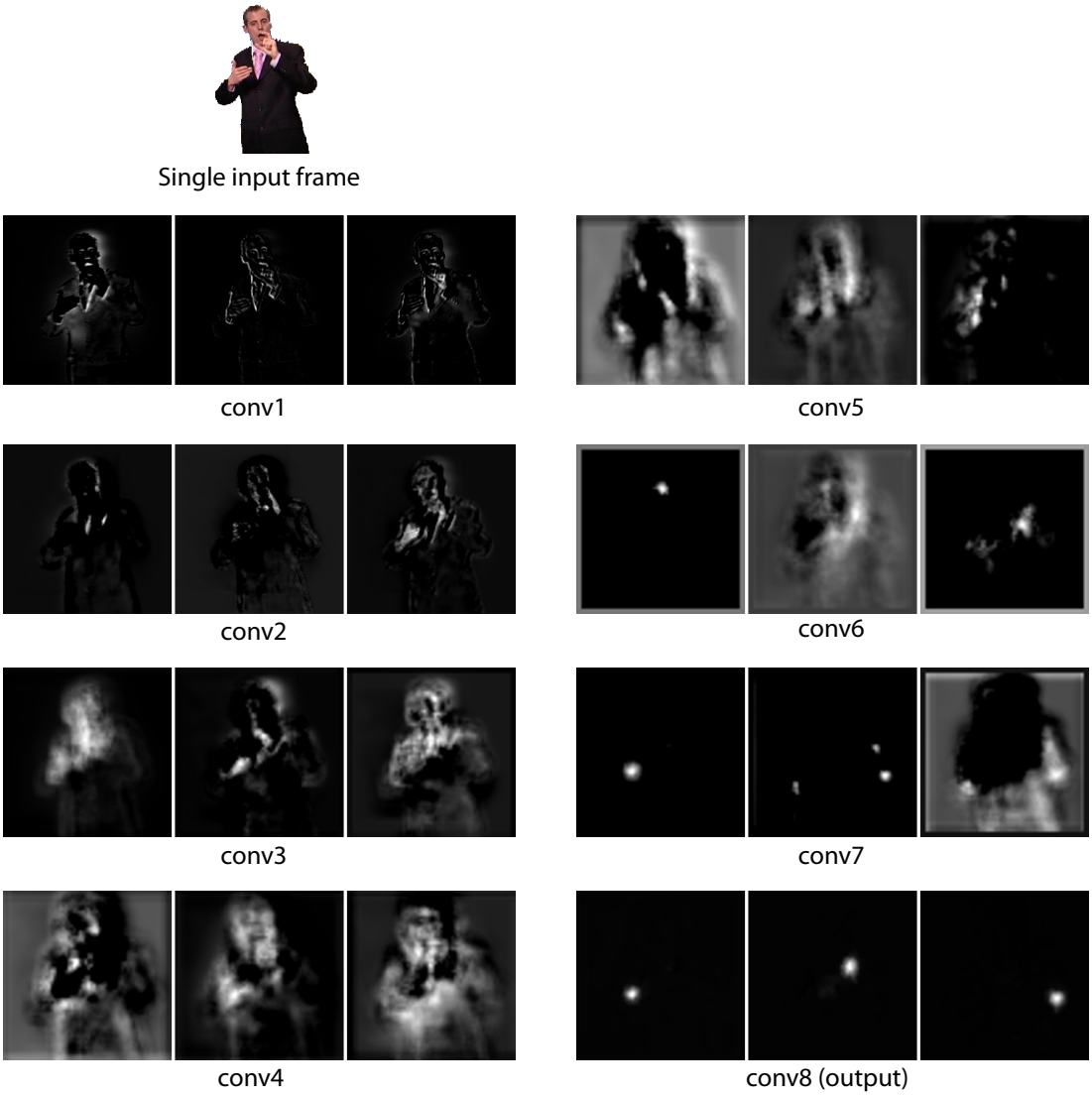


Figure 6.7: **Sample activations for convolutional layers.** Neuron activations are shown for three randomly selected channels for each convolutional layer (resized here to the same size), with the input (pre-segmented for visualisation purposes) shown above. Low down in the net, neurons are activated at edges in the image (*e.g.* conv1 and conv2); higher up, they start responding more clearly to body parts (conv6 onwards). The outputs in conv8 are shown for the right elbow, left shoulder and left elbow.

precisely estimated than when only using a single heatmap.

We further show that a *parametric pooling* of these warped heatmaps can be learnt, effectively learning how to weigh the expert opinions for different neigh-

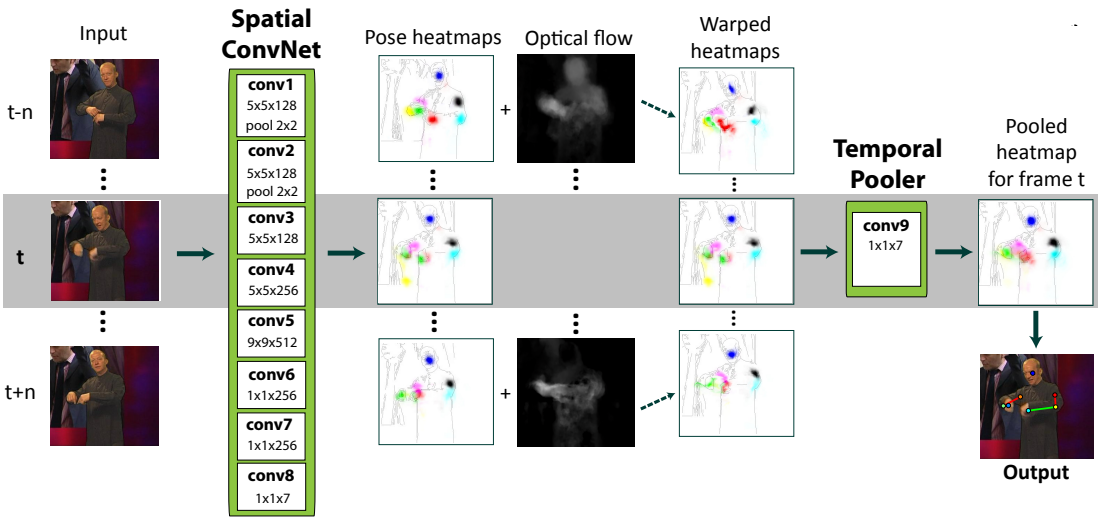


Figure 6.8: **Deep expert pooling architecture for pose estimation.** The network takes as an input all RGB frames within a n-frame neighbourhood of the current frame t. The fully convolutional Spatial ConvNet (consisting of 8 convolutional layers) predicts a confidence heatmap for each body joint in these frames (shown here with different colour per joint). These heatmaps are then temporally *warped* to current frame t using optical flow. The warped heatmaps (from multiple frames) are then *pooled* with another convolutional layer (the temporal pooler), which learns how to weigh the warped heatmaps from nearby frames. The final body joints are selected as the maximum of the pooled heatmap (illustrated here with a skeleton overlaid on top of the person).

bouring frames. This is in contrast to Chapter 5, where we simply summed the warped heatmaps.

We show that this method outperforms all other methods in this thesis by a large margin.

Related work. Temporal information in videos was initially used with ConvNets for action recognition [Simonyan and Zisserman, 2014], where optical flow was used as an input feature to the network. Following this work, we and [Jain et al., 2014b] investigated the use of temporal information for pose estimation in

a similar manner, by inputting flow or RGB from multiple nearby frames into the network, and predicting joint positions in the current frame. However, pose estimation differs from action recognition in a key respect which warrants a different approach to using optical flow: in action recognition the prediction target is a class label, whereas in pose estimation the target is a set of (x, y) positions in the image. Since the targets are positions in the image space, one can use dense optical flow vectors not only as an input feature but also to *warp predicted positions* in the image. To this end, our work explicitly predicts joint positions for *all* neighbouring frames, and temporally aligns them to frame t by warping them backwards or forwards using tracks from dense optical flow. This effectively reinforces the confidence in frame t with a strong set of 'expert opinions' from neighbouring frames, from which joint positions can be more precisely estimated.

6.3.1 Deep expert pooling architecture

Figure 6.8 shows an overview of our ConvNet architecture. Given a set of input frames within a temporal neighbourhood of n frames from a frame t, a spatial ConvNet regresses joint confidence maps ('heatmaps') for each input frame separately. These heatmaps are then individually *warped* to frame t using dense optical flow. The warped heatmaps (which are effectively 'expert opinions' about joint positions from the past and future) are then pooled into a single heatmap for each joint, from which the pose is estimated as the maximum.

Given the heatmaps from the Spatial ConvNet from multiple frames, the heatmaps are reinforced with optical flow. This is done in three steps: (1) the confidences

from nearby frames are aligned to the current frame using dense optical flow; (2) these confidences are then *pooled* into a composite confidence map using an additional convolutional layer; and (3) the final upper body pose estimate for a frame is then simply the positions of maximum confidence from the composite map. Below we discuss the details of the first two steps.

Step 1: Warping confidence maps with optical flow. For a given frame t, as in Chapter 4, pixel-wise temporal tracks are computed from all neighbouring frames within n frames from $((t-n)$ to $(t+n))$ to frame t using dense optical flow, and these optical flow tracks are used to warp confidence values in neighbouring confidence maps to align them to frame t. In practice, single-frame optical flow is computed for all consecutive frames in the video; given a frame t, the flow from $(t-n)$ to t is then simply obtained as a sum of the flow vectors at frames $(t-n)\ldots(t-1)$ (and is then used to perform the warping separately for each neighbouring frame).

Step 2: Pooling the confidence maps. The output of Step 1 is a set of confidence maps that are warped to frame t. From these 'expert opinions' about the joint positions, the task is first to select a confidence for each pixel for each joint, and then to select one position for each joint. One solution, as described in Chapter 4, would be to simply average the warped confidence maps. However, not all experts should be treated equally: intuitively, frames further away (thus with more space for optical flow errors) should be given lower weight.

To this end we learn a parametric pooling layer that takes as an input a set of

warped heatmaps for a given joint, and as an output predicts a single 'composite heatmap'. The input to this layer is a $i \times j \times t$ heatmap volume, where t is the number of warped heatmaps (*e.g.* 31 for a neighbourhood of $n = 15$). As the pooling layer, we train a 1×1 kernel size convolutional layer for each joint. This is equivalent to cross-channel weighted sum-pooling, where we learn a single weight for each input channel (which correspond to the warped heatmaps). In total, we therefore learn $t \times k$ weights (for k joints).

6.3.2 Implementation details

A temporal neighbourhood of $n = 15$ is input into the parametric pooling layer. Optical flow is computed using FastDeepFlow [Weinzaepfel et al., 2013] with Middlebury parameters. For training the network, we first pre-train the HeatmapNet, then attach the pooling layer and backpropagate through the whole network. The warping is implemented as a Caffe layer.

6.4 Experiments

We first define the evaluation protocol and details, then present comparisons to alternative network architectures, then provide an in-depth analysis of alternative CoordinateNet architectures, and finally give a comparison to state of the art.

Example predictions of the HeatmapNet are shown in Figures 6.15, 6.18, 6.20 and 6.21.

6.4.1 Evaluation protocol and details

Evaluation protocol. In all pose estimation experiments we compare the estimated joints against frames with manual ground truth. We present results as graphs that plot accuracy vs distance from ground truth in pixels. A joint is deemed correctly located if it is within a set distance of d pixels from a marked joint centre in ground truth. Unless otherwise stated, the experiments use $d = 6$.

Experimental details. All frames of the videos are used for training (with each frame randomly augmented as detailed above). The frames are randomly shuffled prior to training to present maximally varying input data to the network.

The hyperparameters (early stopping *etc.*) are estimated using the validation set.

6.4.2 Net architecture comparison

Figure 6.9 shows results for our methods on wrists in the BBC Pose and Extended BBC Pose datasets. With the HeatmapNet, we observe a significant boost in performance (6.6% – 79.6% to 86.1% at $d = 6$) by automatically obtaining additional labelled training samples using the tracker from Chapter 4. In contrast, the CoordinateNet is unable to make effective use of this additional training data.

We observe a further boost in performance from using optical flow (2.6% – 86.1% to 88.7% at $d = 6$). Figure 6.10 shows the learnt pooling weights. We see that for this dataset, as expected, the network learns to weigh frames temporally

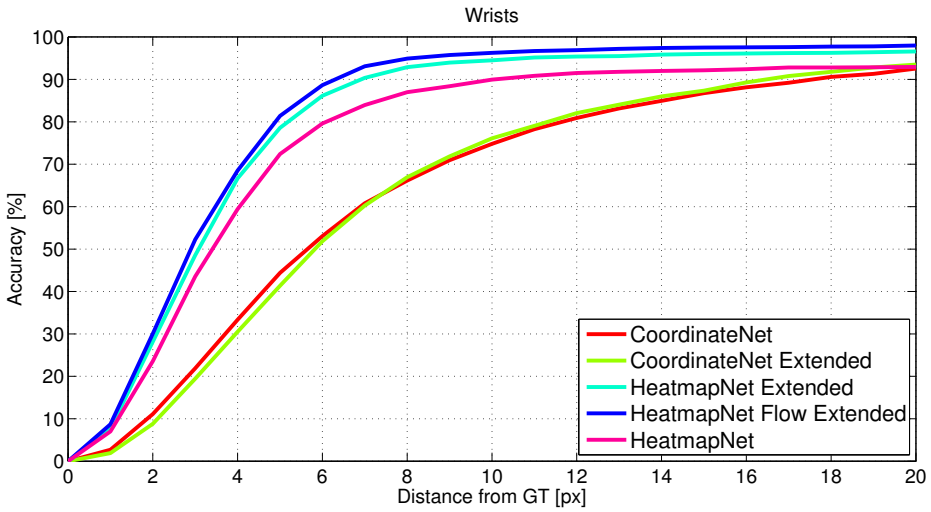


Figure 6.9: **Net performance comparison for wrists on (Extended) BBC Pose.** CoordinateNet is the network from the previous section; HeatmapNet is the heatmap network; and HeatmapNet Flow is the heatmap network with the parametric pooling layer. 'Extended' indicates that the network is trained on Extended BBC Pose instead of BBC Pose. We observe a significant gain going from CoordinateNet to HeatmapNet; for the HeatmapNet from using additional (automatically labelled – see Chapter 3) training data, and a further boost from using optical flow information (and selecting the warping weights with the parametric pooling layer). Plots show accuracy per joint type (average over left and right body parts) as the allowed distance from manual ground truth is increased.

close to the current frame higher (presumably because they contain less errors in optical flow).

Figure 6.11 shows a comparison of different pooling types (for cross-channel pooling). We compare learning a parametric pooling function to sum-pooling and to max-pooling (maxout Goodfellow et al. [2013]) across channels. As expected, parametric pooling performs best, and improves as the neighbourhood n increases. In contrast, results with both sum-pooling and max-pooling deteriorate as the neighbourhood size is increased further, as they are not able to down-weigh predictions that are further away in time (and thus more prone

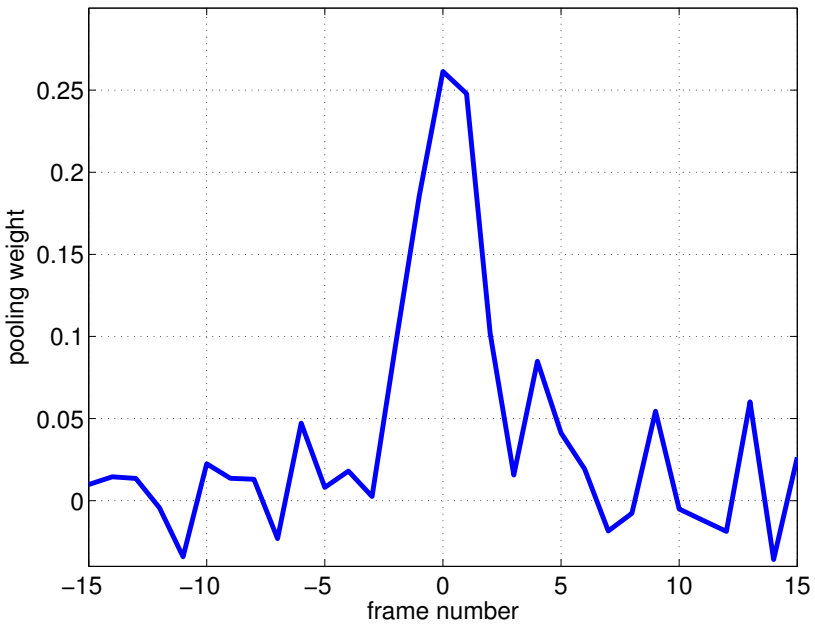


Figure 6.10: **Learnt pooling weights for BBC Pose with $n = 15$.** Weights shown for the right wrist. The centre frame receives highest weight.

to errors in optical flow). As expected, this effect is particularly noticeable for max-pooling.

As shown in Figure 6.12, we found the CoordinateNet to perform badly for wrists, but, for this particular dataset, to perform well for elbows and shoulders (not shown), which move less, and thus are easier to learn. We therefore use this network for elbows and shoulders in Figure 6.13.

We also evaluate the effect of pre-segmenting the foreground of the video with the HeatmapNet in Figure 6.12. Foreground segmentation makes no difference for wrists, but using foreground segmentations leads to a boost for elbows, and a small boost for shoulders.

The HeatmapNet and CoordinateNet differ in two ways: (i) loss target (heatmap

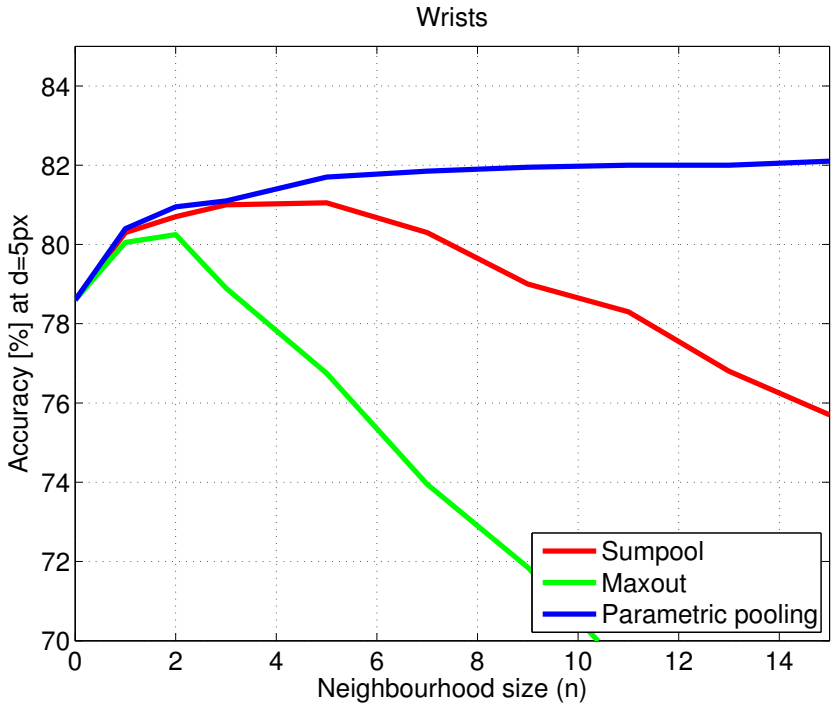


Figure 6.11: **Comparison of pooling types.** Results are shown for wrists in BBC Pose at threshold $d = 5$px. Parametric pooling (learnt cross-channel pooling weights) performs best.

vs coordinates), and (ii) network architecture ([Sermanet et al., 2014] vs the new heatmap architecture). We note here that we also observed a significant boost in performance when 'convolutionising' the CoordinateNet architecture (replacing the last fully-connected layers with 1×1 convolution layers, reducing max-pooling and changing the target to a heatmap), but this did not obtain as good performance as our HeatmapNet architecture.

Failure modes. The main failure case occurs when there are multiple modes in the heatmap, and the wrong one is selected (shown in Figure 6.19). This could be addressed by adding a spatial model on top of the heatmap net.

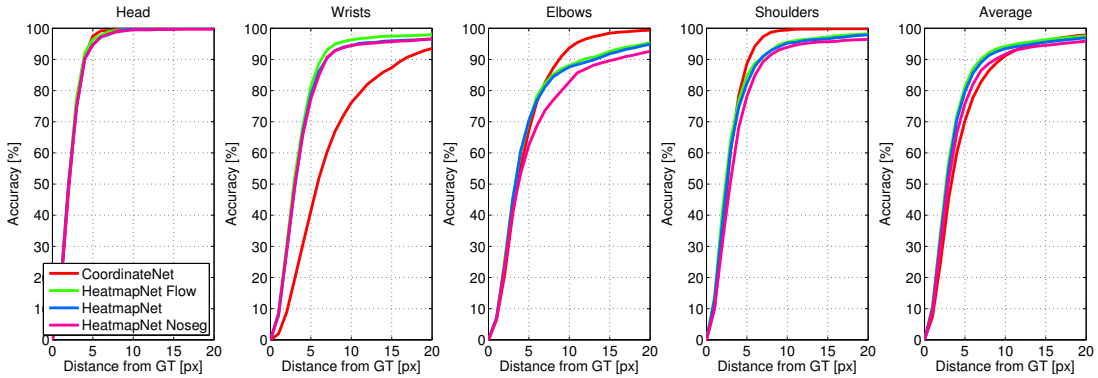


Figure 6.12: **Net performance comparison for CoordinateNet, HeatmapNet, HeatmapNet without foreground segmentation (HeatmapNet Noseg), and HeatmapNet with optical flow (HeatmapNet Flow).** For wrists, HeatmapNet with and without foreground segmentations perform equally well (the lines in the graph cover each other). Results are shown for Extended BBC Pose. Plots show accuracy per joint type (average over left and right body parts) as the allowed distance from manual ground truth is increased.

6.4.3 In-depth evaluation of CoordinateNet architectures

We will next report a series of experiments on the CoordinateNet architecture.

Table 6.1 shows comparisons to CoordinateNets with different number of input frames, input representations, levels of preprocessing and pretraining. We next discuss these results in detail.

1. Ridge regression from ImageNet fully-connected layer features. ConvNets trained on ImageNet have been shown to generalise extremely well to a wide variety of classification tasks [Chatfield et al., 2014, Razavian et al., 2014, Zeiler and Fergus, 2014]. Often the state of the art can be achieved by simply using the output from one of the fully connected layers as a feature. Here we investigate the performance of this approach for pose estimation.

We extract features from the last fully connected layers of a network trained on ImageNet (a 4096 dimensional feature) applied to the original BBC TV broadcast dataset, and learn a ridge regressor. As shown in Table 6.1(row 1), this performs poorly. This implies that these features that are extremely powerful for various real-world image classification tasks may not be quite as powerful when it comes to predicting precise locations of parts with high appearance variation in an image.

2. ImageNet pretraining/fine-tuning. Here we first pretrain network weights on ImageNet ILSVRC-2012, and then fine-tune them on BBC pose. This performs better than ridge regression from the output of the fully connected layers, but still does not match the performance when the network is trained from scratch.

3. Single mean image. We investigate using a single mean image for all training and testing (instead of our idea of using a per-video mean). When using a single mean image computed off the whole dataset, the average evaluation measure drops from 72.0% to 57.6%. The drop is caused by nearly completely failed tracking in some test videos. The network overfitted to the backgrounds in the training data, and did not generalise to videos with different static backgrounds (as is the case for some of the test videos).

4. Smaller ConvNet. We compare the CoordinateNet architecture to a smaller (and slightly faster) network (set up with same architecture as the "CNN-M" network in [Chatfield et al., 2014] – *i.e.*, same depth as our other network,

Training	Aug	Multi	Seg	Head	Wrists	Elbows	Shoulder	Average
Last only	✓			15.4	5.8	8.4	18.3	12.0
FT all	✓			95.6	44.0	53.6	80.8	68.5
Scratch				94.3	52.1	51.9	87.9	71.5
Scratch	✓			95.9	47.1	56.0	89.1	72.0
Scratch	✓	✓		95.6	50.1	58.1	89.5	73.3
Scratch	✓		✓	96.1	58.0	66.8	91.2	78.0

Table 6.1: **Evaluation of different CoordinateNet architectures.** The evaluation measure is the percentage of predictions within 6 pixels from ground truth. 'Scratch/Last only/FT all' refer to training from scratch/training the last layer from scratch (keeping the rest of the ImNet-pretrained network fixed)/finetuning all layers of an ImNet-pretrained network; 'Aug' to training time augmentation; 'Multi' to using multiple input frames; and 'Seg' to using an input representation with the foreground pre-segmented.

"CNN-S", but with fewer parameters). This performs slightly worse than the larger network used in this work (70.4% vs 72.0%).

5. No training augmentation. As shown in Table 6.1(row 3), training augmentation yields a small improvement over no training augmentation (using the centre crop of the image only).

6. Multi-frame net. Here we test the improvement from using multiple input frames. Table 6.1(row 5) shows a consistent performance improvement over wrists, elbows and shoulders, with a particularly noticeable improvement in wrist predictions. The head predictions are slightly worse, likely because the head is fairly stationary and hence does not benefit from the additional temporal information.

The multi-frame network has two parameters: the number of input frames m (how many frames are used as input for each pose estimate) and the temporal

spacing t between the input frames (time between each of the n input frames). In a parameter optimisation experiments we searched over $m = \{1, 3, 5\}$ and $t = \{1, 2, 3, 5, 8, 10, 15, 25\}$ on the validation set. $m = 3$ and $t = 1$ (three input frames with one-frame time spacing) were selected as the optimal parameters. We also explored using difference images (subtracting the current frame from the additional input frames), however this did not improve performance.

7. Foreground-segmented input. We test our approach with the input representation described in Chapter 4, where we pre-segmented the foreground of the input frames, and black out the background. As shown in Table 6.1(row 6), even though our method does not require foreground segmentations, it does benefit from using them. On the downside, using them would require manual tuning of segmentation parameters and significantly slow down the runtime (from 100fps to around 5fps).

6.4.4 Comparison to previous work

We next turn to comparing all our network architectures to previous work.

BBC Pose. Figure 6.13 shows a comparison to state of the art on the BBC Pose dataset. We compare against all previous reported results on the dataset. These include [Buehler et al., 2011]; the Random Forest from Chapter 4; the Sequential Forest from Chapter 5; and [Yang and Ramanan, 2013]'s deformable parts-based model.

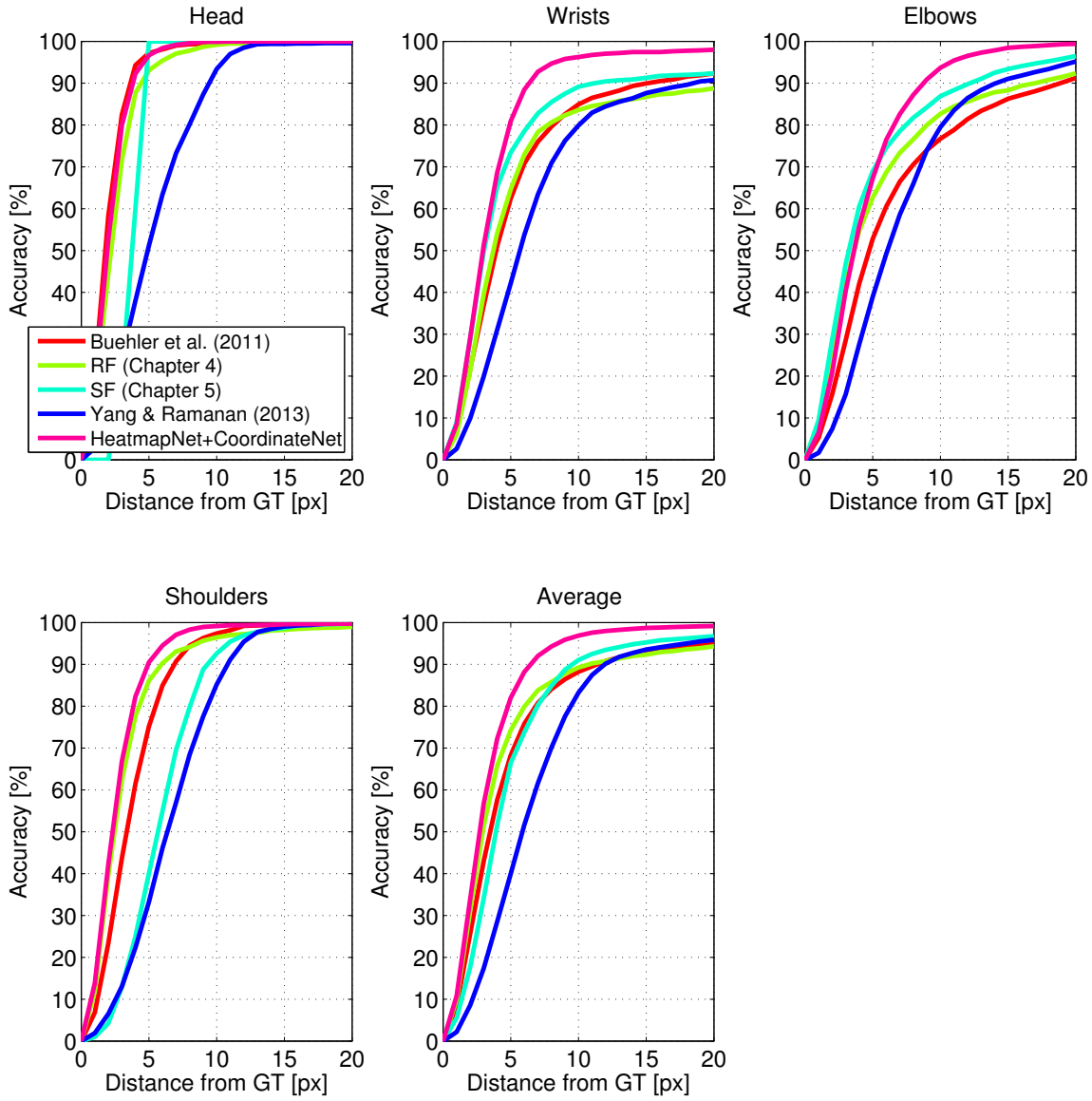


Figure 6.13: **Comparison to previous work.** We outperform all previous work by a large margin; notice particularly the performance for wrists, where we outperform the best competing method by over 10% at $d = 6$. Our method (HeatmapNet+CoordinateNet) uses CoordinateNet Extended for elbows and shoulders, and HeatmapNet Flow Extended for head and wrists. CoordinateNet uses Extended BBC Pose for training; Buehler *et al.*, RF, SF and Yang & Ramanan use BBC Pose. Plots show accuracy per joint type (average over left and right body parts) as the allowed distance from manual ground truth is increased.

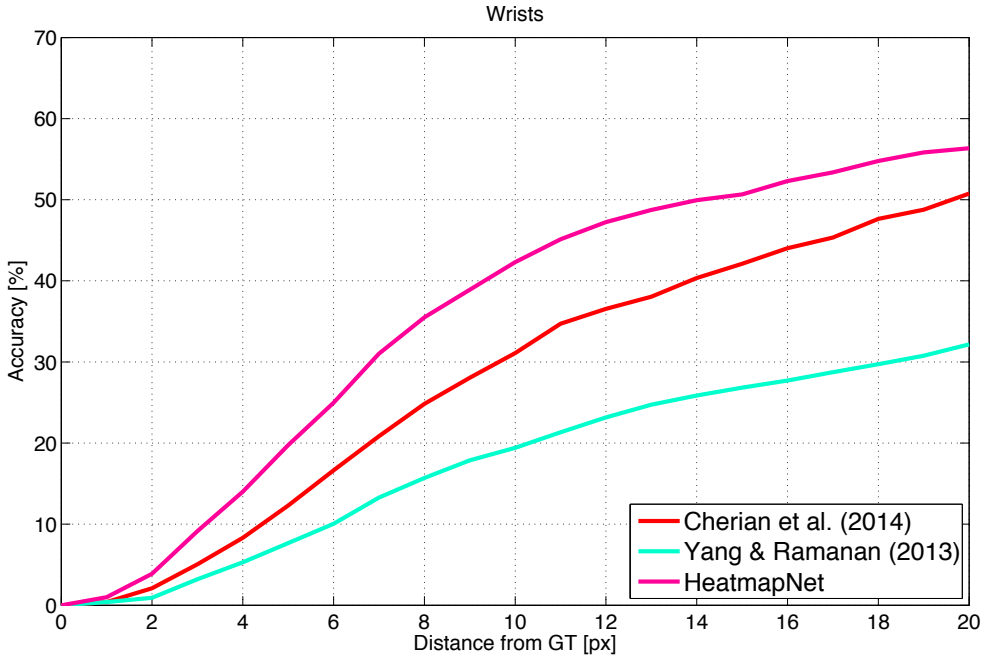


Figure 6.14: **Comparison to state of the art on Poses in the Wild.** Plot shows accuracy for wrists as the allowed distance from manual ground truth is increased. We note that our method outperform state of the art by a large margin (12% at $d = 10$).

We outperform all previous work by a large margin, with a particularly noticeable gap for wrists (10% improvement compared to SF at $d = 6$). Our method results on average in much better constrained poses and a significantly higher area under the curve, without requiring any of the video-specific manual segmentation algorithm tuning (as in Chapters 4 and 5) or manual annotation (as in [Buehler et al., 2011]). We hypothesise that the better constrained poses are due to the ConvNet learning constraints for what poses a human body can perform (*i.e.*, the constraints of the human kinematic chain). Further, the ConvNets rarely predicts incorrect joint positions in the background (Figure 6.17 shows examples of corrected frames). We conjecture that this is due to the high capacity of the model, which enables it to learn to ignore any pixels in the background.

Figure 6.15: **Example predictions on a variety of videos in Poses in the Wild (HeatmapNet).**

Poses in the Wild. Figure 6.14 shows a comparison to the state of the art on Poses in the Wild. We replicate the results of the previous state of the art method using code provided by the authors Cherian et al. [2014]. We outperform the state of the art on this dataset by a large margin (by 12% at $d = 10$).

6.4.5 Computation time

Figure 6.16 shows a comparison of the computation time of our methods to previous work. The computation times are measured on a 16-core 3.3GHz Intel Xeon E5-2667 CPU and a GeForce GTX Titan GPU. We improve by an order of magnitude over previous methods using the same CPU. If we instead predict with a single GPU, performance increases by yet another order of magnitude.

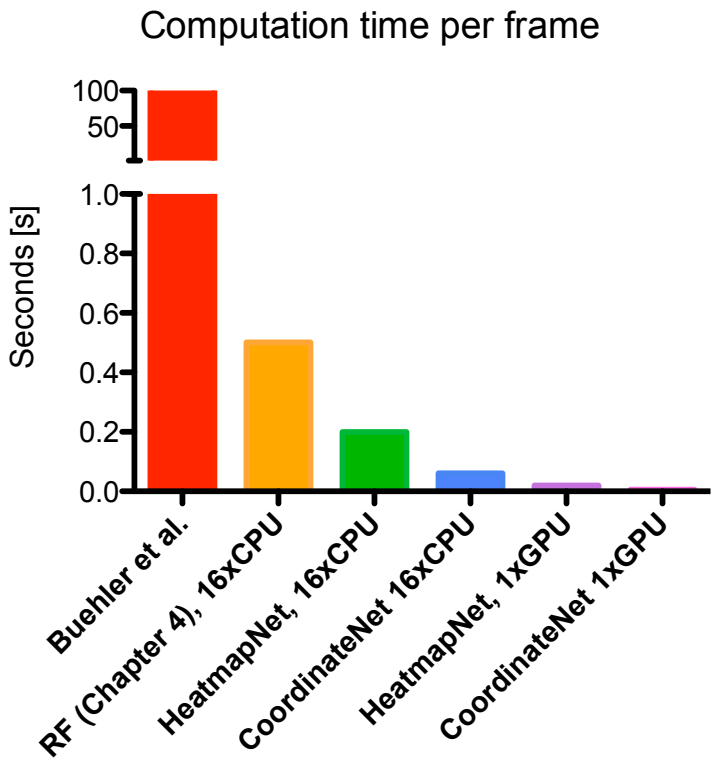


Figure 6.16: **Computation time.** Comparison of computation times versus the method in Chapter 4 and [Buehler et al., 2011]. Note that the reliable semi-automatic method of Buehler *et al.* is off the scale as computing a pose for a single frame takes around 100s. Our method outperforms previous methods by over an order of magnitude using the same hardware. Using a single GPU instead of CPUs increases speed by another order of magnitude.

6.5 Conclusion

We have presented two types of convolutional networks, one that regresses coordinates and another that regresses confidence maps. We have shown that both of these significantly outperform the trackers in earlier chapters, and that combining them further improves performance. Finally, we have demonstrated how a parametric pooling layer can be used in conjunction with optical flow to effectively learn to 'pool confidence' from neighbouring frames, yielding further

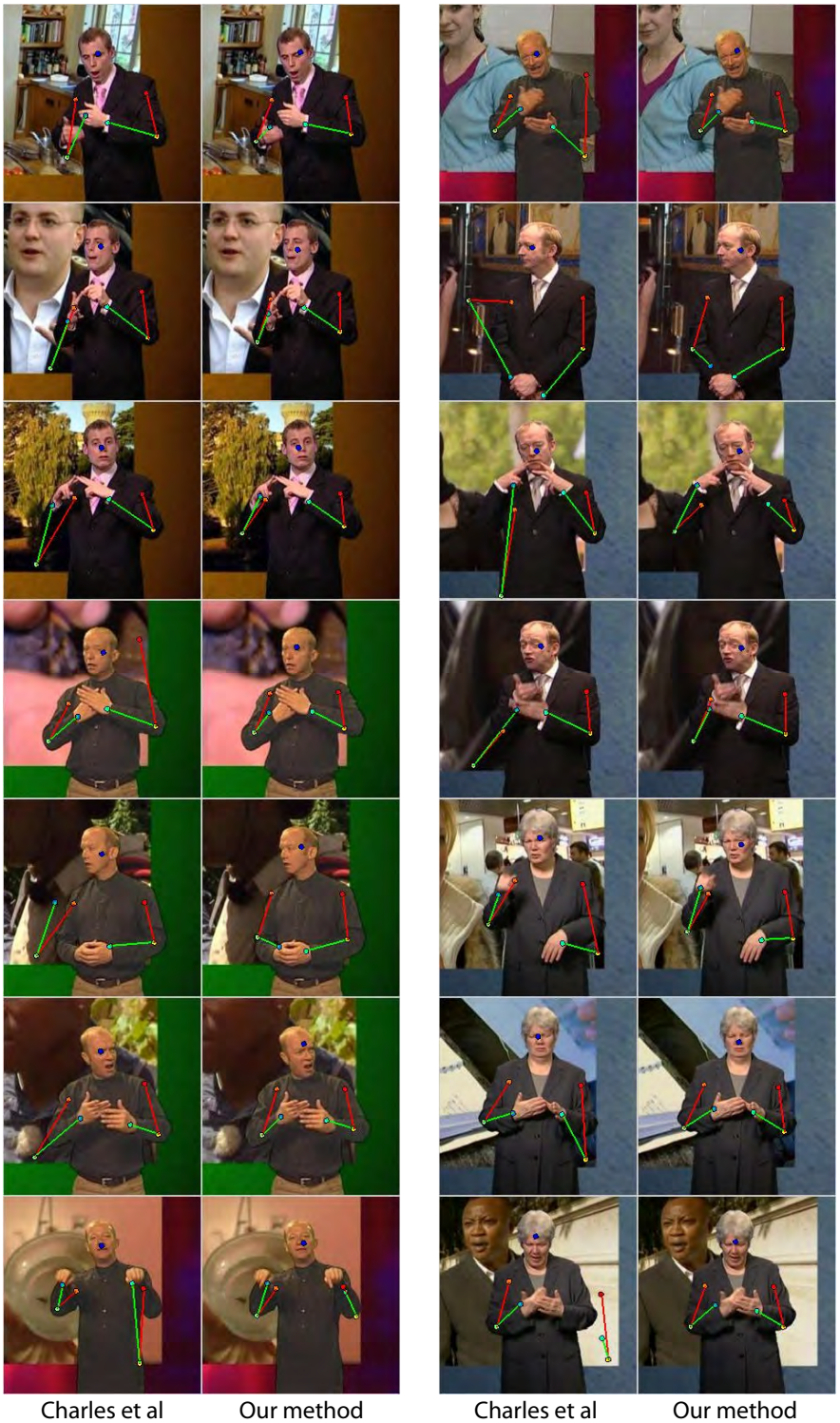


Figure 6.17: **Example test set frames comparing the CoordinateNet pose estimates to Chapter 4.** The pose estimates here are much better localised.

Figure 6.18: **Example predictions on two videos in Poses in the Wild (HeatmapNet).** Predictions and the corresponding heatmaps are shown.

improvements upon the performance of the heatmap network.

In the work on optical flow, we have demonstrated the idea of reinforcing position predictions for the application of human pose estimation. However, we note that this idea is more generally applicable to predicting positions of arbitrary objects in videos: for other objects (than human shoulders, elbows, wrists and head), confidences from neighbouring frames can likewise be propagated to reinforce predictions.

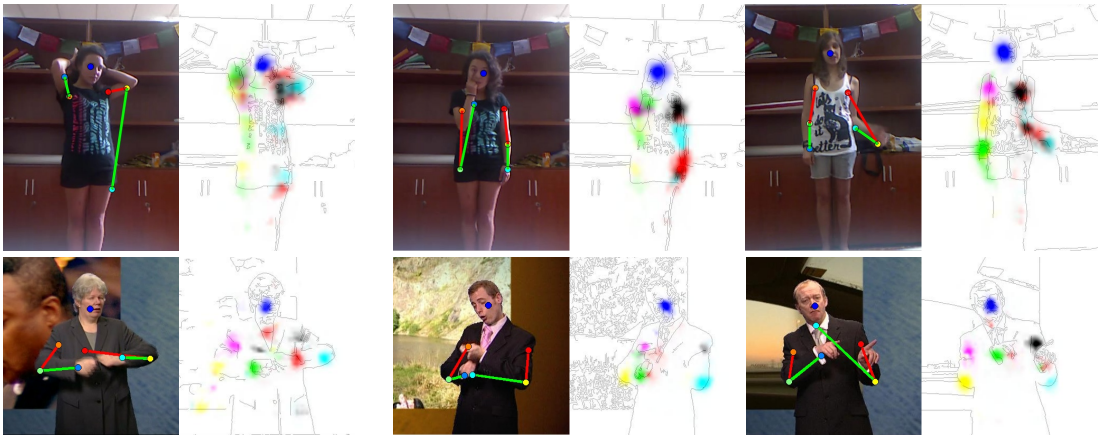


Figure 6.19: **Failure cases.** As shown, failure cases contain multiple modes for the same joint in the heatmap (and the wrong mode has been selected). This could be addressed with spatial model.

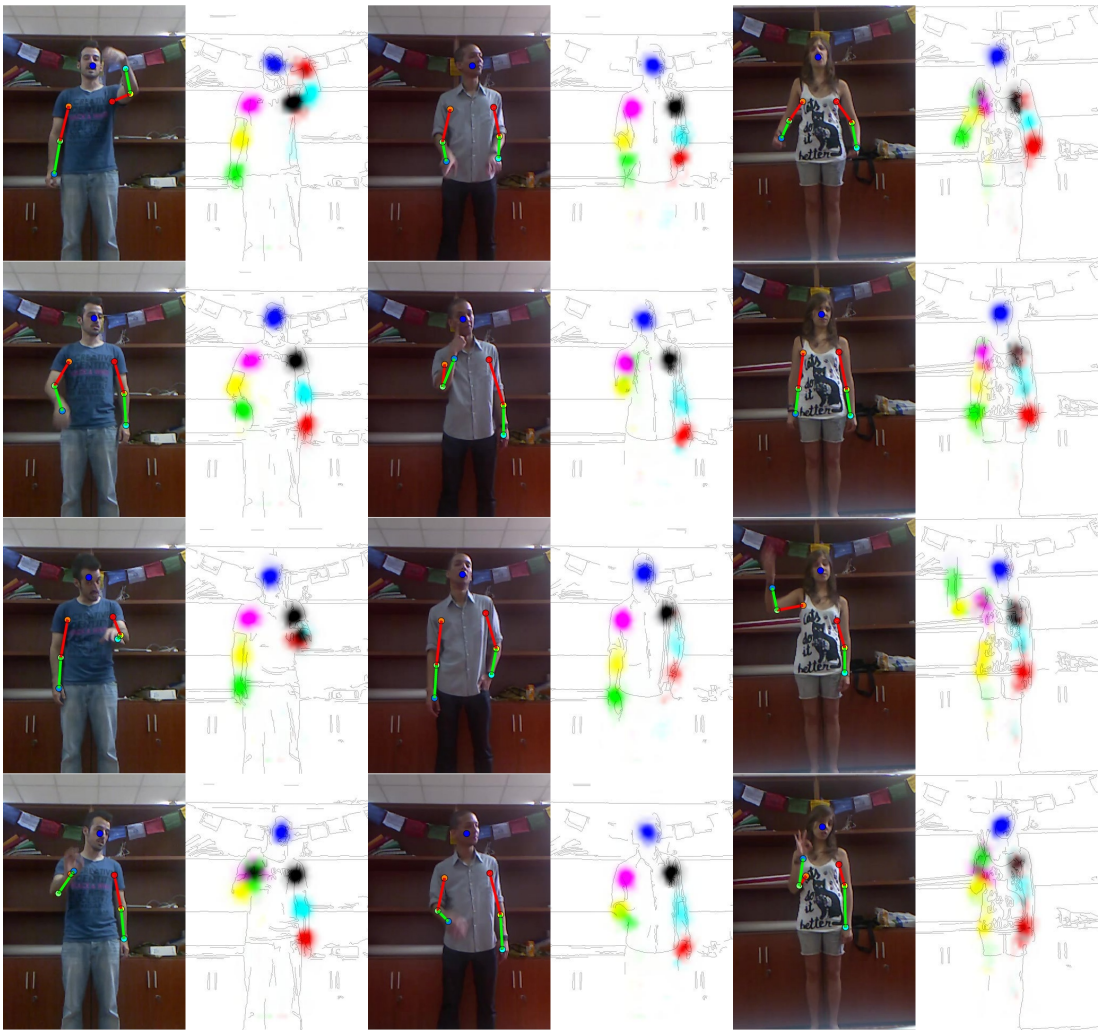


Figure 6.20: **Example predictions on ChaLearn (HeatmapNet Flow).**

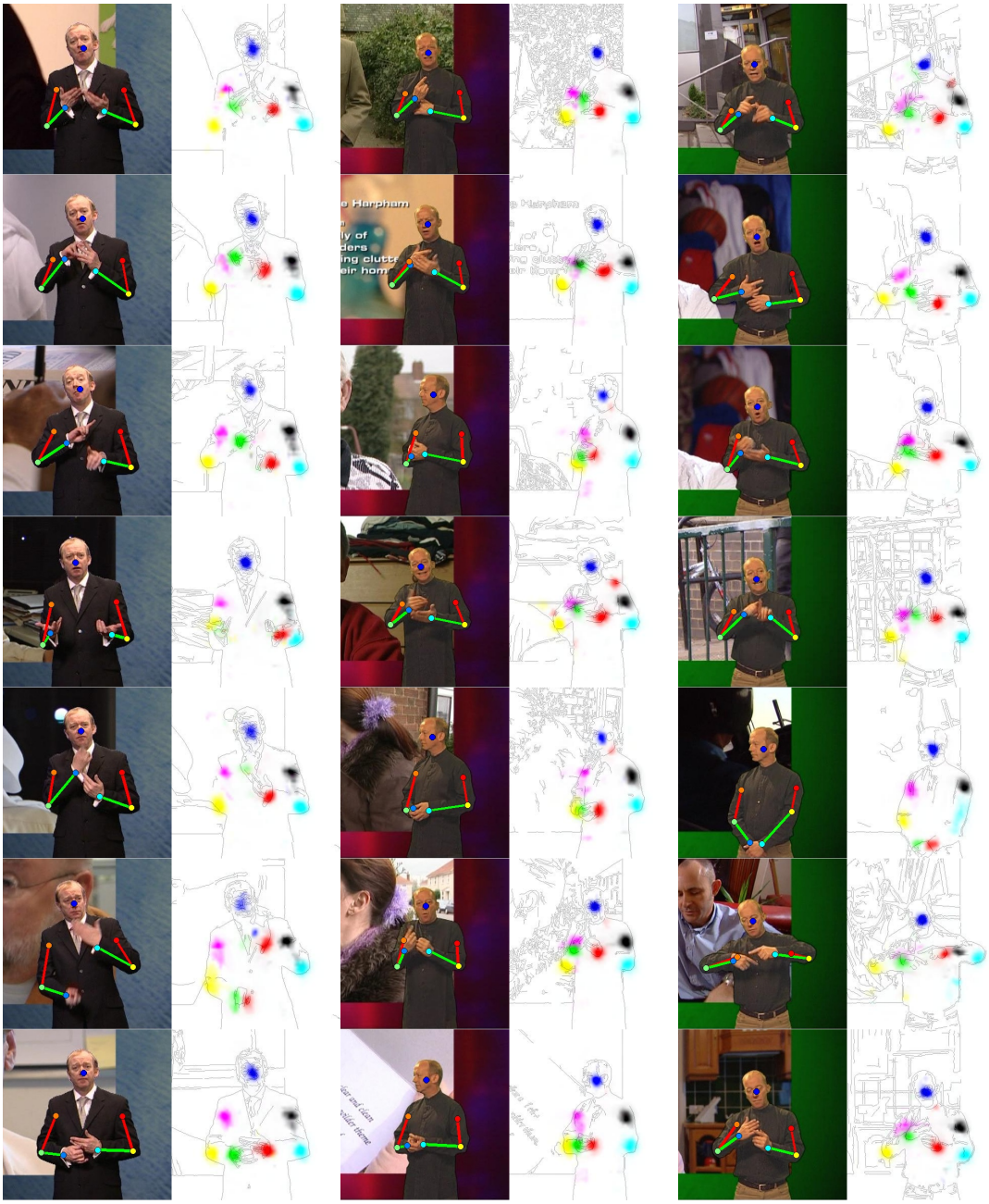


Figure 6.21: **Example predictions on BBC Pose (HeatmapNet Flow).**

Part II

Learning Gestures from Weak Supervision

Chapter 7

Learning Sign Language by Watching TV

In this chapter, we show that, given the accurate upper-body tracker from Part I, we can learn sign language automatically from TV broadcasts using the weak supervision from subtitles.

In particular, we show that we can learn signs by using an English word to select a set of subtitles (and associated videos) that contain the word (the positive sequences) and another set of videos that do not contain the word (the negative sequences), and then looking for signs that occur frequently in the positive sequences, but not in the negative sequences.

This is however challenging in practice because the subtitle supervision is both *weak* (in temporal alignment: a sign (8–13 frames) can be anywhere in the positive sequences (400 frames)), and *noisy* (a word occurring in the subtitle only implies

the sign is *usually* signed (in about 60% of cases)).

To solve this sign-subtitle word correspondence problem, we make the following contributions:

1. Multiple Instance Learning (MIL) for automatically extracting signs. We develop a novel MIL [Dietterich et al., 1997, Maron and Lozano-Pérez, 1998] method using an efficient discriminative search, which determines a candidate list for the sign with both high recall and precision. The training data here are visual descriptors (hand trajectories), with weak supervision from subtitles. (Section 7.2)

2. *Mouthing* for isolating and learning signs. We show that, somewhat counter-intuitively, mouth patterns are highly informative for isolating words in a language for the Deaf, and their co-occurrence with the hand motion in signing can be used to significantly reduce the correspondence search space. This is because signers often mouth the word that they are signing, which is an important cue that has been overlooked in previous work. (Section 7.1)

Our method (shown applied on an example video in Figure 7.1) consists of three steps: (i) the search space for correspondences is significantly reduced by detecting mouth movement, and filtering away irrelevant temporal intervals (ones that do not contain mouth movement) ; (ii) candidates for the signs are obtained using an efficient discriminative search (based on temporal correlation scores) over all remaining sequences; and finally (iii) these candidates are selected or rejected using the MIL Support Vector Machine (MI-SVM [Andrews et al., 2002])

framework. Figure 7.2 describes this processing pipeline in more detail.

In evaluations, we demonstrate that with the reduction in search space and discriminative learning, simple features (namely only hand trajectories) are sufficient to successfully extract the signs. We show that this method achieves superior results to previous work, and does so at a much lower computational cost.

Related work. The closest work to ours is [Buehler et al., 2009] (described in Section 2.1.6) who used similar weak and noisy supervision from subtitles. However, their method does not exploit mouth motion, and relies on performing a computationally expensive brute force search over all temporal windows – here we avoid both the exhaustive search and also the necessity to represent hand shape and orientation as they did.

7.1 Pruning the Correspondence Search Space using Mouthing

A key contribution of this chapter is the discovery that mouth patterns are very helpful for aligning signs. This is because the Deaf in most countries commonly use their mouth to express the lip pattern of the English (or other written/spoken language) equivalent word that they are signing, also known as 'mouthing'. This mouth information is valuable in two distinct ways: (i) knowing that the Deaf mouth the word in the majority of signs, one can discard frames where the mouth

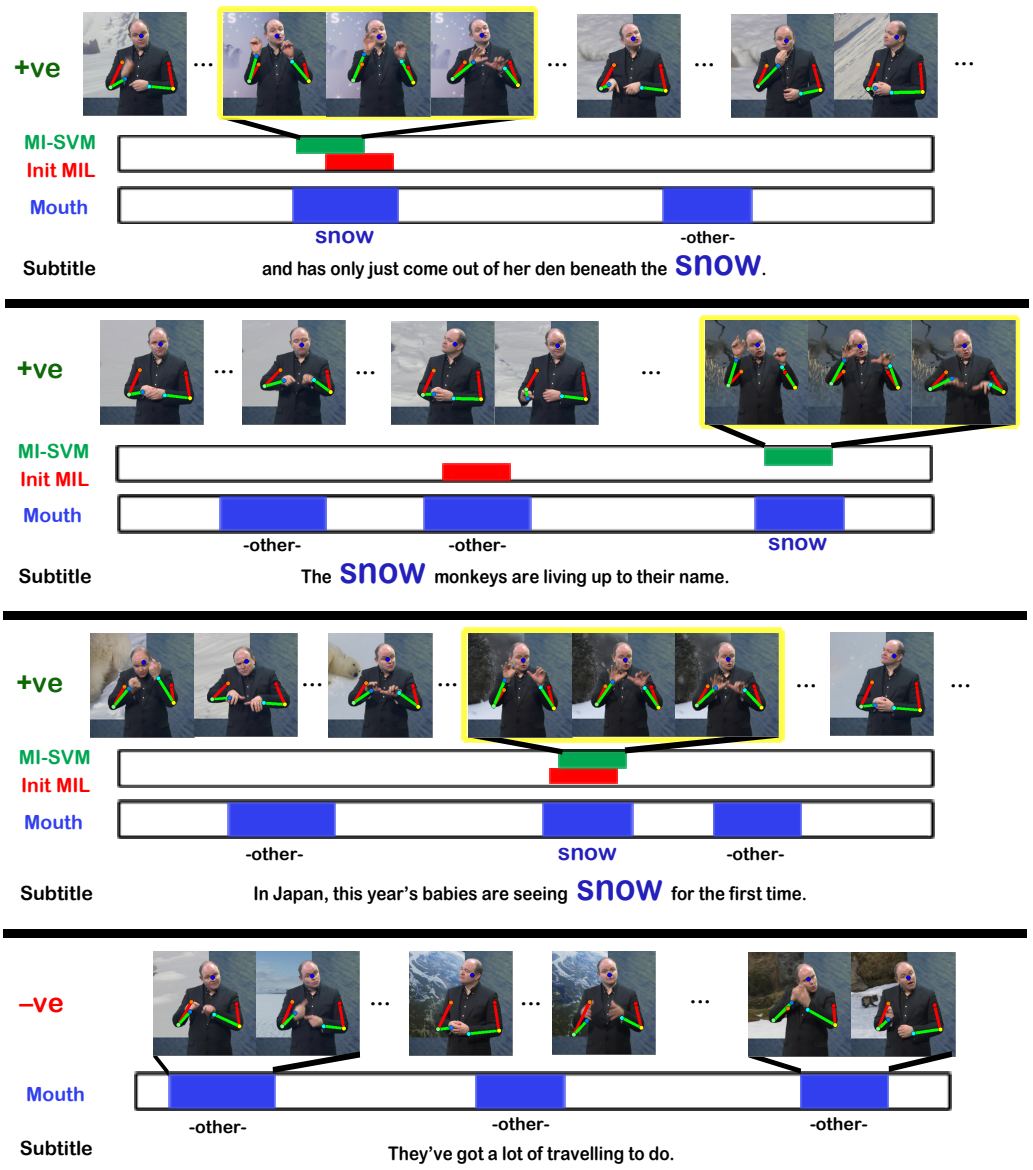


Figure 7.1: **Learning signs from co-occurrences of subtitle text, mouth and hand motion.** The top three rows are positive subtitle sequences which contain the text word and sign for 'snow'. The final row is an example of a negative subtitle sequence which does not contain 'snow'. Signs are learnt from this weakly aligned and noisy data. A fixed size temporal window is slid across the frames in which mouth motion occurs (blue). The rest of the sequence can be ignored, thus reducing the temporal search space. Candidate signs are proposed by a discriminative MIL search using temporal correlation. A subset of these candidates (red) are used to initialise a MI-SVM, resulting in the final correspondence matches (green). The red and green lines on the signer show the detected limbs and head.

Preprocess all videos
for each frame
– segment signer and locate signer's head and hands *(Part I)*
– classify mouth as open/closed and extract mouth SIFT descriptor *(Sect 7.3.2)*
➡ **output:** hand and head positions, mouth open/closed probability and mouth SIFT descriptor

For a particular word, e.g. 'snow'
– find positive & negative sequences using subtitles *(Sect 7.3.1)*
– obtain temporal windows by sliding a fixed-size window over frames with an open mouth in each sequence *(Sect 7.1)*
– extract feature vector for each temporal window *(Sect 7.3.3)*
– find sign candidates using discriminative MIL search *(Sect 7.2.1)*
– train a MI-SVM classifier with initial candidates *(Sect 7.2.2)*
➡ **output:** instances of the sign 'snow'

Figure 7.2: **Sign Extraction pipeline.** The preprocessing of videos (above) and the learning for a given word (below) are shown.

is not open, and thus considerably narrow down the search space when matching signs; and, to a lesser extent, (ii) the similarity of lip patterns across repetitions of the same sign can be used as an additional cue for matching different instances of the same sign.

In order to use the mouthing information to discard frames where the signer is not speaking, we train a per-frame classifier for predicting whether the mouths are 'speaking' vs 'not speaking' (details in Section 7.3.2). The output of this classifier (after temporal smoothing and thresholding – see Section 7.3.2) is used to prune the search space (*i.e.* filtering away frames in which the signer is predicted not to be mouthing, and therefore likely to not be signing). The search space that remains after this step is marked blue in Figure 7.1.

This pruning results in a substantial decrease in the number of temporal win-

dows that need to be considered for correspondences. We make this reduction more concrete with an example: in the sequences of Figure 7.1, the search space is $l = 400$ frames, and sign correspondences are searched for with a fixed-size temporal window of $w = 13$ frames. Without using mouthing information, this would yield $n = l - w + 1 = 388$ candidate temporal windows per positive sequence. However, by cutting this search space down to three $l = 25$ frame windows using mouthing information, the number of candidate windows drops 90% to 39. This order-of-magnitude reduction in search space not only improves the 'signal-to-noise' ratio in the correspondence search, but also considerably speeds up the search.

7.2 Automatic Sign Extraction with Multiple Instance Learning

Given a target word occurring in the subtitles, and the pruning method that uses mouthing, we show here how to extract examples of the corresponding sign.

The key idea is to search for signs that are common across the positive sequences (the sequences where the target word occurs in the subtitles), but uncommon in negative sequences (where the target word doesn't occur in the subtitles). Since the positive labels are on a subtitle sequence level rather than on a window level, the task can naturally be formulated as a Multiple Instance Learning (MIL) [Dietterich et al., 1997, Maron and Lozano-Pérez, 1998] problem, as shown in Figure 7.1. MIL is a variation of supervised learning for problems that have

incomplete knowledge about the training set's labels – unlike supervised learning in which each training instance has a label, in MIL the labels are on a *bag* level, where each bag consists of *instances*. If a bag is *positive*, then *at least one instance* in the bag is positive. If it is *negative*, then *no instance* in the bag is positive.

In our learning scenario, the MIL 'bags' are the sequences, and the 'instances' are features computed from fixed sized temporal windows within the temporal intervals in which the signer is mouthing. The positive bags are from positive sequences, and negative bags from negative sequences. Details of the features used as an input to this learning (that describe the fixed sized temporal windows) are given in Section 7.3.3.

To give a concrete example demonstrating the difficulty of solving this problem, imagine a dataset in which the word 'snow' occurs in 30 subtitles. This yields 30 positive sequences, each around 400 frames long. Out of these 400 frames, empirically on average six subsets (each 30 frames long, examples shown in blue in Figure 7.1) contain mouthing. With a temporal window size of 13 (which was used in this work), this yields 108 temporal windows for each positive sequence, or in total 3,240 temporal windows for all positive sequences. However, an additional challenge is posed by the fact that 'snow' is only signed in 10 out of the 30 sequences (*i.e.*, the annotation is weak). In this case, our task is to find the 10 out of 3,240 temporal windows that contain the target sign. With a 'signal-to-noise' ratio of less than 0.4%, this is a very challenging problem even when using mouthing to cut down the search space (in this example, mouthing reduced the

number of temporal windows from 11,640 to 3,240).

The MIL method proceeds in two stages (each described below): (i) finding good candidates for the target sign temporal windows using *temporal correlation scores*, and (ii) refining this 'candidate list' using MI-SVM. Example outputs from these two steps are given in Figure 7.1.

7.2.1 Multiple instance learning with a discriminative temporal correlation score-based search

The method for finding candidate temporal windows relies on computing *temporal correlation scores* between the fixed-size temporal windows.

The input is a set of feature vectors $\{x_i\}$, each representing the temporal motion of the hands and mouth over a fixed sized temporal window. Each vector x_i is composed of blocks covering aspects of the signing of a given temporal window, such as lip motion and distance between the hands (*e.g.* with window size 13 and feature dimensionality D, each feature vector is $13 \times D$-dimensional). The vector is normalised such that the dot product $x_i \cdot x_j$ between two such vectors, x_i and x_j, gives the *temporal correlation score* of the 'signals' (lip motion, hand motion) over the two temporal windows (Section 7.3.3 gives the details on the feature vector). The temporal correlation measures how similar the hand and lip trajectory is between the two windows, with a value of 1 indicating perfect correlation, and -1 indicating anti-correlation.

The task is to determine for each feature vector x_i how likely it is to be the

target sign. This is accomplished by using each x_i to classify the positive and negative sequences. The idea is that if x_i is actually the sign then its correlation (i.e. $x_i \cdot x_j$) with some vector x_j in a positive sequence will be higher than with any vector x_k in a negative sequence. To this end, for each x_i all sequences (both positive and negative) are ranked using the 'classifier' score $x_i \cdot x_j$, and the performance of the classifier is assessed using the area under its ROC curve (AUC). For the purpose of annotating the vectors when computing the ROC curve, any vector in a positive sequence is deemed positive (+1), and any vector in a negative sequence deemed negative (−1). A good candidate temporal window x_i will rank the positive sequences first, and thus have a higher AUC, than a poor candidate. Note that the annotation of the 'positives' here is noisy, since only a fraction of the windows in positive sequences actually contain the sign.

In summary, we measure the 'quality' of each temporal window in the positive sequences using the AUC of its ROC curve – a form of one shot-learning. The windows are then ranked according to their AUC scores, and this 'ranked candidate list' is used below for initialising and training the MI-SVM.

Discussion. Before temporal correlation scores are computed, the features x are whitened to $\hat{x} = \Sigma^{-1/2}(x - \mu)$, where Σ is the cross-correlation matrix, x is an L_2-normalised input feature vector and μ is the mean of the input feature vector (over space-time features). Whitening effectively 'equalises' the features, thus making feature variations more comparable, leading to better learning.

This MIL initialisation method is not limited to applications in sign language

– the same idea can be used to initialise MIL in other weakly supervised tasks.

7.2.2 Temporal correlation-based MI-SVM

In MI-SVM, given the positive and negative bags as input, a classifier w is learnt to select the positive instances x from the positive bags by an algorithm that alternates between: (i) selecting the positive instance in each bag as those with maximum score $w \cdot x$, and (ii) standard SVM training using the selected positives and all negative instances. More formally, given a set of input temporal windows $x_1, \ldots, x_n$ grouped into bags $\mathbf{B}_1, \ldots, \mathbf{B}_m$ according to which positive/negative video subsequence they belong to, where each bag $\mathbf{B}_I$ is associated with a label $Y_I \in \{-1, 1\}$, we optimise

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_I \xi_I \tag{7.1}$$

$$\text{s.t.} \quad \forall I \; : \; Y_I \max_{i \in I}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_I, \quad \xi_I \geq 0 \tag{7.2}$$

where i are indices for instances and I are indices for bags.

However, a good initialisation is necessary for the algorithm to succeed. The ranked candidate list from Section 7.2.1 is used to first initialise and then train the MI-SVM. In detail, the shortlist uses the top 20% highest-ranked candidate windows as positives. The shortlist candidates are allocated to their positive sequences and a weight vector w is learnt using the MI-SVM algorithm, where the

top ranked candidates in each sequence are used for initialisation. The candidates with maximum SVM score in each sequence form the final result. A separate MI-SVM is trained and evaluated for each target word.

Training in this manner means that the weight vector is learnt from instances with far fewer false positives (higher 'signal-to-noise'). It is demonstrated in Section 7.4.2 that this substantially improves its performance (compared to MI-SVM learning from all windows directly). Figure 7.1 shows the selection 'in action' on three positive subtitle sequences.

Discussion. The weight vector w is learnt discriminatively and thus can learn to suppress part of the feature vector. For example, if the distance between the hands carried no discriminative information for a particular set of positive sequences, then this block of the feature vector need not be selected. The vector w is a stronger classifier than the exemplar LDA classifier above, since w uses multiple positive samples for training, rather than being constructed from only a single sample.

7.3 Implementation Details

This section describes the implementation details of how training data (positive/negative sequences, and a feature vector based on upper-body joint locations and mouth features) are automatically generated from subtitles and video material.

First, Section 7.3.1 provides details of how the video is divided into 'positive' (likely to contain target sign) and 'negative' (unlikely to contain the target sign) sets; then Section 7.3.2 discusses mouth features; and finally, Section 7.3.3 gives a summary of the full feature vector used to find correspondences between signs.

7.3.1 Text processing: extracting positive and negative sequences

Each subtitle text consists of a short text, and a start and end frame indicating when the subtitle is displayed. The subtitles are stemmed (common inflections *e.g.* "s", "ed", "ing" are removed) and stop words are filtered away.

Positive sequences. A positive sequence is extracted for each occurrence of the target word in the subtitles. Since the alignment between subtitles and signs is very imprecise due to latency of the signer (who is translating from the soundtrack) and differences in language grammar, some 'slack' is padded to the sequence window. Given a subtitle where the target word appears, the frame range of the positive sequence is defined as from the start of the *previous* subtitle to the end of the *next* subtitle. This results in sequences of about 400 frames in length. In contrast, signs are generally 7–13 frames long.

Negative sequences. Negative sequences are extracted by searching for subtitles where the target word *does not* appear. This yields on average about 100,000 negative temporal windows per video.

7.3.2 Mouthing classifier

To train the mouthing classifier, we first detect facial landmarks using the method of [Everingham et al., 2006]. A similarity transform is then applied to the mouth feature points to yield a scale, rotation and translation normalised *mouth patch*. Given these mouth patches, a binary Chi-squared kernel SVM is trained to classify each such patch as mouthing / non-mouthing using Local Binary Pattern (LBP) [Ojala et al., 2002] features (with cell size 8 extracted from the mouth patch of size 32×52 pixels). The dimensionality of the feature vector is 1,392 per frame.

At test time, the SVM output scores are thresholded and temporally smoothed with a Gaussian to yield windows in which mouth motion is detected (blue areas in Figure 7.1). Details about training and performance of the mouthing classifier are given in Section 7.4.2.

7.3.3 Feature vector for temporal windows

For each frame, we obtain the position of the signer's head and hands (see Part I), and a descriptor for the mouth shape (a 128-dimensional SIFT descriptor [Lowe, 1999] computed using the facial landmarks from Section 7.3.2).

The feature vector for each temporal window consists of features computed from the position of the signer's hands, and the SIFT descriptor of the mouth.

A number of feature blocks are computed based on the trajectory of the hands: (i) the relative (x, y) coordinates of each hand compared to the position of the

head, (ii) the differences $(x_{\text{right}} - x_{\text{left}}, y_{\text{right}} - y_{\text{left}})$ in coordinates between the two hands, and (iii) a vector describing the direction and magnitude of motion between the first and last frame in the temporal window: $(x_{\text{last}} - x_{\text{first}}, y_{\text{last}} - y_{\text{first}})$.

In addition, the feature vector contains a block for the SIFT descriptor of the mouth patch for each frame in the temporal window.

The feature dimension per frame is 134, out of which 128 is a SIFT describing the mouth, and the remainder describes the joints. For a temporal window size of 13, the total feature vector dimensionality is 1,746 (134×13, plus two vectors that give the direction and magnitude of motion between the first and last frame of each hand).

Each feature block is l_2-normalised, so that $x_i \cdot x_j$ becomes the temporal correlation between two temporal windows with feature vectors x_i, x_j. The entire feature vector is a concatenation of all the l_2-normalised blocks, and this is then l_2-normalised.

7.4 Experiments

We first describe the experimental setup (Section 7.4.1); then the performance of the sign extractor and mouthing classifier are reported (Section 7.4.2); then results are compared to the state of the art (Section 7.4.3); and finally the computation time is discussed (Section 7.4.4).

The dataset used in the evaluation of this chapter is the sign extraction dataset (Section 3.2.1), which consists of 35 videos, in total 30 hours of data, with 41

manually ground truthed signs that spanning multiple videos.

7.4.1 Experimental setup

Evaluation measures. Given an English word, the goal is to identify many examples of the corresponding sign. We use two evaluation measures: *sign-level* (coarser) and *instance-level* measures. In the sign-level measure used by [Buehler et al., 2009], the output is deemed a success if at least 50% of retrieved candidates (maximum one per subtitle sequence) show the true sign (defined as a temporal overlap of at least 50% with ground truth). In the instance-level measure, we report precision and recall computed per word and then averaged across words. Precision measures the percentage of retrieved windows that contain the correct sign; recall measures the percentage of sign instances that are retrieved.

Train/test sets for mouthing classifier. The mouthing SVM classifier is trained on the mouth LBPs (described in Section 7.3.2) of five unseen signers that are not in the sign extraction dataset. Mouths were manually annotated as either open or closed in 800 frames for each signer (4,000 frames in total). Testing for the mouthing classifier is conducted on three randomly chosen signers in the sign extraction dataset, each with 200 manually annotated frames (600 frames in total).

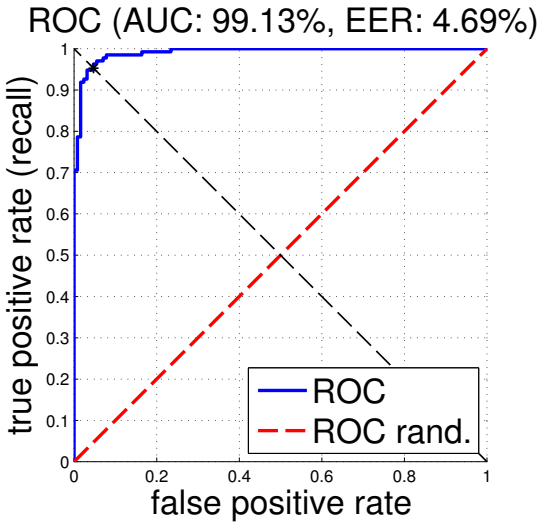


Figure 7.3: **ROC curve of the person-independent mouthing classifier.**

7.4.2 Experiments

In this subsection the performance of the mouthing classifier is evaluated, the sign extraction results are reported, and the advantages of using mouthing for sign extraction, and initialisation for MI-SVM, are described.

Mouthing classifier. Figure 7.3 shows the ROC curve for the mouthing classifier when trained and tested on different signers as described in Section 7.4.1. As the ROC curve demonstrates, the classifier gives an impressively reliable measure of whether the mouth is open or closed. On average, 71.2% of the search space is discarded using this method, demonstrating its value.

Sign extraction results. The instance-level average precision is 57.1% and recall is 78.0%. With the sign-level evaluation measure, the performance is 92.7%. A more detailed set of results is given in Table 7.1. We achieve good results for a

wide variety of signs: (i) signs where the hand motion is important (*e.g.* 'snow'), (ii) signs where the hand shape is particularly important (*e.g.* 'jewish' where the hand indicates an imaginary beard), and even (iii) signs which are performed in front of the face (*e.g.* 'pork'), which makes detecting mouth motion difficult. MI-SVM suppresses the mouth part of the feature vector by assigning it a lower w weight.

Advantages of using mouth-signing co-occurrences and MI-SVM initialisation. The importance of the components and stages of the algorithm is evaluated next. If the mouthing classifier is not used for cutting down the search space, some signs can be detected, but the overall results are much poorer. This is reflected in the instance-level evaluation measure: average precision over 41 words drops to 17.8% and recall to 39.3%. The overwhelming majority of the ground truth signs are mouthed, so using it to cut down the search space results in only a minor loss of positive instances. If the initialisation step is omitted, and MI-SVM is instead initialised with all windows from the positive sequences (after search space reduction), results drop to 5.7% precision and 2.5% recall.

We have also qualitatively evaluated our method over the 1,000 words. For more than half of the words, our method returns the correct sign one or more times in the top 10 final temporal windows selected by MI-SVM.

Sign	#(pos)	#(GT)	TP_m	FP_m	$TP_{\not m}$	$FP_{\not m}$
animal	12	9	6	4	0	11
antique	15	8	7	6	3	11
asian	11	9	5	2	1	10
bank	21	10	10	7	6	13
beacon	28	21	8	6	4	24
bear	14	10	2	2	1	13
beautiful	12	5	3	3	0	11
beef	15	8	7	7	3	10
bike	15	6	4	4	1	12
blood	15	4	3	2	2	8
buy	13	3	2	3	0	10
Chinese	33	11	11	11	0	28
chocolate	20	6	6	3	3	17
epigenome	20	13	7	9	7	13
fake	14	11	11	2	0	13
feel	12	8	6	3	3	8
gram	25	12	7	5	1	15
heart	14	5	4	4	0	10
heat	15	5	10	4	2	12
industry	19	9	9	4	4	15
jelly	11	1	4	4	4	4
jewish	27	14	14	5	8	11
kill	17	9	6	6	4	13
market	16	12	8	5	2	14
milk	10	6	4	4	1	7
pay	37	25	12	11	12	22
reindeer	15	9	5	5	2	13
rugby	11	7	5	4	2	11
school	13	5	3	2	1	10
science	18	10	6	5	1	17
sell	15	5	4	3	4	9
simple	12	10	3	1	0	11
snow	29	11	8	5	1	24
song	19	10	10	6	4	15
sound	26	5	5	2	2	21
target	23	4	4	4	2	21
vision	17	10	7	5	6	7
war	11	6	3	5	2	8
winter	23	12	7	7	2	21
work	22	14	3	3	2	17
year	32	8	7	6	4	26

Table 7.1: **Ground truth and performance for 41 words.** The first three columns show statistics of the training data, and the remaining four columns show the performance of the method on those words with (m) and without ($\not m$) using the mouth classifier. #(pos) is the number of positive sequences for the word in the same row. #(GT) is the number of times the sign actually occurs in the positive sequences. The number of correctly detected signs in the positive sequences is given in TP, while FP gives the number of incorrect detections.

7.4.3 Comparison to previous publications

Direct comparison to previous works is not possible since we do not have access to the same TV programmes with the same signs performed by the same signer. Moreover, previous work used standard-definition TV broadcasts, where the resolution was not good enough to detect facial feature points reliably. However, we show that our results are competitive when performing similar experiments to those in [Buehler et al., 2009] (see Section 2.1.6) with a similar-sized vocabulary.

Using the sign-level evaluation measure, our 92.7% success rate far exceeds Buehler *et al.*'s rate of 78% on the same number of signs. However, we must also point out that this measure is only concerned with recall, and not precision. In fact, although our method has a good recall, it has lower precision (*i.e.* higher false positive rate) on the 41 word test set. This is to be expected given that we are using less discriminative features than Buehler *et al.*, who also use hand shape and hand orientation. However, our method is much simpler both in terms of features and learning framework, and is extremely fast ($\approx$ 2min/word).

[Cooper and Bowden, 2009] detect the signs for 53.7% of 23 words in a 30 min TV broadcast. The results are not directly comparable as different performance measures are used, but we detect 92.7% of nearly twice as many words, at a fraction of the computational cost.

Figure 7.4: **Example sequences for the signs "snow" (top) and "vision" (bottom)** performed by two different signers and learnt automatically.

7.4.4 Computation time

The following computation times are on a single core of a 2.4GHz Intel Quad Core I7 CPU. Segmentations, joints and mouthing classifier scores for one frame are computed in 0.3s (3fps). The runtime for the MIL initialisation step is on average 20s per subtitle word, and MI-SVM converges on average in 1min 30s, totalling 1min 50s per word. Initially there are on average 4,000 temporal windows, of which the MIL initialisation step returns around 800 as a shortlist.

7.5 Conclusion

We have shown a framework for automatically learning a large number of signs from sign language-interpreted TV broadcasts. Our method exploits co-occurrences of mouth and hand motion to substantially improve the 'signal-to-noise' ratio in the correspondence search. Moreover, we have proposed a principled method for

initialising the correspondence search, which significantly improves performance. We achieve superior results to those in previous work, with much simpler features and a much lighter learning framework.

These ideas are not constrained to sign language. For the idea of using co-occurrences to cut down the search space, co-occurrences of lip motion and speech can be used to aid speech recognition [Bregler and Konig, 1994, Dupont and Luettin, 2000, Petajan et al., 1988], and the vicinity of objects can be used to aid human action recognition [Prest et al., 2012] (*e.g.* a phone close to human and a pose with a hand close to the head can be used to detect that the action is 'phoning'). Both the idea of exploiting co-occurrences and obtaining MIL temporal correlation candidates by discriminative learning from a single exemplar window could be applied to a variety of fields where weak supervision is available, such as learning actions [Laptev et al., 2008], gestures and names of TV characters [Everingham et al., 2009].

Chapter 8

Learning Gestures from Weak and Strong Supervision

We saw in Chapter 7 that it is possible to learn sign language using weak supervision from subtitles. However, as discussed there, this subtitle supervision is very weak and noisy, which makes learning from it difficult.

In this chapter, we show that we can further improve upon this learning of signs by combining the weak supervision with *strong supervision* from dictionaries (such as sign language dictionaries). We further show that this method extends beyond sign language, and present results on a dataset of Italian gestures.

The main idea of the chapter is showcased in Figure 8.1. Here, we have two strongly supervised videos (dictionaries) with an example of each gesture, and a large dataset of TV broadcasts with weak supervision from subtitles, which contain some instances of the same gestures. In the dictionaries there is strong

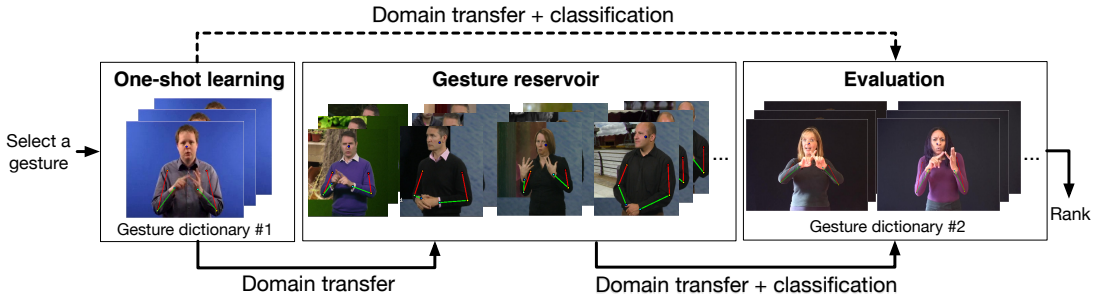


Figure 8.1: **Learning gestures from strong and weak supervision.** A single strongly supervised training example from a dictionary is used to train a *one-shot learner*. This one-shot learner is used to obtain additional training data from a large weakly supervised gesture repository of another domain. These new samples are used to 'boost' the one-shot learner with additional discriminative power. Evaluations are carried out under further domain adaptation on another strongly supervised dictionary dataset. Dashed line shows the **baseline** and the solid lines show the **proposed method**.

supervision (but only a single example), whereas in the weakly supervised dataset there are a lot of examples, which exhibit variations in people, expression, speed – but there is only weak supervision, as the temporal interval is not tight.

We show that we can learn a much stronger classifier by using both the dictionary and the weakly supervised videos compared to using either one alone.

Our method (described in Section 8.1) works in four main steps: (i) learn a weak classifier from a dictionary (one-shot learning); (ii) use the weak one-shot classifier to select further examples from a weakly supervised dataset; (iii) filter away false matches by comparing the similarity of the hand shape of the examples and dictionary entry (Section 8.3); and (iv) train a stronger classifier from the remaining examples, which has more robustness to variations in gesturing speed, person specifics *etc*. This is a form of semi-supervised learning, but since the dictionary and weakly supervised videos are very different (different resolutions,

people with different body shape, with gestures performed at significantly different speed), the affinity function requires domain adaptation in going between them (Section 8.2).

We evaluate the method on two large gesture datasets: one for sign language (the extended sign extraction dataset in Section 3.2.2), and the other for Italian hand gestures (ChaLearn in Section 3.2.4). In both cases performance exceeds the previous best results, including the best skeleton-classification-only entry in the 2013 ChaLearn challenge.

Related work. Most gesture recognition methods rely on strongly supervised learning, which requires ground truthing large quantities of training data. To avoid the expense of data labelling, recent works have attempted to learn gestures at the other extreme – from single training examples using one-shot learning [Guyon et al., 2013]. However, given the vast variability in how gestures are performed (and the variation in people and camera viewpoints), learning generalisable models with so little supervision is challenging. An alternative is using weak supervision as in Chapter 7; however, the supervision is quite weak and noisy, making it difficult to learn from it alone. Here we show that combining both of these approaches (weak and one-shot supervision) is advantageous.

8.1 Learning from Strongly Supervised Dictionaries and Weakly Supervised Videos

In this section we describe the method for learning gestures (and sign language) from both strongly supervised dictionaries and weakly supervised videos.

This method takes as an input hand trajectories from both datasets which have been temporally and spatially aligned. The method of obtaining hand trajectories is described in Chapter 4, and the alignment method is described in detail in Section 8.2 below.

In more detail, the method proceeds in four steps: (i) train a discriminative gesture detector from the first dictionary, separately for each gesture; (ii) that detector is then used to discover new samples of the same gesture in the weakly supervised gesture reservoir – the search for the sample is restricted to a temporal interval provided by the weak supervision; (iii) these new samples are used to train what is effectively a stronger version of the original one-shot gesture classifier; and (iv) this strong classifier is evaluated on a second one-shot dictionary.

Given the two strongly supervised video dictionaries (one for training, the other for evaluation) and a weakly labelled gesture reservoir, let $\boldsymbol{\delta}^1, \boldsymbol{\delta}^2$ and $\boldsymbol{\nu}$ denote their respective features (here the hand trajectories). For example, $\boldsymbol{\delta}^1 = \{\boldsymbol{\delta}^1_1, \ldots, \boldsymbol{\delta}^1_q, \ldots\}$ where $\boldsymbol{\delta}^1_q$ is a variable-length vector (depending on the length of the gesture video) of hand positions over all frames in the q^{th} gesture video of dictionary 1.

Imagining that we are learning a gesture for 'snow' in BSL (shown in Figure 8.2), the four learning steps proceed as follows:

1. Train one-shot learner on strongly supervised dictionary. We first train a discriminative one-shot gesture detector for 'snow' on the features of the first dictionary ($\boldsymbol{\delta}^1$). To do this, we use a time-and-space-aligned gesture kernel ψ (which we discuss in detail in Section 8.2) in a dual SVM to learn weights α from

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \psi(\mathbf{x}_j, \mathbf{x}_k) \quad \forall i \; 0 \leq \alpha_i \leq C \quad \sum_i \alpha_i y_i = 0 \qquad (8.1)$$

where we set the learning feature to $\mathbf{x} = \boldsymbol{\delta}^1$ (hand trajectories of the videos in dataset δ^1), y_i are binary video labels (1 for the 'snow' dictionary video, -1 for others), and $\psi(,)$ is the kernel. This is the one-shot learning part of our method – effectively an exemplar SVM [Malisiewicz et al., 2011].

2. Use one-shot learner to extract training examples from weakly supervised videos. We use the one-shot learner to discover new samples of 'snow' in the weakly supervised gesture reservoir (restricted to the temporal intervals provided by the weak supervision). A very large number of samples in the reservoir are scored to find gestures that are most similar to 'snow' (and dissimilar to the other gestures) in the first dictionary. This yields a vector of scores s

$$s(\boldsymbol{\nu}) = \sum_i \alpha_i y_i \psi(\mathbf{x}_i, \boldsymbol{\nu}) + b \tag{8.2}$$

where $\boldsymbol{\nu}$ are the features for reservoir subsequences with a weakly supervised label 'snow'. Here, $s(\boldsymbol{\nu})$ is a vector of scores of length $|\boldsymbol{\nu}|$ (the number of samples in the weakly supervised sequences of the gesture reservoir). The top scored samples represent gestures in the reservoir that are most similar to 'snow' in the the first dictionary, but with high variability in space, time and appearance (thanks to the time and space adaptations from Section 8.2).

3. Train a stronger classifier on a subset of the extracted training examples. The top samples of $s(\boldsymbol{\nu})$ (by score), along with a set of negative samples from the gesture reservoir, are used to train a stronger version of the original one-shot gesture classifier for 'snow' (the training details for this step are given in Section 8.4). We do this by retraining (8.1) with this new training set $\mathbf{x} = \boldsymbol{\nu}_{\text{retrain}}$ (of cardinality around 2,000 samples). Due to only selecting the top samples of the gesture reservoir for training, we develop resilience to noisy supervision.

4. Evaluate the stronger classifier on another strongly supervised dictionary. Finally, this stronger model is evaluated on the second dictionary by ranking all gesture videos using the score $s(\boldsymbol{\delta}^2)$ of the stronger classifier. This provides a measure of the strength of the classifier without requiring any expensive manual annotation.

Figure 8.2: **Frames showing variation in gesture speed across two domains** (top and bottom). The example gesture shown here is 'snow' in BSL, which mimics snow falling down. Although the frame rate is the same, the speed at which the gestures are produced are considerably different.

8.2 Domain Adaptation of Hand Trajectories

A major challenge in gesture recognition is that not only are the gestures performed by different people with different body shapes, but the same gestures can be performed at very different speeds across domains and people. For two datasets in this work, this is showcased in Figure 8.2).

We tackle this problem by measuring distance under domain adaptations in both space and time. We next discuss the domain adaptations used to define this kernel ψ that is used in the dual-form SVM in Section 8.1.

8.2.1 Time alignment

Dynamic Time Warping (DTW) [Sakoe, 1978, Sakoe and Chiba, 1970] is a popular method for obtaining the time alignment between two time series and measuring their similarity. However, there have been problems incorporating it into a discriminative framework (*e.g.* into kernels [Baisero et al., 2013, Gaidon et al.,

2011, Shimodaira et al., 2001, Zhou and De la Torre, 2012]) due to the DTW 'distance' not satisfying the triangle inequality. As a result, it cannot be used to define a positive definite kernel. Furthermore, it is unlikely to be robust as a similarity measure as it only uses the cost of the minimum alignment.

In this work we use a recently proposed positive definite kernel, the Global Alignment (GA) kernel [Cuturi, 2011, Cuturi et al., 2007]. In addition to being positive definite, it has the interesting property of considering *all* possible alignment distances instead of only the minimum (as in DTW). The kernel computes a soft-minimum of all alignment distances, generating a more robust result that reflects the costs of all paths:

$$k_{\mathrm{GA}}(\mathbf{x}, \mathbf{y}) = \sum_{\pi \in \mathcal{A}(n,m)} e^{-D_{\mathbf{x},\mathbf{y}}(\pi)} \tag{8.3}$$

where $D_{\mathbf{x},\mathbf{y}}(\pi) = \sum_{i=1}^{|\pi|} \|x_{\pi(i)} - y_{\pi(i)}\|$ denotes the Euclidean distance between two time series $\mathbf{x}, \mathbf{y}$ under alignment π, and $\mathcal{A}(n, m)$ denotes all possible alignments between two time series of length n and m. In our case $\mathbf{x}, \mathbf{y}$ are two time series of spatially aligned human joint positions, *i.e.* the joint 'trajectories' of two gestures that are being compared. By incorporating all costs into the kernel we improve classification results compared to only considering the minimal cost.

8.2.2 Spatial alignment

Since the hands play an important role in gestures, knowing where the wrists are is valuable to any gesture recognition method. However, an issue with human

joint positions is that they are not directly comparable across domains due to differences in both position, scale and human body shape. We use two simple yet effective affine transformations, one global and another local in time, that allow for translation and anisotropic scaling. This encodes a typical setup in gesture datasets, where, for a particular gesture, the persons stay at roughly the same distance from the camera (global transform), but may move slightly left or right (local transform). The global transformation learns the anisotropic scaling and translation, and the local transformation estimates an x translation, mapping into a canonical frame in which poses from different domains can be directly compared.

The global transform is computed from the median positions of the shoulders and elbows (selected since they are comparable across videos) over the whole video. The x translation is estimated locally from the median head and shoulder positions over a small temporal window (50 frames). Figure 8.3 shows a visualisation of the transformation.

Even after spatial transformations, the absolute position for the gesture (relative to the torso) generally differs slightly. We solve that by adding some 'slack' to allow for slight absolute position differences. We do this by minimising the l_2 distance between wrist trajectories (of the two videos that are compared) over a small local square patch of width $u =$ (dist. between shoulders)/10. Figure 8.4(c–d) shows an example of the original and corrected positions.

The composition of the global and local transformations define the spatial transformation $\boldsymbol{\phi}$, *i.e.* $\boldsymbol{\phi}(x)$ is the mapping from the trajectory in the video to the

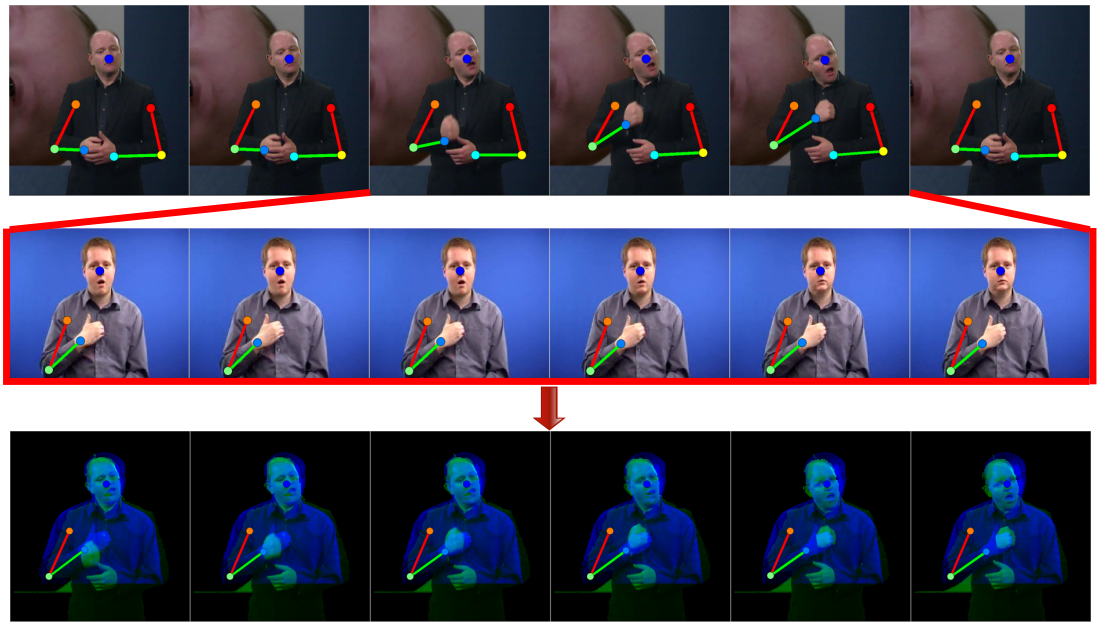


Figure 8.3: **Domain adaptation in space and time. Top:** video sequence from the gesture reservoir; **Middle:** automatic time alignments to another sequence from a one-shot learning domain; **Bottom:** domain-adapted (space and time aligned) sequences, with the middle sequence overlaid on the top one. For ease of visualisation the example only uses a dominant hand, so only the dominant hand is matched (this is determined from the one-shot learning dictionary). In most cases, the transformation involves both hands.

spatial canonical frame.

8.2.3 Final transformation kernel ψ

The final kernel is a composition of the the time alignment k_{GA} and spatial transformations $\boldsymbol{\phi}$. This yields the final kernel:

$$\psi(\mathbf{x}, \mathbf{y}) = k_{\mathrm{GA}}(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y})). \tag{8.4}$$

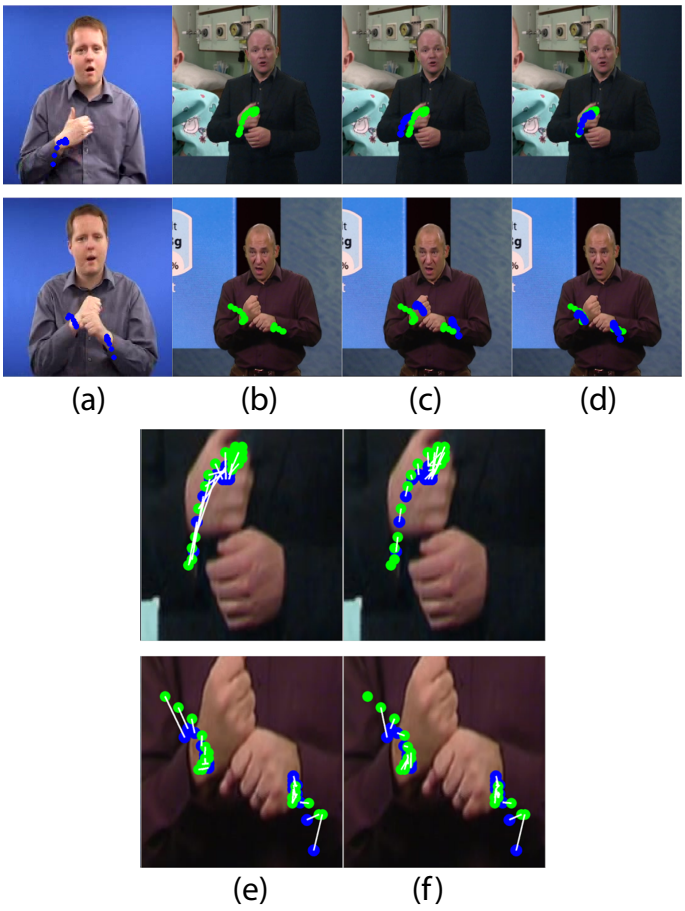


Figure 8.4: **Human pose transformation across domains.** (upper row) BSL for 'heart' with overlaid wrist trajectory; (bottom row) BSL for 'gram'. (a) and (b): wrist trajectories for domains 1 and 2, (c) trajectory of domain 1 mapped onto domain 2 with a spatial transformation, (d) transformation with minimisation of the local position 'slack', (e) zoomed-in similarity without temporal alignment (white lines represent wrist point correspondences across the two domains), and (f) similarity with temporal alignment. As shown, the distance (proportional to the sum of the lengths of the white lines) is lower under alignment.

8.3 Using Hand Shape as a Filter

As Figure 3.6 demonstrates, hand shape carries much of the discriminative information in gestures, particularly in complex gesture languages such as sign language, and needs to be included in order to successfully learn gestures. We

use a hand shape descriptor to discard false positives of reservoir samples where the wrist trajectories of the one-shot learning domain and the gesture reservoir match, but the hand shape is different (the similarity score is below a threshold, where the threshold is determined on a validation set).

Comparing hand shapes across domains is not straightforward since the domains may be of different resolution, contain different persons, lighting *etc*. Moreover, our pose estimator only provides wrist positions (not hand centres). We next describe a domain-independent, somewhat lighting-invariant hand shape descriptor that addresses these challenges.

We follow the method of [Buehler et al., 2009] (described in Section 2.1.6) where hands are first segmented, and then assigned to a cluster index. The clusters are used both to provide a distance between hand shapes and also to aid in the segmentation. To compare two hands in different domains, we assign them to their nearest hand cluster exemplars and measure their similarity as the distance between the HOGs of their cluster exemplars (shown in Figure 8.5).

In detail, GraphCut [Boykov and Jolly, 2001, Rother et al., 2004] is used for an initial segmentation (with skin colour posteriors obtained from a face detector), and the segmented hands are represented using HOG features (of dimensionality $15 \times 15 \times 31$). The segmentation is performed within a box defined by an estimate of hand centre position (based on the elbow-wrist vector). Hand exemplars are then formed by clustering HOG vectors for examples that are *far away* from the face using k-means ($K = 1000$). These are effectively 'clean' hand clusters, without face regions in the foreground segmentation. For an input image,

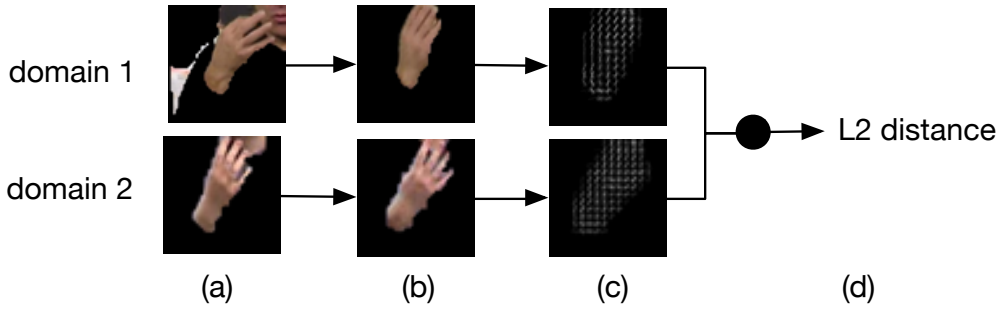


Figure 8.5: **Hand shape descriptor.** (a) Badly segmented hands (due to overlap with skin) in two domains, (b) hands assigned to their hand cluster exemplars, (c) HOG of size-normalised exemplars, and (d) hands are compared across domains in terms of l_2 distance between the HOGs of the hand exemplars.

HOG vectors are matched to their nearest hand cluster, resulting in a 'cleaned' segmentation of the hand.

8.4 Implementation Details

In this section we provide the remaining implementation details.

Learning framework. For each word, the positive training samples are obtained from the top ranked positive samples of each reservoir video. If there are w_c occurrences of the word in the subtitles of a reservoir video, then the top $5w_c$ positive samples are used – note, no non-maximum suppression is used when sliding the classifier window so there are multiple responses for each occurrence. The number of positives is capped at 1,000, and 1,000 randomly sampled reservoir gestures are used as negatives. We use the dual formulation of SVMs since our space and time alignment method provides the alignments as kernels, not in feature space (so primal optimisation is not suitable).

Hand shape. We precompute a $K \times K$ hand distance matrix offline for any one-shot learning domain and gesture reservoir video pair. At runtime, the comparison of two gestures is reduced to looking up the distance in the matrix for each pair of time-aligned frames, and summing up the distances.

8.5 Experiments

Four datasets are employed in this work: the extended sign language extraction dataset containing TV broadcasts (Section 3.2.2); two sign language dictionaries (Section 3.2.3); and a dataset of Italian hand gestures (Section 3.2.4).

8.5.1 Detecting gestures in TV broadcasts by training a one-shot learner on a dictionary

Here we evaluate the first main component of our method, *i.e.* how well can we spot gestures in the weakly supervised videos given a one-shot learning example from a dictionary? We compare this component to Chapter 7, where we used Multiple Instance Learning to extracting gestures purely from weakly supervised TV broadcasts. We show that we outperform Chapter 7 with this conceptually simpler approach.

Manual ground truth. The test dataset for this experiment (a six hour subset of BSL-TV) is annotated for six gestures (bear, gram, heart, reindeer, snow and winter), with on average 18 occurrences for each gesture, and frame-level manual

ground truth as described in Section 3.2.1 (where we spent a week to label 41 words frame-by-frame). A benefit of the domain adaptation method is that it renders this expensive manual labelling less important, since the training and test sets no longer need to be of the same domain, which enables the use of supervised datasets from other domains (*e.g.* other dictionaries) for testing (as we show in the next experiment).

Task. The task for each of the six gestures is, given one of the 15s temporal windows of continuous gestures, to find which windows contain the target gesture, and provide a ranked list of best estimates. Only about 0.5s out of 15s actually contain an instance of the gesture; the remainder contain other gestures. A gesture is deemed 'correct' if the intersection over union score is over 0.5.

Results. Precision-recall curves for the gestures are given in Figure 8.6. As shown, thanks to our domain-adapted one-shot learning method, we vastly outperform the purely weakly supervised method of Chapter 7. This shows clearly the value of using strong supervision.

8.5.2 Learning gestures from strong and weak supervision

In this key experiment we evaluate our method trained on the 155 hour BSL-TV weakly supervised dataset.

The method is evaluated on a second strongly supervised dictionary dataset ('BSL dictionary 2') on the same gestures as in the first strongly supervised

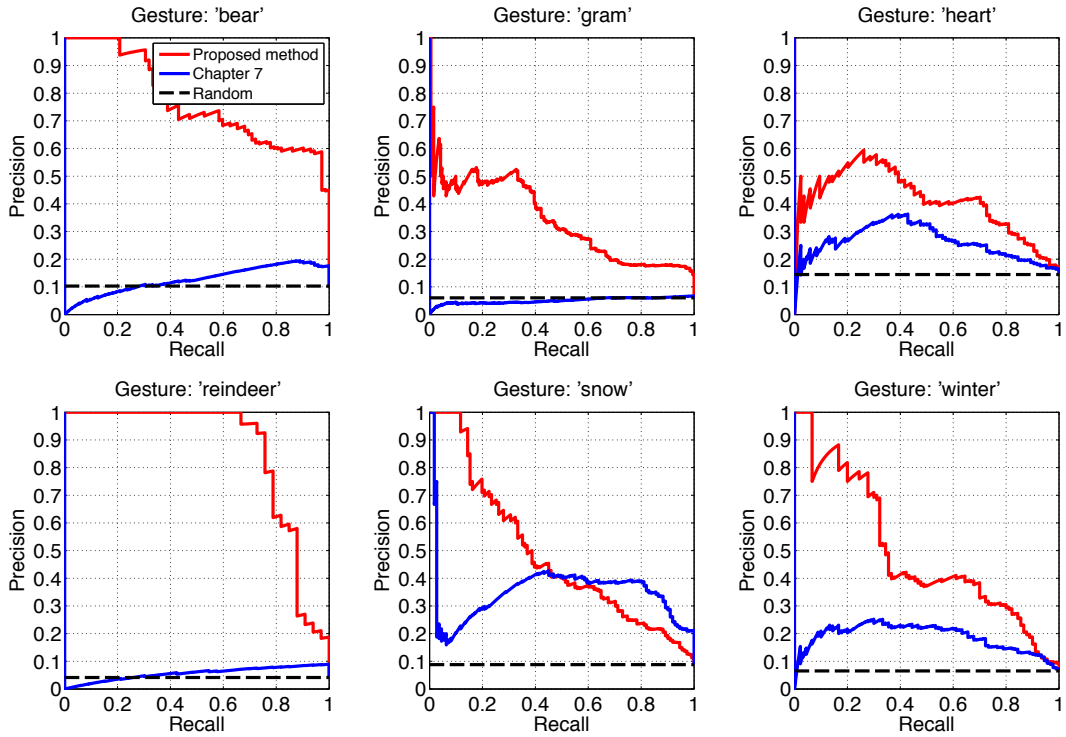


Figure 8.6: **Gesture spotting accuracy on the gesture reservoir for BSL-TV**, with a comparison to Chapter 7. PR curves are for six gestures with ground truth.

dictionary ('BSL dictionary 1'), but in a different domain, signed at different speeds by different people. The second dictionary is used as the testing set to reduce annotation effort (the BSL-TV reservoir does not come with frame-level labels). Section 3.2.3 explains how these cross-dictionary gesture 'pairs' that contain the same sign signed the same way are found.

Baseline one-shot learner. We compare our method to a classifier trained only on a sign language dictionary ('BSL dictionary 1'), without any weakly supervised additional training data from the gesture reservoir (this baseline method is illustrated at the top of Figure 8.1). The method is otherwise the same – it uses the time and space domain adaptations *etc*.

Training and testing set. The cross-dictionary gesture 'pairs' that contain the same sign signed the same way (found as explained in Section 3.2.3) define an 'in-common' set of 500 signs. The training set consists of the 150 gestures from BSL dictionary 1 from the in-common set for which a sufficient number of examples exist in the BSL-TV gesture reservoir (set to at least 16 subtitle occurrences). The testing set consists of the same set of 150 gestures from BSL dictionary 2.

Test task & evaluation measure. Each of the 150 training gestures is evaluated independently. For each gesture, the gesture classifier is applied to all 150 test gestures, one of which contains the correct gesture. The output of this step is, for each gesture classifier, a ranked list of 150 gestures (with scores). The task is to get the correct gesture first. Each gesture classifier is assigned the rank of the position in the 150-length list in which the correct gesture appears.

Results. Figure 8.7 shows recall at rank k for the baseline and our proposed method using the gesture reservoir. We clearly see that, although the baseline ranks 66% of the gestures within the first top 60, learning from the reservoir beats it, with all gestures ranked within the first 25, 13% as rank 1, 41% within the first 5, and 70% within the first 10. We believe this is due to the high training data variability that the additional supervision from the gesture reservoir provides (from multiple persons, with gestures performed with many different speeds *etc.*).

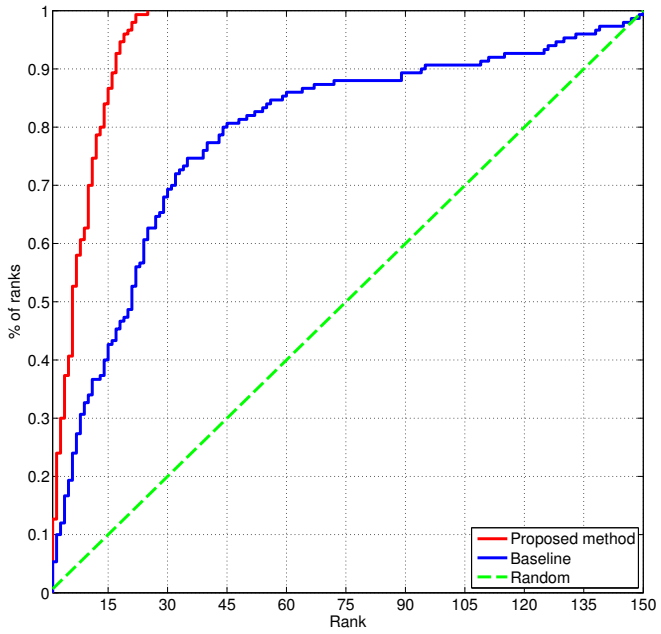


Figure 8.7: **Gesture classifier accuracy evaluated on gesture dictionary 'BSL dictionary 2'.** The graph shows the recall at rank k for the baseline and our proposed method. For example, 87% of gestures are ranked within the top 15.

Failure modes. There are two principal failure modes: first, the majority of gestures with ranks above 15 are due to several gestures out of the test gestures having very similar hand trajectories and hand shapes. With an already challenging discrimination problem, this causes confusions when the gesture in the evaluation set is performed very differently from any gesture in the training reservoir. The other major problem source is inaccurate pose estimates, which results in inaccurate hand trajectory and hand shape estimates.

Component evaluation. Each of the components of our method is evaluated by switching one off at a time, and reporting rank-15 accuracy. Changing the time alignment method from global alignment to DTW decreases the rank-15 accuracy

from 87% to 51%; switching off hand shape lowers it to 72%; switching off time alignment for the one-shot dictionary learner drops it to 46%; and switching off geometric alignment drops it to 8%. The causes for these drops in accuracy are intuitive: when turning off hand shape, signs such as 'gram' or 'reindeer' (where hand shape is informative) are less well detected; likewise, when turning off time alignment, signs such as 'snow' (with large speed variation across people) are less well detected.

Summary. Our method works despite the domain adaptations between the dictionaries and weakly supervised dataset being very challenging: different resolutions, settings, people, gesture speed and regional variations; and one domain (the dictionaries) containing non-co-articulated gestures (*i.e.* having breaks between gestures) whereas other (the weakly supervised dataset) only contains continuous gestures. To add to all of this, the supervision in the weakly supervised dataset is very weak and noisy. Despite all these challenges, we show a considerable performance boost. We consistently outperform the one-shot learning method, and achieve much higher precision and recall than previous methods in selecting similar gestures from the gesture reservoir using weak supervision.

8.5.3 Comparison on ChaLearn multi-modal dataset

On the ChaLearn dataset we define the one-shot learning domain as the training data for one person, and keep the remaining training data (of the 26 other persons) as the unlabelled 'gesture reservoir'. Only Kinect skeletons are kept for

the reservoir. We compare this setup to using all the ground truth for training.

Task. The task here is, given a test video (also containing distractors), to spot gestures and label them as one of 20 gesture categories.

Audio for gesture segmentation. Gestures only appear in a small subset of the dataset frames, so it makes sense to spot candidate windows first. To this end we use the same method as the top entries in the ChaLearn competition: segment gestures (into temporal windows) using voice activity detection (this is possible since the persons pronounce the word they gesture). However, we do not use audio for classification, since our purpose here is to evaluate our vision-based classifier. We therefore compare only to methods that do not use audio for classification, but only use it for segmentation into temporal windows (this includes the winner's method without audio classification).

Baseline, our method & upper bound. The baseline is learning from a single supervised training example (where training data comes from a single person from the 27 person training dataset; we report an average and standard deviation over each possible choice). This is compared to our method that uses the one-shot learner to extract additional training data from the unlabelled 'gesture reservoir'. The upper bound method uses all training data with manual ground truth.

Overview of experiments. In Experiment 1, we compare in detail to the competition winner [Wu et al., 2013] with the same segmentation method (audio, with our re-implementation of that method), using only skeleton features for classification, and evaluating in terms of precision and recall on the validation set. In Experiment 2, we compare to competition entrants using the standard competition evaluation measure on test data, the Levenshtein distance $L(R, T)$, where R and T are ordered lists (predicted and ground truth) corresponding to the indices of the recognised gestures (1–20); distances here are summed over all test videos and divided by the total number of gestures in ground truth.

Results for Experiment 1. Our method achieves very respectable performance using a fraction of the manually labelled data that the other competition entrants use. The competition winner's method gets Precision $P = 0.5991$ and Recall $R = 0.5929$ (higher is better) using skeleton features for classification [Wu et al., 2013]. Using this exact same setup and test data, the baseline achieves $P = 0.4012$ (std 0.015) and $R = 0.4162$ (std 0.011) – notably by only using a single training example (the winner used the whole training set containing more than 400 training examples per class). Our results are improved further to $P = 0.5835$ (std 0.021), $R = 0.5754$ (std 0.015) by using gestures extracted from the gesture reservoir, still only using one manually labelled training example per gesture. Using the whole training set yields $P = 0.6124$, $R = 0.6237$.

Results for Experiment 2. In terms of Levenshtein distance, our method improves from the baseline 0.5138 (std 0.012) to 0.3762 (std 0.015) (lower is

better). With only a single manually labelled training example (two orders of magnitude less manually labelled training data than other competition entries) we achieve similar performance to the best method using skeleton for classification ('SUMO', score 0.3165 [Escalera et al., 2013]); and using the full training set we outperform them at 0.3015.

8.5.4 Computation time.

The computation times for hand segmentation is 0.1s/frame. Time alignment is approx 0.001s per gesture pair, or 1,000s for a 1000×1000 kernel matrix. Other costs (*e.g.* space alignments, SVM training and testing, subtitle preprocessing *etc.*) are negligible in comparison (a few seconds per gesture/video).

8.6 Conclusion

We have presented a method that goes beyond learning from weak supervision only (as in Chapter 7), instead learning from both strongly and weakly supervised training data. We show this results in a much stronger classifier than if one learns from strong or weak supervision alone. We further show that this method extends beyond sign language, providing results on both the sign language dataset and a dataset of Italian gestures. The performance exceeds that of Chapter 7 and the previous state-of-the-art result on the Italian gesture dataset.

Chapter 9

Contributions and Future Work

This chapter summarises the main contributions of this thesis, and discusses future work.

This thesis has presented novel methods in two areas of computer vision: human pose estimation and gesture recognition.

For human pose estimation, we made the following contributions:

1. We proposed a co-segmentation algorithm for segmenting humans out of videos (which we use in our pose estimator), and an evaluator that predicts whether the estimated poses are likely to be correct or not.

2. We further extended this random forest-based pose estimator to new domains (with a transfer learning approach), and enhanced its predictions with new methods that predict poses sequentially (rather than independently), and use temporal information in the videos (rather than predicting the poses from a single frame).

3. We showed that convolutional neural networks can be used to estimate human pose even more accurately and efficiently than with a random forest.

4. We proposed two new convolutional network architectures, and showed how optical flow can be employed within these architectures to further improve the predictions.

For gesture recognition, we made the following contributions:

1. We explored the idea of using readily available weak supervision to learn gestures (instead of strong supervision, which is expensive to collect), and showed that we can use this weak supervision to essentially learn sign language automatically by 'watching TV'.

2. We showed that correlations between sign language signers' mouth and hand movement can be used to significantly cut down the search space when learning sign language gestures.

3. We further showed that if even a small amount of strong supervision is available (as there is for sign language, in the form of sign language video dictionaries), this strong supervision can be combined with weak supervision to learn even better models.

9.1 Future Work

This section discusses some potential future directions for the two areas explored in this thesis.

9.1.1 Pose estimation

- **Automatically extracting parameters for domain adaptation.** In Chapter 5 we proposed to use domain adaptation to generalise a pose estimator trained on long-sleeved persons to function on short-sleeved persons. In that method, the side information (the sleeve length) was manually specified for each input video. In future work this information could be automatically recovered (*e.g.* by training a sleeve length regressor from a large set of videos with containing people with different sleeve lengths).

- **Architectural improvements to ConvNets.**

  - **Spatial model on top of ConvNet.** The current ConvNet pose estimator estimates joint positions without an explicit spatial model. This spatial model could be learnt with another ConvNet (taking the heatmaps as an input), *e.g.* a conditional random field could be used as in [Jain et al., 2014a, Tompson et al., 2014].

  - **Sequential pose prediction for ConvNet.** The ConvNet pose estimator predicts joints independently. However, as we discussed in Chapter 5, the human kinematic chain places strong priors for possible locations of a joint (given the positions of the other joints).

  - **Multi-resolution input into ConvNet.** The ConvNet pose estimator predicts joints from a single fixed-height input image. However, using multiple input sizes fed into parallel networks with shared weights has been shown empirically to improve performance signifi-

cantly [Tompson et al., 2014].

– **Predicting joint coordinates and heatmaps jointly.** We explored two different kinds of networks, one that predicts joint positions directly, and one that predicts heatmaps for joints. One (heatmap) was shown to perform well on wrists (which have high variability in position), and the other (coordinates) better on shoulders and wrists (whose positions vary much less, and thus are easier to learn). This encourages a study into using both loss targets jointly – *e.g.* by computing losses separately for the two targets and having a weighed average loss on top that is then backpropagated.

– **Learning higher-level temporal features.** Our current work on temporal pose estimation focuses on the use of optical flow. It would be interesting to also explore higher level temporal features, which could potentially be learnt with spatiotemporal convolution.

– **Recurrent Neural Nets (RNNs) for time and space.** Another approach to encoding temporal information would be using RNNs, which naturally work well for tasks where the output is a sequence with dependencies between the sequences. They have been very successful in a variety of tasks – most recently in video (Google) [Ng et al., 2015]. One could also explore its use for learning a spatial model for pose estimation, as done recently for scene labelling in [Pinheiro and Collobert, 2013].

- **Evaluator network for pose estimation.** In Chapter 4 we briefly touched upon the idea of training an evaluator, which for each pose estimate predicts a score measuring the likelihood that the estimate is correct. This is useful as the score could be used to: (i) discard/down-weigh the prediction when estimating pose in video (instead interpolating the position from previous frames using *e.g.* optical flow) – the down-weighing would be straightforward to implement using our optical flow network; or (ii) 'clean up' automatically labelled additional training data (we are currently exploring this in the context of personalised pose estimation).

- **Occlusion prediction.** Our current work does not really deal with occlusions – the joints are assumed to always be visible. To rectify this, one could imagine predicting an occlusion flag (a probability for occlusion) for each joint, and taking this into account in the loss function.

- **Training data.** One of the main limitations for current pose estimation research is the lack of large training datasets, such as ImageNet for object detection. One approach would be to detect millions of humans with a person detector *e.g.* in YouTube videos and use crowdsourcing (*e.g.* Amazon mechanical turk) to label them (with a large budget). Another interesting approach would be to collect a large amount of training data with motion capture in general (*e.g.* parks, schools *etc.*) scenes (with multiple cameras) – this kind of automatic labelling has recently been explored with ConvNets by the NYU team [Elhayek et al., 2015].

9.1.2 Gesture recognition

- **Depth data and Kinect.** The work on learning sign language from TV in this thesis has been limited to somewhat low-res RGB TV broadcasts, which do not contain depth information, and thus are more difficult to track humans in. We have an ongoing project with BBC and Red Bee Media where we will be recording Kinect data in the sign language interpretation studios. The hope is that this data will be easier to learn from, and the learnt models (from Kinect output – depth, skeleton and RGB) will more readily transfer to real-life application scenarios, *e.g.* with a signer signing into a Kinect and the signing being translated into text/speech.

  This project will also provide a large amount of training data for pose estimation (with the training labels – somewhat noisy – from the Kinect tracker). This training data could be filtered with an evaluator (see above discussion on pose estimation), and the 'trusted' training samples could be used to train a pose estimator with a ConvNet – either a general RGB pose estimator, or a depth map-based pose estimator (as Kinect).

- **Linguistic models and sentence constraints.** The work in this thesis learns individual signs in a sentence separately. However, in practice many dependencies and constraints exist within sentences: co-occurrences, grammar, and word order. These could be exploited with weak sentence constraints [Bojanowski et al., 2014] or with a linguistic model (explicit, or implicitly learnt).

- **Replacing hand-engineered descriptors with learnt descriptors.** The hand and mouth descriptors *etc.* in this thesis could be learnt with a ConvNet – or the output from the last layers of a ImageNet-trained network could be used. These would replace the SIFT & HOG descriptors in this work.

- **Hand & finger tracking.** Understanding hands and hand shape is key to gesture recognition. Our work used a simple hand shape descriptor based on predicted wrist position. A much more useful descriptor would be the 3D orientation and position of both the hand and the fingers. This is difficult to obtain from low-res RGB TV broadcasts such as those used in this work – however, given the new Kinect data described above, more detailed hand tracking is now feasible. This is a well studied problem [Krejov and Bowden, 2013, Oikonomidis et al., 2011, Sharp et al., 2015], and off-the-shelf solutions to it exist [Oikonomidis et al., 2011].

Bibliography

The standard dictionary of the British sign language. DVD, 2005.

A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *PAMI*, 28(1):44–58, 2006.

K. Alahari, G. Seguin, J. Sivic, and I. Laptev. Pose estimation and segmentation of people in 3D movies. In *Proc. ICCV*, 2013.

O. Alsharif and J. Pineau. End-to-end text recognition with hybrid HMM maxout models. In *Proc. ICLR*, 2014.

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Proc. NIPS*, 2002.

M. Andriluka, S. Roth, and B. Schiele. Discriminative appearance models for pictorial structures. *IJCV*, 99(3):259–280, 2012.

O. Aran, T. Burger, A. Caplier, and L. Akarun. A belief-based sequential fusion approach for fusing manual signs and non-manual signals. *Pattern Recognition Letters*, 42(5):812–822, 2009.

A. Baisero, F. T Pokorny, D. Kragic, and C. Ek. The path kernel. In *Proc. International Conference on Pattern Recognition Applications and Methods*, 2013.

B. Benfold and I. Reid. Colour invariant head pose classification in low resolution video. In *Proc. BMVC*, 2008.

P. Bojanowski, F. Bach, , I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. In *Proc. ICCV*, 2013.

P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *Proc. ECCV*, 2014.

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proc. CVPR*, 2009.

R. Bowden, D. Windridge, T. Kadir, A. Zisserman, and J. M. Brady. A linguistic feature vector for the visual interpretation of sign language. In *Proc. ECCV*, 2004.

Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. ICCV*, 2001.

C. Bregler and Y. Konig. "Eigenlips" for robust speech recognition. In *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 1994.

C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. CVPR*, 1998.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Bristol Centre for Deaf Studies. Signstation. `http://www.signstation.org`, 2014. [Online; accessed 1-March-2014].

P. Buehler. *Automatic Learning of British Sign Language from Signed TV Broadcasts*. PhD thesis, University of Oxford, 2010.

P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language TV broadcasts. In *Proc. BMVC*, 2008.

P. Buehler, M. Everingham, and A. Zisserman. Learning sign language by watching TV (using weakly aligned subtitles). In *Proc. CVPR*, 2009.

P. Buehler, M. Everingham, and A. Zisserman. Employing signed TV broadcasts for automated learning of British sign language. In *Workshop on Representation and Processing of Sign Languages*, 2010.

P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Upper body detection and tracking in extended signing sequences. *IJCV*, 95(2):180–197, 2011.

L. Campbell, D. Becker, A. Azarbayejani, A. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. In *Proc. FG*, 1996.

X. Chai, G. Li, Y. Lin, Z. Xu, Y. Tang, X. Chen, and M. Zhou. Sign language recognition and translation with Kinect. In *Proc. FG*, 2013.

Y. Chai, V. Lempitsky, and A. Zisserman. Bicos: A bi-level co-segmentation method for image classification. In *Proc. ICCV*, 2011.

J. Charles and M. Everingham. Learning shape models for monocular human pose estimation from the Microsoft Xbox Kinect. In *Proc. ICCV Workshops*, 2011.

J. Charles, T. Pfister, M. Everingham, and A. Zisserman. Automatic and efficient human pose estimation for sign language videos. *IJCV*, 110(1):70–9, 2013a.

J. Charles, T. Pfister, D. Magee, D. Hogg, and A. Zisserman. Domain adaptation for upper body pose tracking in signed TV broadcasts. In *Proc. BMVC*, 2013b.

J. Charles, T. Pfister, D. Magee, D. Hogg, and A. Zisserman. Upper body pose estimation with temporal sequential forests. *Proc. BMVC*, 2014.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *BMVC*, 2014.

X. Chen and A. Yuille. Articulated pose estimation with image-dependent preference on pairwise relations. In *Proc. NIPS*, 2014.

A. Cherian, J. Mairal, K. Alahari, and C. Schmid. Mixing body-part sequences for human pose estimation. In *Proc. CVPR*, 2014.

W. Chunli, G. Wen, and M. Jiyong. A real-time large vocabulary recognition system for Chinese Sign Language. *Gesture and sign language in HCI*, 2002.

H. Cooper and R. Bowden. Learning signs from subtitles: A weakly supervised approach to sign language recognition. In *Proc. CVPR*, 2009.

H. Cooper, E.J. Ong, N. Pugeault, and R. Bowden. Sign language recognition using sub-units. *J. Machine Learning Research*, 13:2205–2231, 2012.

T. Cootes, M. Ionita, C. Lindner, and P. Sauer. Robust and accurate shape model fitting using random forest regression voting. In *Proc. ECCV*, 2012.

A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2):81–227, 2012.

M. Cuturi. Fast global alignment kernels. In *ICML*, 2011.

M. Cuturi, J.P. Vert, Ø. Birkenes, and T. Matsui. A kernel for time series based on global alignments. In *Proc. International Conference on Acoustics, Speech, and Signal Processing*, 2007.

N. Dalal and B Triggs. Histogram of Oriented Gradients for Human Detection. In *Proc. CVPR*, 2005.

M. Dantone, J. Gall, G. Fanelli, and L. Van Gool. Real-time facial feature detection using conditional regression forests. In *Proc. CVPR*, 2012.

T. Dietterich, R. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

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. *Proc. ICML*, 2014.

P. Dreuw, T. Deselaers, D. Rybach, D. Keysers, and H. Ney. Tracking using dynamic programming for appearance-based sign language recognition. In *Proc. FG*, 2006.

O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *Proc. CVPR*, 2009.

S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE MM*, 2000.

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

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

A. Elhayek, E. de Aguiar, J. Tompson, A. Jain, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. Efficient convnet-based marker-less motion capture in general scenes with a low number of cameras. In *Proc. CVPR*, 2015.

H. Ershaed, I. Al-Alali, N. Khasawneh, and M. Fraiwan. An Arabic sign language computer interface using the Xbox Kinect. In *Proc. Annual Undergraduate Research Conf. on Applied Computing*, 2011.

S. Escalera, J. Gonzalez, X. Baro, M. Reyes, O. Lopes, I. Guyon, V. Athistos, and H.J. Escalante. Multi-modal gesture recognition challenge 2013: Dataset and results. In *Proc. ICMI*, 2013.

M. Everingham and A. Zisserman. Identifying individuals in video by combining generative and discriminative head models. In *Proc. ICCV*, 2005.

M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *Proc. BMVC*, 2006.

M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *Image and Vision Computing*, 27(5), 2009.

G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Proc. CVPR*, 2011.

G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. Random forests for real time 3D face analysis. *IJCV*, 101(3):437–458, 2012.

S. Fanello, I. Gori, G. Metta, and F. Odone. Keep it simple and sparse: real-time action recognition. *J. Machine Learning Research*, 14(1):2617–2640, 2013.

A. Farhadi and D. Forsyth. Aligning asl for statistical translation using a discriminative word model. In *Proc. CVPR*, 2006.

A. Farhadi, D. Forsyth, and R. White. Transfer learning in sign language. In *Proc. CVPR*, 2007.

P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Proc. CVPR*, 2000.

P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

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

P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Proc. CVPR*, 2010.

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

M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.

D. Forsyth and M. Fleck. Body plans. In *Proc. CVPR*, 1997.

A. Gaidon, Z. Harchaoui, and C. Schmid. A time series kernel for action recognition. In *Proc. BMVC*, 2011.

R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *Proc. ICCV*, 2011.

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

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

G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. In *Proc. CVPR*, 2014.

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *J. Machine Learning Research*, 2013.

I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *Proc. ICLR*, 2014.

I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, and H. Escalante. ChaLearn gesture challenge: Design and first results. In *Proc. CVPR Workshops*, 2012.

I. Guyon, V. Athitsos, P: Jangyodsuk, H. Escalante, and B. Hamner. Results and analysis of the ChaLearn gesture challenge 2012. In *Proc. ICPR*, 2013.

D. Hochbaum and V. Singh. An efficient algorithm for co-segmentation. In *Proc. ICCV*, 2009.

D. Hogg. Model-based vision: a program to see a walking person. *Image and vision computing*, 1(1):5–20, 1983.

G. Hua, M. Yang, and Y. Wu. Learning to estimate human pose with data driven belief propagation. In *Proc. CVPR*, 2005.

M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *Proc. NIPS Workshops*, 2014.

A. Jain, J. Tompson, M. Andriluka, G. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. *Proc. ICLR*, 2014a.

A. Jain, J. Tompson, Y. LeCun, and C. Bregler. MoDeep: A deep learning framework using motion features for human pose estimation. *Proc. ACCV*, 2014b.

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.

J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields - an efficient, non-parametric approach to image labeling problems. In *Proc. CVPR*, 2012.

Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

S. Johnson and M. Everingham. Combining discriminative appearance and segmentation cues for articulated human pose estimation. In *Proc. ICCV Workshops*, 2009.

N. Jojic and Y. Caspi. Capturing image structure with probabilistic index maps. In *Proc. CVPR*, 2004.

N. Jojic and B. Frey. Learning flexible sprites in video layers. In *Proc. CVPR*, 2001.

A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *Proc. CVPR*, 2010.

T. Kadir, R. Bowden, E. J. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *Proc. BMVC*, 2004.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014.

V. Kazemi, M. Burenius, H. Azizpour, and J. Sullivan. Multi-view body part recognition with random forests. In *Proc. BMVC*, 2013.

Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *Proc. ICCV*, 2007.

D. Kelly, J. McDonald, and C. Markham. Weakly supervised training of a sign language recognition system using multiple instance learning density matrices. *Trans. Systems, Man, and Cybernetics*, 41(2):526–541, 2011.

O. Koller, H. Ney, and R. Bowden. Read my lips: Continuous signer independent weakly supervised viseme recognition. In *Proc. ECCV*, 2014.

P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. Geof: Geodesic forests for learning coupled predictors. In *Proc. CVPR*, 2013.

P. Krejov and R. Bowden. Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima. In *Proc. FG*, 2013.

R. Krishnan and S. Sarkar. Similarity measure between two gestures using triplets. In *Proc. CVPR Workshops*, 2013.

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

P. Kumar, P. Torr, and A. Zisserman. Learning layered motion segmentations of video. *IJCV*, 76:301–319, 2008.

P. Kumar, P. Torr, and A. Zisserman. Efficient discriminative learning of parts-based models. In *Proc. ICCV*, 2009.

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

V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28(9): 1465–1479, 2006.

R. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Proc. FG*, 1998.

Y. Lin, X. Chai, Y. Zhou, and X. Chen. Curve matching from the view of manifold for sign language recognition. *Proc. ACCV Workshops*, 2014.

D. Lowe. Object recognition from local scale-invariant features. In *Proc. ICCV*, 1999.

T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *Proc. ICCV*, 2011.

O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Proc. NIPS*, 1998.

S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics*, 37(3):311–324, 2007.

T. Moeslund. *Visual analysis of humans: looking at people*. Springer, 2011.

G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *Proc. ECCV*, 2002.

S. Nayak, K. Duncan, S. Sarkar, and B. Loeding. Finding recurrent patterns from continuous sign language sentences for automated extraction of signs. *J. Machine Learning Research*, 13(1):2589–2615, 2012.

J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *arXiv preprint arXiv:1503.08909*, 2015.

T. D. Nguyen and S. Ranganath. Tracking facial features under occlusions and recognizing facial expressions in sign language. In *Proc. FG*, 2008.

T. D. Nguyen and S. Ranganath. Recognizing Continuous Grammatical Marker Facial Gestures in Sign Language Video. In *Proc. ACCV*, 2010.

I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proc. BMVC*, 2011.

T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

E. J. Ong and R. Bowden. Robust lip-tracking using rigid flocks of selected linear predictors. In *Proc. FG*, 2008.

S. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *PAMI*, 27(6):873–891, 2005.

M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Weakly supervised object recognition with convolutional neural networks. Technical report, INRIA, 2014a.

M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. CVPR*, 2014b.

J. O'Rourke and N. Badler. Model-based image analysis of human motion using constraint propagation. *PAMI*, 2(6):522–536, 1980.

M. Osadchy, Y. LeCun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *JMLR*, 8:1197–1215, 2007.

M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *PAMI*, 32(3):448–461, 2010.

N. Payet and S. Todorovic. $(RF)^2$ - random forest random field. In *Proc. NIPS*, 2010.

E. Petajan, B. Bischoff, D. Bodoff, and M. Brooke. An improved automatic lipreading system to enhance speech recognition. In *SIGCHI*, 1988.

T. Pfister, J. Charles, M. Everingham, and A. Zisserman. Automatic and efficient long term arm and hand tracking for continuous sign language TV broadcasts. In *Proc. BMVC*, 2012.

T. Pfister, J. Charles, and A. Zisserman. Large-scale learning of sign language by watching TV (using co-occurrences). In *Proc. BMVC*, 2013.

T. Pfister, J. Charles, and A. Zisserman. Domain-adaptive discriminative one-shot learning of gestures. *Proc. ECCV*, 2014a.

T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. *Proc. ACCV*, 2014b.

P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. *arXiv preprint arXiv:1306.2795*, 2013.

A. Prest, C. Schmid, and V. Ferrari. Weakly supervised learning of interactions between humans and objects. *PAMI*, 34(3):601–614, 2012.

N. Pugeault and R. Bowden. Spelling it out: Real-time ASL fingerspelling recognition. In *Proc. ICCV Workshops*, 2011.

D. Ramanan. Learning to parse images of articulated bodies. In *Proc. NIPS*, 2006.

D. Ramanan, D. A. Forsyth, and A. Zisserman. Strike a pose: Tracking people by finding stylized poses. In *Proc. CVPR*, 2005.

S. Rautaray and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *Proc. CVPR Workshops*, 2014.

X. Ren, A. C. Berg, and J. Malik. Recovering human body configurations using pairwise constraints between parts. In *Proc. ICCV*, 2005.

K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image understanding*, 59(1):94–115, 1994.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *Proc. ACM SIGGRAPH*, 2004.

C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. In *Proc. CVPR*, 2006.

H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

H. Sakoe and S. Chiba. A similarity evaluation of speech patterns by dynamic programming. In *Proc. Nat. Meeting of Institute of Electronic Communications Engineers of Japan*, 1970.

J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Proc. CVPR*, 2010.

B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *Proc. CVPR*, 2013.

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

B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *Proc. CVPR*, 2011.

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *Proc. ICLR*, 2014.

G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. ICCV*, 2003.

T. Sharp. Implementing decision trees and forests on a GPU. In *Proc. ECCV*, 2008.

T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Proc. accurate, robust, and flexible real-time hand tracking. In *CHI*, 2015.

H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *Proc. NIPS*, 2001.

J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Proc. CVPR*, 2008.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. CVPR*, 2011.

J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *PAMI*, 35(12):2821–2840, 2013.

K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *NIPS*, 2014.

J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *IJCV*, 67(2):189–210, 2006.

T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer, 1997.

T. Starner, J. Weaver, and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *PAMI*, 20(12):1371–1375, 1998.

W. Stokoe. Sign language structure: An outline of the visual communication systems of the american deaf. *J. Deaf Studies and Deaf Education*, 2005.

M. Sun, P. Kohli, and J. Shotton. Conditional regression forests for human pose estimation. In *Proc. CVPR*, 2012.

R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *Proc. CVPR*, 2000.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. CVPR*, 2014.

J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proc. CVPR*, 2012.

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

J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. *Proc. CVPR*, 2015.

A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. *CVPR*, 2014.

D. Tran and D. Forsyth. Improved human parsing with a full relational model. In *Proc. ECCV*, pages 227–240, 2010.

Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 2010.

C. Vogler and S. Goldenstein. Analysis of facial expressions in american sign language. In *Proc. Intl. Conf. on Universal Access in Human-Computer Interaction*, 2005.

C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *Proc. ICCV*, pages 363–369, 1998.

C. Vogler and D. Metaxas. Handshapes and movements: Multiple-channel American sign language recognition. In *Proc. Gesture-Based Communication in Human-Computer Interaction Workshop*, 2004.

U. von Agris, M. Knorr, and K.F. Kraiss. The significance of facial features for automatic sign language recognition. In *Proc. FG*, 2008.

J. Wan, Q. Ruan, W. Li, and S. Deng. One-shot learning gesture recognition from RGB-D data using bag of features. *J. Machine Learning Research*, 14(1):2549–2582, 2013.

P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proc. ICCV*, 2013.

M. Wheatley and A. Pabsch. *Sign language legislation in the European Union*. European Union of the Deaf, 2010.

World Federation of the Deaf, 2012. URL `http://www.wfdeaf.org/human-rights/crpd/sign-language`.

J. Wu, J. Cheng, C. Zhao, and H. Lu. Fusing multi-modal features for gesture recognition. In *Proc. ICMI*, 2013.

J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proc. CVPR*, 1992.

H. Yang and I. Patras. Sieving regression forest votes for facial feature detection in the wild. In *Proc. ICCV*, 2013.

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

Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *PAMI*, 35(12):2878–2890, 2013.

Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti. American sign language recognition with the Kinect. In *Proc. ICMI*, 2011.

M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *Proc. ECCV*, 2014.

F. Zhou and F. De la Torre. Generalized time warping for multi-modal alignment of human motion. In *Proc. CVPR*, 2012.

Z. Zhou, G. Zhao, and M. Pietikäinen. Towards a practical lipreading system. In *Proc. CVPR*, 2011.

X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proc. CVPR*, 2012.