Ceilidh dance recognition from an overhead camera

Jeremiah Aizeboje

Master of Science
School of Informatics
University of Edinburgh
2016

Abstract

This thesis presents a novel linear-chain CRF model for ordered state transition modelling aiming to recognise 2 ceilidh dances. The algorithm uses a two stage training process. It learns the feature of each dancer and then it learns the dance itself. The final trained model was evaluated using k-fold cross validation. The proposed linear-chain CRF model had an accuracy of 96.85% for dance 1 and 94.42% for dance 2 where the state lag tolerance is set to +/- 5 frames. The proposed linear-chain CRF model also outperformed the traditional linear-chain CRF model by an average of 6.79%.

Acknowledgements

I want to thank my supervisor Prof. Robert Fisher for sharing his wisdom with me at the times of need. His feedback and problem solving ideas were always on point. I would to thank Edigleison Carvalho for his additional supervision and keeping me on track.

I also want to thank my family and friends who encouraged me to stay focused and helped me through the demanding parts of the thesis.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jeremiah Aizeboje*)

Table of Contents

Chapter 1

Introduction

Nowadays, computers and video surveillance systems are relied on for security purposes. Humans are also relied on to use such tools to make important decisions. However, humans have certain defects that prevent them from being efficient even when the tools have already been provided. Behavioural analysis and activity recognition is a stimulating topic in the medical field especially during controlled experiments i.e. to understand the interactions between micro cells over a long period of time.

The need to analyse a behaviour helps one to gain a better insight about the object in question. Animals are primarily used for biological experiments. The main issue is that they have no way of intelligently explaining how they feel during an experiment, so biologists look out for certain behaviours. Behavioural patterns are infinite and might be dismissed at first sight to the human eye. On the other hand, computers are known for their multi-tasking abilities and their computational powers. Adding some form of intelligence to such machines with the aim to classify similar behavioural patterns together seems like a formidable approach. This would yield better and more accurate experimental results when compared to relying only on a humans wit.

It goes without saying that higher quality video (high video resolution and number of frames per second) entails more defining details and would go a long way in producing more accurate results. However, the higher the video quality, the more power and time required to process this videos. This problem introduces some form of quality versus accuracy trade-off. These trade-off helps to decide the bounds of this project. It helps in choosing what level of behaviour to monitor. For example, a real-time hand monitoring application using high quality video would be able to model finger

gestures, whereas using low quality videos would only enable modelling of the hand movement as a whole. With the current speed at which technology is advancing, the quality-accuracy trade-off is becoming an important topic of interest.

The aim of this project is to push the capabilities of video analysis through the use of a simple context: ceilidh dancing. Ceilidh dancing is a group activity where each of the dancers have specific roles to play. These dancers create some patterns when viewed from above. This is an interesting fact because it is these patterns that are seen as the behaviour of the dance. Applications that emits patterns can similarly use the approach used in this project to recognise their different states.

On each frame, the position of each dancer is spatially and locally related. The same can also be said of their activities. In order to identify a dancers' activity from an observation sequence, a spatio-temporal model is a primary requirement. Such a model uses each frame in an observation sequence to learn the activities within the observation sequence. Treating each frame as independent and identically distributed (i.i.d.) would fail to make the most of the sequential patterns that is observed in the transition between neighbouring frames. A conditional random field (CRF) relaxes the i.i.d. assumption and makes it a suitable model for this project.

1.1 Goal

A CRF would be mainly used to achieve the goals of this project as well as 640 by 480 videos of 10 ceilidh dancers from an overhead camera. The goals of this project are as follows:

- To explore how a probabilistic method can be used to recognise activities.
- To create an activity recognition system
- To create a ceilidh dance differentiation system.

The completion of these goals determine the success of this project. As this project would be built around a probabilistic framework, probabilistic models would be mainly reviewed in the next chapter.

1.2 Overview

This project can be broken down into 3 different phases:

I **Object detection and tracking:** The position of each dancer at every time step would be required to test the CRF. Object detection and tracking is the approach that will be used to achieve this task. Chapter 4 explains more about the approach used.

II **Learning activity features:** There are 10 dancers and 12 features observed in all the dances. About half of the features are higher dimensional or real valued; the features are not binary in nature. For example, if we had a feature to detect if a kettle was hot, instead of having a Bernoulli distribution explain the hot feature $\{on, off\}$, a multinomial approach is used instead to better understand how hot the kettle is $\{high, medium, low, off\}$. Learning the activity features would help reduce the number of features the dance CRF (chapter 6) is trained with in order to learn the dance.

III **Learning the dance:** This phase involves training the dance CRF using the preprocessed feature values (chapter 5). The preprocessing step helped reduce the dimensionality of the features which is analogous to the pooling layer of a convolution neural network. It also helped reduce the amount of noise in the data.

Teaching the CRF with the correct data is a crucial task in this project. Correct labelling of the training data is fundamental to the accuracy obtained during training. Chapter 3 discusses the approach taking to ensure correct labelling.

The proposed linear-chain CRF model produced 96.85% for dance 1 and 94.42% for dance 2. The experiments carried out is discussed in chapter 7. The proposed linear-chain CRF model also outperformed the traditional linear-chain CRF model by an average of 6.79%.

1.3 The Dataset

The videos were obtained from an overhead camera staring approximately 90° to the ground as seen in figure 4.1 There are 2 different ceilidh dances used for this project.

Dance 1:

- 9 different videos recorded
- contains 10 ceilidh dancers
- the dance has a duration of 16 to 20 seconds at a rate of approximately 8fps
- 640 by 480 resolution
- 15 labelled states

Dance 2:

- 7 different videos recorded
- contains 10 ceilidh dancers
- the dance has a duration of 41 to 44 seconds at a rate of approximately 8fps
- 640 by 480 resolution
- 44 labelled state

Chapter 2

Background

Video analysis via automated means has been frequently researched into. Multiple approaches from regression techniques [3] to deep belief networks [21] have been explored. One of the main goals of video analysis is to somewhat make sense of what is going on within the video. Such videos may contain complex activities. This approach to video analysis has influenced computer vision research fields like object detection and tracking as well as activity recognition.

2.1 Object detection and tracking

Object detection and tracking is a core topic in computer vision. There are many approaches to detecting and tracking objects [10]. Lee et al. [11] use colour histogram to detect object. This technique is applicable where the objects to be tracked have distinctive colours. Lee et al. combined this technique with a local binary pattern approach to detect objects at different scales. The local binary is calculated using the intensity of neighbouring pixels.

Using colour histograms to track multiple objects is an approach being used by Morioka et. al [13]. However, before such an approach can be used, the objects of interest needs to be extracted. A simple background subtraction approach is used [19]

$$foregorund_{image} = current_{image} - background_{image} \tag{2.1}$$

Overall, object detection and tracking algorithms are usually unique and tweaked to the task in hand as there are many factors (i.e. lighting, noise) that could affect its performance .

2.2 Activity Recognition

An activity has two main states: start and end. It can be described as the path taken from the start state to the end state. Activity recognition aims to understand the path between the start and end states as each distinct activity should ideally have unique paths.

A probabilistic reasoning approach is currently being applied to activity recognition problems. The main problem with a supervised learning approach is the vast amount of labelled ground truth data required to train the model. However, Hossain et al. [6] attempt to solve this problem by introducing an active learning factor. Their approach proposes the use of a dynamic k-means clustering to classify activities for the active learner. An interesting finding from this approach was the ability for the active learner to discover unseen activities. This is useful to prevent wrong classification of activities. Also, if a traditional model only relied on the ground truth data, it could fail to correctly classify unseen activities.

Probabilistic graphical models such as the Markov model and its descendants are often used in modelling sequential data. The next section would dive into the Markov model and a few of its descendants.

2.2.1 Markov Model

The Markov model [2] is well known for its sequential modelling properties of time series data. That is, it does not perceive observations to be independently and identically distributed (i.i.d.) ; it conditions the observations at time $t-1$ to predict the conditional distribution of the observation at time t as seen in equation 2.2

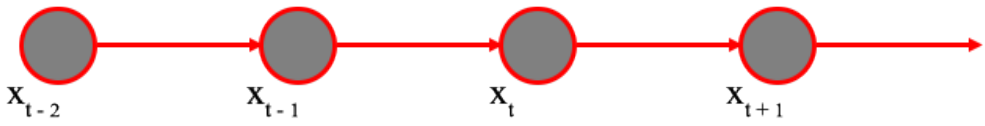


Figure 2.1: A first order Markov chain. The graphical model of equation 2.2

$$p(\mathrm{x}_t|\mathrm{x}_{t-1}) \tag{2.2}$$

where x_t is the observed data at time t.

This is also known as a first-order Markov model. A second-order Markov model would allow influence from the previous two observations, at time $t-1$ and $t-2$, in order to predict the conditional distribution of the observation at time t and so on. The higher the order, the more complex the model. For example, one might wish to use the Markov model to predict the weather if it is going to rain today based on if it rained in the past week.

The Markov chain also assumes that the observation at $t-1$ (for a first order Markov) incorporates all previous observations, hence equation 2.2 is derived from

$$p(x_t|x_1,...,x_{t-1}) = p(x_t|x_{t-1}) \tag{2.3}$$

where the joint distribution of the model for T observations is

$$p(x_1,...,x_T) = p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1}) \tag{2.4}$$

The next section would illustrate how this model is extended by the hidden Markov model (HMM).

2.2.2 Hidden Markov Model

HMMs are directed graphical models [16]. The HMM introduces latent or unobserved variables to the Markov model. These variables are the hidden states of the model.

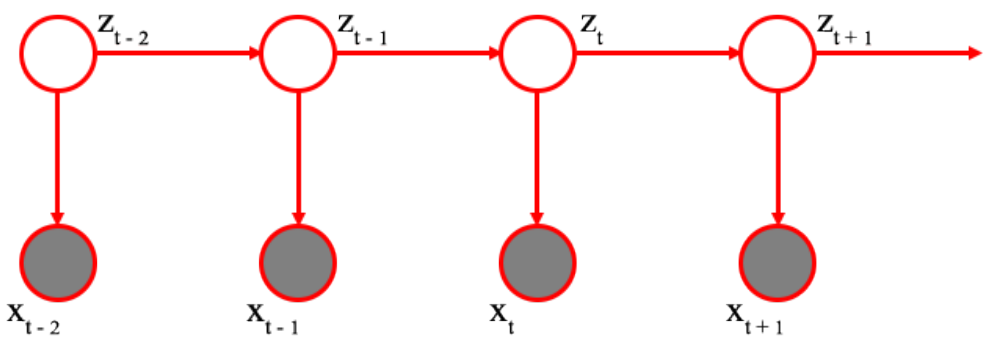


Figure 2.2: A first order HMM. The graphical model of equation 2.5

and the joint distribution is

$$p(x_{1:T}, z_{1:T}) = p(z_1)\prod_{t=2}^{T} p(\mathrm{z}_t|\mathrm{z}_{t-1})\prod_{t=2}^{T} p(\mathrm{x}_{t-1}|\mathrm{z}_{t-1}) \tag{2.5}$$

where z_t is the hidden state of the model at time t and x_t is the observed data at time t.

In comparison to the example given for the Markov model, one might wish to use the HMM to predict the weather if it rained last night based on some observations like if the grass is wet. The HMM model partially observes the previous data while the Markov model fully observes the previous data. This model is known to play the key role in the human genome project [5] and speech recognition applications [17].

2.2.3 Linear-Chain Conditional Random Field

A Conditional random field (CRF) can be viewed as a descendant of the HMM approach. It is well known for its uses in natural language processing [23][22][14][15]. In comparison to HMM, linear-chain CRFs [20] are undirected graphical models and also discriminative in nature which means it is built on a model that is conditionally distributed $p(z|x)$. The conditional probability of a linear-chain CRF takes the form

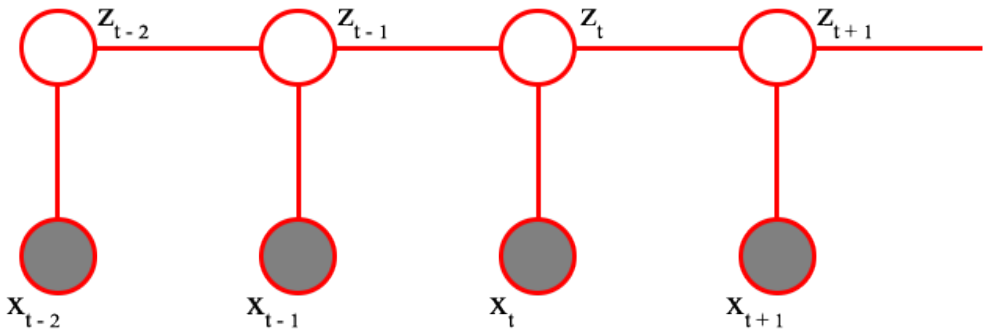


Figure 2.3: A Linear-Chain CRF.

$$p(z_{1:T}|x_{1:T}) = \frac{1}{Z}\exp\left(\sum_{t=1}^{T}\sum_{i=1}^{F} w_i f_i(z_{t-1}, z_t, x_{1:T}, t)\right) \tag{2.6}$$

where T is the number of frames in a single observation sequence, F is the number of features, f_i is the feature function (explained in chapter 5.1), w_i is the weight and Z is the normaliser also known as a *partition function*. The partition function ensures that the model produces valid probability values between 0 and 1.

$$Z = \sum_{z_{1:T}} \exp\left(\sum_{t=1}^{T} \sum_{i=1}^{F} w_i f_i(z_{t-1}, z_t, x_{1:T}, t) \right) \tag{2.7}$$

Training or estimating the CRF parameters involves finding w that maximizes the conditional log likelihood of the training data. This requires a data set (that is labelled and sequential in nature) for training $\{ (\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), ..., (\mathbf{x}^{(N)}, \mathbf{z}^{(N)}) \}$ where $\mathbf{x}^{(n)} = x_{1:T}^{(n)}$ is an observation sequence. The objective function takes the form

$$\sum_{n=1}^{N} \log p(\mathbf{z}^{(n)} | \mathbf{x}^{(n)}) \tag{2.8}$$

The gradient of the model is used to train the weights of the CRF [7]: $w_i \leftarrow w_i + \delta w_i$. The gradient can be computed as:

$$\frac{\partial}{\partial w_i} \sum_{n=1}^{N} \log p(\mathbf{z}^{(n)} | \mathbf{x}^{(n)}) \tag{2.9}$$

$$= \frac{\partial}{\partial w_i} \sum_{n=1}^{N} \left(\sum_{t=1}^{T} \sum_{i=1}^{F} w_i f_i(z_{t-1}^{(n)}, z_t^{(n)}, x_{1:T}^{(n)}, t) - \log Z^{(n)} \right) \tag{2.10}$$

$$= \sum_{n=1}^{N} \sum_{t=1}^{T} \left(f_i(z_{t-1}^{(n)}, z_t^{(n)}, x_{1:T}^{(n)}, t) - E_{z'_{t-1}, z'_t}[f_i(z'_{t-1}, z'_t, x_{1:T}^{(n)}, t)] \right) \tag{2.11}$$

where the derivative of $\log Z$ is

$$E_{z'_{t-1}, z'_t}[f_i(z'_{t-1}, z'_t, x_{1:T}^{(n)}, t)] = \sum_{z'_{t-1}, z'_t} p(z'_{t-1}, z'_t | x_{1:T}^{(n)}) f_i(z'_{t-1}, z'_t, x_{1:T}^{(n)}, t) \tag{2.12}$$

The derivative of $\log Z$ is made up of the edge marginal probability $p(z'_{t-1}, z'_t | x_{1:T}^{(n)})$. An edge can be defined as the connection between two nodes (z_{t-1} and z_t).

Once a CRF has been trained, it is only normal to use it for prediction or inference. Similarly to the HMM, the forward-backward algorithm can also be used [16][7].

The forward procedure:

$$\alpha_{i=0}(z|x) = \begin{cases} 1, & \text{if } z = \text{ start} \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

$$\alpha_{i>0}(x) = \alpha_{i-1}(x) M_i(x) \tag{2.14}$$

where $\alpha_i(x)$ is the forward vector with i its index $\{0,..,m+1\}$

The backward procedure:

$$\beta_{i=m+1}(z|x) = \begin{cases} 1, & \text{if } z = \text{ stop} \\ 0, & \text{otherwise} \end{cases} \tag{2.15}$$

$$\beta_{i<m+1}(x)^{\mathrm{T}} = M_{i+1}(x)\beta_{i+1}(x) \tag{2.16}$$

where $\beta_i(x)$ is the backward vector

Both procedures use $M(x)$ the exponential of the state transition matrix $\Lambda(x)$.

$$M_i(z_{t-1}, z_t|x_{1:T}) = \exp(\Lambda_i(z_{t-1}, z_t|x_{1:T})) \tag{2.17}$$

$$M_i(x) = [M_i(z_{t-1}, z_t|x_{1:T})] \tag{2.18}$$

The edge marginal probability used in equation 2.12 can further be explained as

$$p(z'_{t-1}, z'_t|x^{(n)}_{1:T}) = \frac{\alpha_{i-1}(z'_{t-1}|x^{(n)}_{1:T})M_i(z'_{t-1}, z'_t|x^{(n)}_{1:T})\beta_i(z'_t|x^{(n)}_{1:T})}{Z(x^{(n)}_{1:T})} \tag{2.19}$$

A more comprehensive description of the model exists in the paper, ”An Introduction to Conditional Random Fields”, by Sutton and McCallum. [20].

Chapter 3

Data Labelling

The data required for this project can be divided into three sets:

1. Positional labelling for each dancer
2. Feature labelling for each dancer
3. State labelling for each dance

Positional labelling for each dancer is an easy task. All that is required is to record the (x,y) coordinates of each dancer within a frame for each dance (figure 3.2).

Labelling the feature for each dancer is a more tedious task because the video has about 8 frames per second (fps). This is a low fps to capture the full sequential properties of fast moving objects, especially an object with non-linear motion. A human judgemental call is required to determine when a feature begins and ends. For example, a dancer could be seen walking left, then walking down but after observing more frames, it can be later seen that the dancer is actually rotating anti-clockwise. Such features are hard for humans to be precise on during the labelling task. The CRF uses 12 features; walk left, walk right, walk down, walk up, twirling, standing, north, south, west, east, horizontal and vertical. These features are discussed in chapter 5

An even more difficult task is the state labelling for each dance. Ideally, in the context of this project, a state can be defined as an activity (feature) change between one or more dancers. The problem is that some of the dancers could be out of sync with the other dancers. For example, assuming the transition from state 7 to 8 is defined the activity change of 6 dancers but only 2 dancers had an activity change. Then 3

time steps later, the remaining 4 dancers had an activity change. Where does this state begin? Is it when 2 of the dancers have changed activity or when all 6 have changed their activity? This problems was solved by taking an average estimate between the start of the state transition (when at least 1 dancer has changed activity) and end of the state transition (when all 6 dancers have changed) as a new state. The first dance is broken down into 15 states and the second dance into 44 states.

Not all the data provided had exactly the same pattern. Some observation had distorted dance patterns (especially from the first dance); variations of the same dance. This meant the CRF would be trained some noisy observations. This would help the CRF to generalise properly during inference.

In each of the dances, the data is strictly labelled to have an ordered state sequence. This means that during a dance, the transition from one state to another can only be observed once and after a state has changed that state can no longer be observed (figure 3.1). All this information can be encoded in the state transition matrix of the CRF.

FROM \ TO	State 1	State 2	State 3	State 4
State 1	*0.95*	*0.05*	*0*	*0*
State 2	*0*	*0.85*	*0.15*	*0*
State 3	*0*	*0*	*0.90*	*0.10*
State 4	*0*	*0*	*0*	*1.00*

Figure 3.1: A 4-state transition matrix

Figure 3.1 illustrates an example of a CRF's state transition matrix of a dance with only 4 states. The state transition diagram for the two dance pattern are given in

Appendix A.

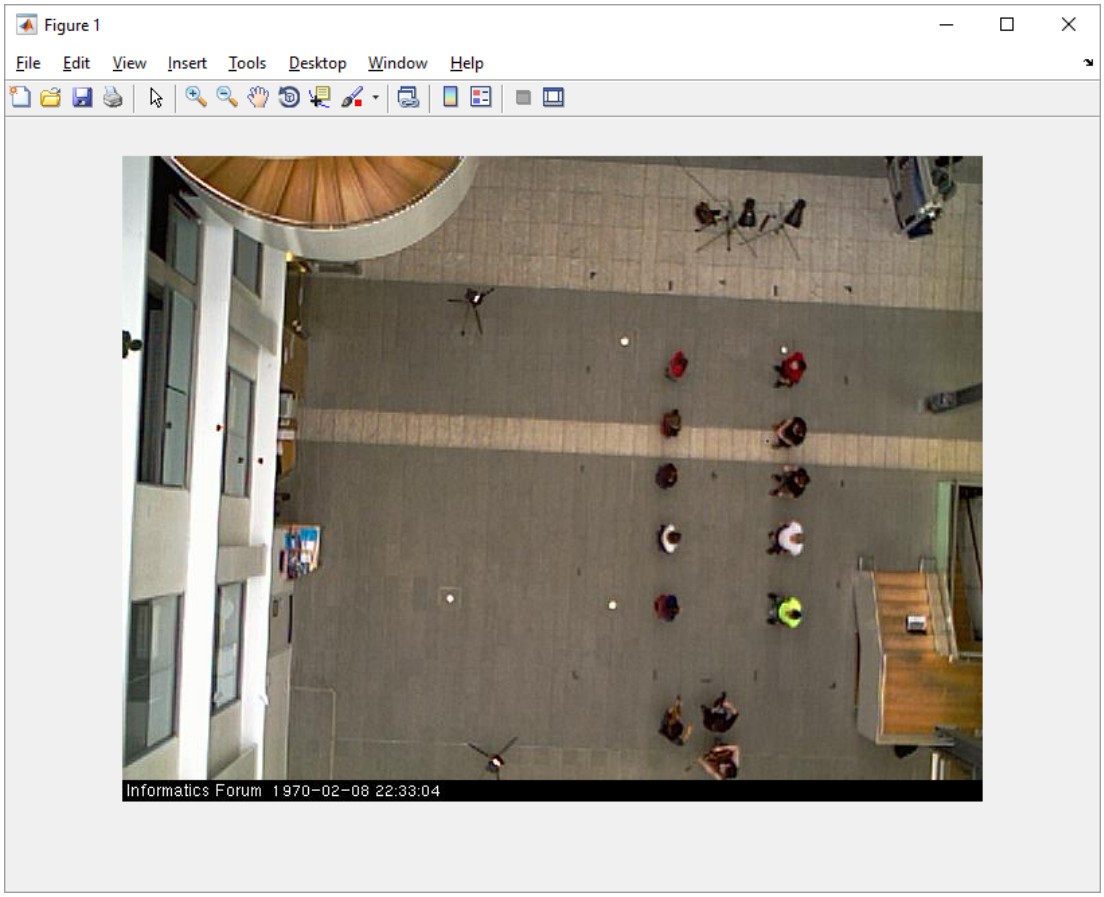


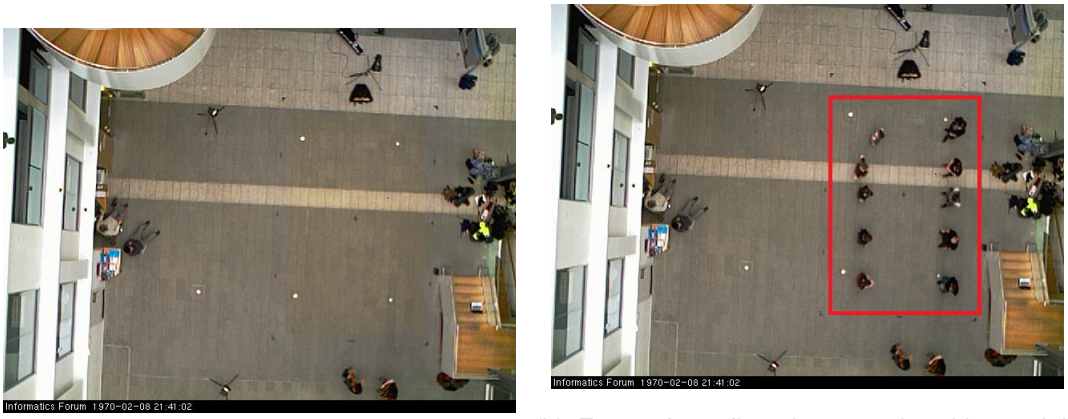
Figure 3.2: The labelling tool used. Clicking on a dancer would record their position.

Chapter 4

Phase I: Object detection and tracking

4.1 Background subtraction

Background subtraction is used for two main reasons. Firstly, the video being processed was taken from a stationary camera. Secondly, the only moving objects within each frame are the ceilidh dancers.



(a) Background frame with no dancers

(b) Frame from first dance style with overlaid border of dance region

Figure 4.1: Example frames from the dance video

In order to extract the dancers seen in figure 4.1b, figure 4.1a (the background image of figure 4.1b) would need to be subtracted from figure 4.1b. The resulting image can be seen in figure 4.2

The background subtraction approach used here is similar to the one used by Ma-

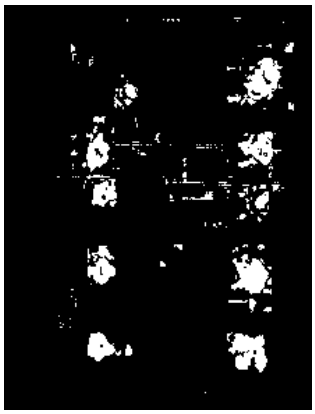


Figure 4.2: A simple background subtraction applied

jecka [12] (background subtraction using chromaticity coordinates)

$$chrom_1 = \frac{Red}{Red + Green + Blue} \tag{4.1}$$

$$chrom_2 = \frac{Green}{Red + Green + Blue} \tag{4.2}$$

$$chrom_3 = \frac{Blue}{Red + Green + Blue} = 1 - chrom_1 - chrom_2 \tag{4.3}$$

Both images (figure 4.1a & 4.1b) are converted to their chromatic coordinates first before the background subtraction process is run. This makes the algorithm robust when there is a difference in lighting between the background image and the current image. This is because the chromatic coordinates are invariant to lighting conditions(luminance).

4.2 Processing foreground

The resulting image (figure 4.2) has some noisy pixels. This problem is solved by further processing the image. Erosion [4] is a common technique used in removing noisy pixels from a binary image. After eroding the image, a dilation [9] function was applied to enlarge the regions of interest as seen in figure 4.3.

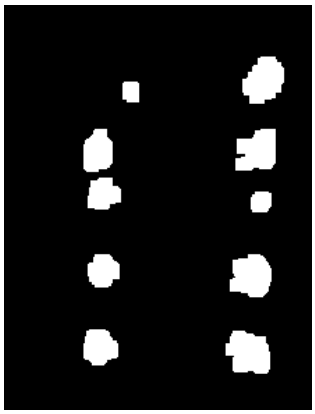


Figure 4.3: After processing figure 4.2 using erosion and dilation

4.3 Tracking

After the background was subtracted and the binary image was cleaned, a tracking algorithm was implemented to track the blobs. The blobs can be describe as a group of pixels joint to getter holding the value of 255 (white) and separated by black pixels (values of 0) i.e. figure 4.3. Using the Matlab function, *bwmorph*, informative properties (centroid, area, bounding box e.t.c.) about each blob was obtained. Some of the properties of the blob was used in the tracking algorithm i.e. the bounding box used to collect the dancer histogram, the centroid is used to estimate the dancers motion.

First of all, a colour histogram for each of the 10 dancers was captured and saved from the first image in the observation sequence. This was used to help distinguish between the dancers based on their colour combination. A Bhattacharyya distance metrics was used for the comparison. The Bhattacharyya distance basically measures the similarity between two probability distributions (continuous or discrete). The Bhattacharyya distance formula for the discrete case takes the form

$$-\ln\Big(\sum_{x\in X}\sqrt{h_1(x)h_2(x)}\Big) \tag{4.4}$$

where $h1(x)$ and $h_2(x)$ are the colour histograms being compared. The smallest absolute value of equation 4.4 is used to identify the dancer of interest; when a dancers' histogram is compared with all the histograms from the detected blob regions. However, the outfit of most of the dancers are quite similar and this causes some confusion

when calculating the Bhattacharyya distance as it could return confusing results i.e. assign to wrong dancer. This problem is complimented with the tracking algorithm.

A condensation tracker [8] was developed for this project. The condensation tracker enabled multiple hypothesis to be encoded. Nine different motion models were encoded:

- up left
- up
- up right
- left
- stop
- right
- down left
- down
- down right

For example, assuming a dancer is in the center of a 3 by 3 grid, then every cell in that grid would have been modelled. The pseudocode is given in Algorithm 1

4.4 Summary

The object detection algorithm used was able to detect the foreground object on every frame. However, the tracking algorithm did not work well and the results were inconclusive. This also meant that the accuracy of the person detection could not be obtained because the centroid values was the distance metric used to calculate the accuracy. As seen in figure **??**, the blobs are joint together to form a big blob which meant that the person detection algorithm would classify this as a false detection (centroid position is away from the dancers real centroid position). The main problems that occurred were:

- Sometimes the blobs of 2 or more dancers merged together in often in dance 1 and frequently in dance 2. A fix was attempted by increasing the number of time

Algorithm 1 Condensation tracker

1: **procedure** BEGIN
2: $t \leftarrow 1$
3: *hypos* $\leftarrow$ *9 motion models*
4: *probs* $\leftarrow$ *probability of each hypotheses*
5: *loop N times to generate hypotheses*$_t$:
6: $h_{t-1} \leftarrow random\_hypothesis(hypos_{t-1})$
7: $p_{t-1} \leftarrow probs(hypos(h_{t-1}))$
8: $s_{t-1} \leftarrow new\_state\_vector(distribution(h_{t-1}))$
9: $s_t \leftarrow new\_state\_vector(current\_observation(s_{t-1}))$
10: $eval\ probability(current\_observation|s_t)$
11: $p_t \leftarrow probability(s_t|current\_observation)$
12: *update hypos*

the image was eroded, but this caused the problems of not seeing other dancers on the other frames. After the dancers have separated the colour histogram was used to try and identify the dancers but not all the dancers had unique colours.

- The temporal spacing between each frame was non-linear. This put the tracker off a lot because if the next frame had a temporal spacing of 3x the norm then the tracker would be lost or would attach itself to a dancer following in the path of the original dancer (especially if they have the same clothing colours). If the temporal spacing between each frame was known then it could have been included in the tracking algorithm.

Due to the time limitation on this project, no other approaches were looked into.

(a) Problem frame from dance 1 (b) Problem frame from dance 2

Figure 4.4: Example of problem blobs

Chapter 5

Phase II: Learning activity features

5.1 Feature functions

The feature function plays a critical role in the CRF. They tell the CRF what to look out for when trying to decode what state the system currently is in. The feature function can be presented in the form

$$f_i(z_{t-1}, z_t, x_{1:T}, t) \tag{5.1}$$

where z_t is the current state and z_{t-1} is the previous state, $x_{1:T}$ is the whole observation sequence and t is the index of the current observation sequence. Each of the feature function contains enough data to detect a feature. For example, a binary *standing* feature function can be encoded

$$f_{standing}(z_{t-1}, z_t, x_{1:T}, t) = \begin{cases} 1, & \text{if } ||x_t - x_{t-1}|| < \tau \text{ and } ||x_t - x_{t+1}|| < \tau \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

where τ is a thresh hold distance (relatively small value).

The ground truth observation data $x_{1:T}$ of each dancers' position within a frame was passed through similar feature functions. The features were made up of spatial and motion features and a single motionless feature, *standing*.

5.2 Motion features

The implemented motion features can be divided into two groups, simple and complicated. The simple features are:

- Walk left (when a dancer is moving towards west)

$$f_{walkleft}(d,t) = \begin{cases} 1, & \text{if } (x_{x,t-1}^{(d)} - x_{x,t}^{(d)}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

- Walk right (when a dancer is moving towards east)

$$f_{walkright}(d,t) = \begin{cases} 1, & \text{if } (x_{x,t}^{(d)} - x_{x,t-1}^{(d)}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

- Walk up (when a dancer is moving towards north)

$$f_{walkup}(d,t) = \begin{cases} 1, & \text{if } (x_{y,t-1}^{(d)} - x_{y,t}^{(d)}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

- Walk down (when a is moving walks towards south)

$$f_{walkdown}(d,t) = \begin{cases} 1, & \text{if } (x_{y,t}^{(d)} - x_{y,t-1}^{(d)}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

where $^{(d)}$ denotes the index of a dancer and $x_{x,y,t}^{(d)}$ denotes the x and y component of the dancers' position at time t.

These features are easily modelled. Only one complicated feature was implemented, *twirling* (when a dancer is moving in a circular motion). This feature is classified as complicated because it is inclusive of at least two simple features in an observation sequence.

5.3 Spatial features

Adding these features improved the accuracy substantially for dancers that had multinomial motional features in an observation sequence i.e. *twirling*. The implemented spatial features are:

- North

$$f_{north}(d,t) = \frac{\max(x_{y,T}^{(d)} - x_{y,t}^{(d)}, 0)}{\max_{u=1:T}(x_{y,T}^{(d)} - x_{y,u}^{(d)})} \tag{5.7}$$

- South

$$f_{south}(d,t) = \frac{\max(x_{y,t}^{(d)} - x_{y,T}^{(d)}, 0)}{\max_{u=1:T}(x_{y,u}^{(d)} - x_{y,T}^{(d)})} \tag{5.8}$$

- West

$$f_{west}(d,t) = \frac{\max(x_{x,T}^{(d)} - x_{x,t}^{(d)}, 0)}{\max_{u=1:T}(x_{x,T}^{(d)} - x_{x,u}^{(d)})} \tag{5.9}$$

- East

$$f_{east}(d,t) = \frac{\max(x_{x,t}^{(d)} - x_{x,T}^{(d)}, 0)}{\max_{u=1:T}(x_{x,u}^{(d)} - x_{x,T}^{(d)})} \tag{5.10}$$

- Horizontal

$$f_{horizontal}(d,t) = \frac{x_{x,t}^{(d)} - \min_{u=1:T}(x_{x,u}^{(d)})}{\max_{u=1:T}(x_{x,u}^{(d)}) - \min_{u=1:T}(x_{x,u}^{(d)})} \tag{5.11}$$

- Vertical

$$f_{Vertical}(d,t) = \frac{x_{y,t}^{(d)} - \min_{u=1:T}(x_{y,u}^{(d)})}{\max_{u=1:T}(x_{y,u}^{(d)}) - \min_{u=1:T}(x_{y,u}^{(d)})} \tag{5.12}$$

The cardinal features(north, south, west and east) are relative to the dancers final position x_T. The features' function returns a normalised value between 0 and 1. For example, if the north feature returns a value of 0.5, this means that dancer d's current position $x_t^{(d)}$ is midway between their final position $x_T^{(d)}$ and the furthest position to the north reached by the dancer

$$\max_{u=1:T}(x_{y,T}^{(d)} - x_{y,u}^{(d)}) \tag{5.13}$$

which is also the normalisation constant used in equation 5.7

The normalised values of the spatial features were multiplied by a constant to increase the detail of the feature. For example, if the vertical feature was multiplied by 3 then one can say a more detailed location of the dancer exists; where the multiplied value lies between of 0 and 1 means $vertical_{up}$, 1 and 2 means $vertical_{middle}$ and 2 and 3 means $vertical_{bottom}$,

5.4 Feature extraction

The dataset is comprised of 2 dances and 10 dancers. Each of the dancers had a CRF assigned to them and trained to learn their features.

5.4.1 Feature selection

It is not all the activity features that is seen in a dance grammar that applies to each dancer. For example, dancer 3 may just be standing while dancer 1 may walk left, twirl and move about throughout the dance. It goes without saying that looking for features that cannot be observed is like looking for noise to add to the data.

Only relevant features were used to train each of the 10 CRFs. These are the features that had the best performance per CRF. The performance metrics used was the test error from the trained CRF in question. Optimizing the CRF this way had a positive effect on the dance CRF (chapter 6)

5.4.2 Training feature CRF

Training a CRF to learn each dancers' features helped in smoothing out the noise in the data. The modelled CRF is analogous to the unsupervised CRF autoencoder by Ammar et al. [1] but with supervision; each feature would have a unique labelled state. If a dancer has 5 features then that dancer is said to have 5 states.

The traditional linear-chain CRF model reviewed in chapter 2.2.3 was used to train the feature CRF. The input vector for the CRF was determined by the number of relevant features selected for the dance. For example, if only 4 features are relevant to a dancer, then the vector would have a size of 4. The features were processed to contain binary values. This meant that if a feature could take 3 values then the feature is treated as 3 relative features where only one of the three features can be active at a time. This technique was also adopted when training the dance CRF in chapter 6 because the results obtained from the training the feature CRF were real values to represent each state (extracted feature).

Chapter 6

Phase III: Learning the dance

Learning the activity features does not only act as a supervised CRF autoencoder but also as a dimensionality reducer.

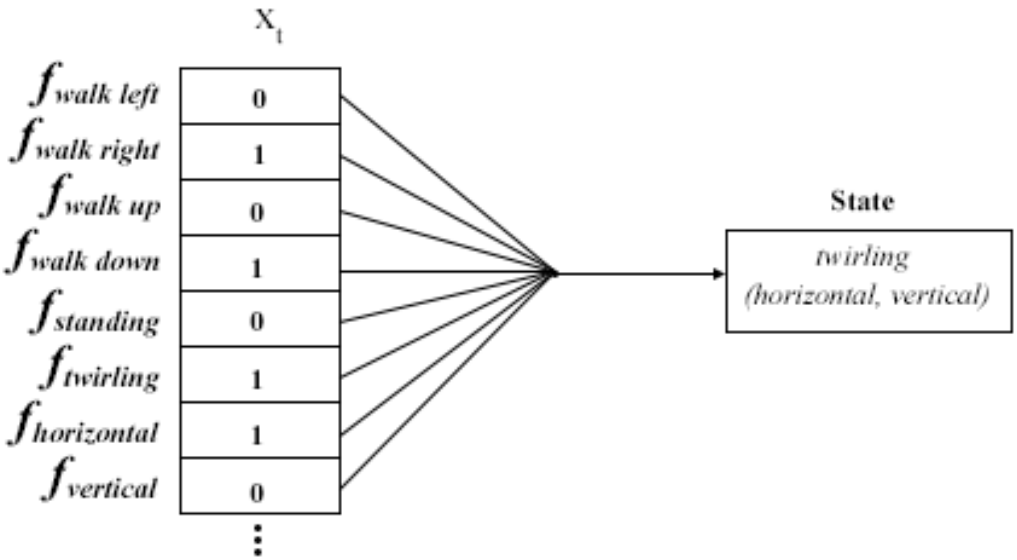


Figure 6.1: Dimensionality and noise reduction from each dancer.

In the scenario illustrated by figure 6.1, 3 motion features are active (*walk right, walk down* and *twirling*) and 1 spatial feature is active (*horizontal*). However, the decoded state only picks 1 motion feature (*twirling*) an all the 2 spatial features(*vertical* and *horizontal*)

First of all, it is not possible for a person to walk in the same but opposite direction at the same time and space i.e up and down, left and right. On the other hand it is

possible to move diagonally, hence the 2 active motion features detected. An act of *twirling* requires a minimum of 2 motion features at a time (most of the time). In this case, if a *twirling* feature has been detected, it would be trained to take precedence over the other active motion features.

Secondly, it is possible for *twirling* or any other motion feature to occur at different locations within the observed frame. For this reason, the spatial values are always encoded in the final state.

6.1 Training

The dance CRF is trained using the 10 input vectors representing the 10 dancers. The size of each vector is determined by the number of decoded features observed for that dancer.

The traditional linear-chain CRF was slightly modified in order to properly train the dance CRF. The new equation for the forward and backward vector from equations 2.14 and 2.16 was changed to

$$\alpha_{i>0}(x) = \alpha_{i-1}(x)(M_i(x) - 1) \tag{6.1}$$

$$\beta_{i<m+1}(x)^{\mathrm{T}} = (M_{i+1}(x) - 1)\beta_{i+1}(x) \tag{6.2}$$

The edge marginal probability from equation 2.19 was also changed to

$$p(z'_{t-1}, z'_t | x^{(n)}_{1:T}) = \frac{\alpha_{i-1}(z'_{t-1}|x^{(n)}_{1:T})(M_i(z'_{t-1}, z'_t|x^{(n)}_{1:T}) - 1)\beta_i(z'_t|x^{(n)}_{1:T})}{Z(x^{(n)}_{1:T})} \tag{6.3}$$

The main reason for subtracting the value of 1 from $M(x)$ (the exponential of the state transition matrix) is to make sure that the 0 probabilities modelled in the state transition matrix is reflected in the CRF model as absolutely 0 because $exp(0) = 1$. This makes sure that edges only exist between the states modelled in the transition matrix $\Lambda_i(z_{t-1}, z_t|x_{1:T})$ from equation 2.17

This modification was required in order to encode the prior knowledge that each dance is strictly labelled to have an ordered state sequence as discussed in chapter

3. This modification also provided improved accuracy when compared against the tradition linear-chain CRF by 8.79% for dance 1 and 4.78% for dance 2 as seen in tables 7.1 & 7.2 and tables 7.3 & 7.4 (using best result at *t +/- 5*)

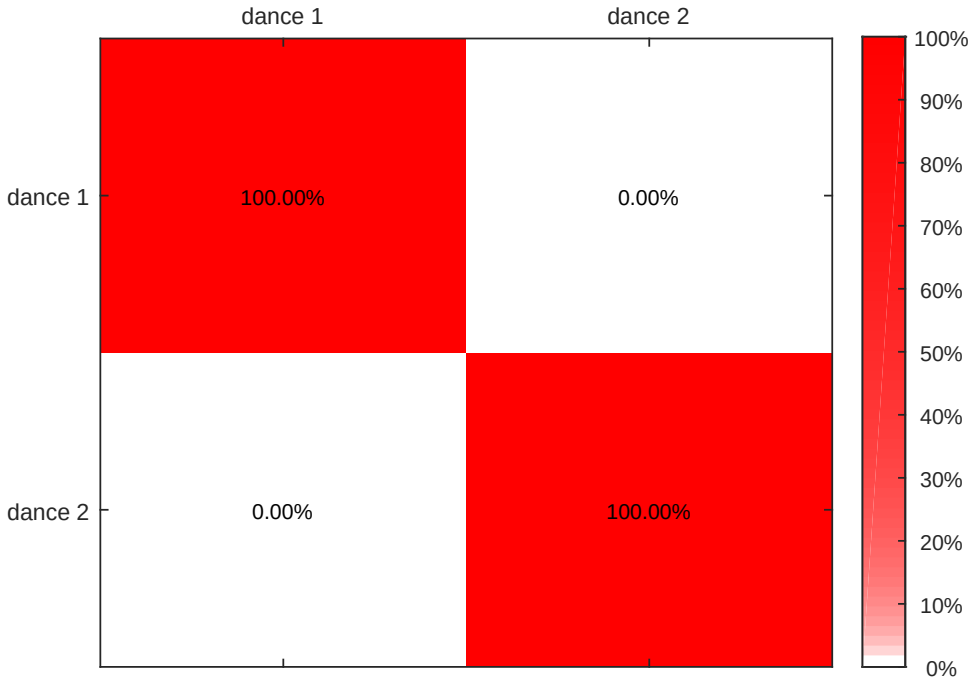


Figure 6.2: Confusion matrix for classifying the two dances

Chapter 7

Experiment and Results

7.1 Experiments

The approach discussed in this paper was carried out on two different ceilidh dances. The first dance was trained using 9 sets of training data (videos) and the second dance was trained using 7 sets of training data (videos). The first dance was also labelled with 15 states while the second dance was labelled with 44 states. The additional test data used to validate the models was the ground truth data with added Gaussian noise ($\sigma = 5$).

The overall performance from the first dance was better than that of the second dance. This could be because the number of possible states in the second dance were more than that of first dance.

The CRF Matlab toolbox that was used for the experiments was developed by Schmidt and Swersky[18].

7.2 Results

Both dances were trained using the k-fold cross validation approach (9-fold for the first dance and 7-fold for the second dance) . The accuracy results are presented in tables 7.1, 7.2, 7.3 , 7.4 7.5 and 7.6 where **t +/- 3** indicates a 3 frame tolerance. For example, if the current state is 6 and the predicted state is 7 but within the next 3 frames, the current state is 7 then this is considered a correct classification.

Fold-*n*	**t +/- 0**	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
1	0.7923	0.8357	0.8599	0.8744	0.8841	0.8937
2	0.7029	0.7543	0.7771	0.7886	0.8000	0.8057
3	0.6892	0.7432	0.7838	0.8108	0.8311	0.8446
4	0.8966	0.9557	0.9852	1.0000	1.0000	1.0000
5	0.8315	0.8913	0.9185	0.9348	0.9511	0.9620
6	0.7047	0.7409	0.7617	0.7824	0.8031	0.8238
7	0.7650	0.8197	0.8470	0.8689	0.8852	0.8907
8	0.7734	0.8177	0.8473	0.8621	0.8719	0.8768
9	0.6879	0.7261	0.7580	0.7834	0.8025	0.8280
mean	0.7604	0.8094	0.8376	0.8561	0.8699	0.8806
s.t.d.	0.0721	0.0775	0.0772	0.0748	0.0701	0.0652

Table 7.1: Results from a normal Linear-chain CRF model on dance 1

Fold-*n*	**t +/- 0**	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
1	0.9517	0.9903	1.0000	1.0000	1.0000	1.0000
2	0.9371	0.9714	0.9886	0.9943	1.0000	1.0000
3	0.9122	0.9595	0.9932	1.0000	1.0000	1.0000
4	0.9064	0.9655	0.9852	0.9951	1.0000	1.0000
5	0.9239	0.9674	0.9891	1.0000	1.0000	1.0000
6	0.8497	0.8912	0.9171	0.9223	0.9275	0.9326
7	0.8852	0.9399	0.9672	0.9836	1.0000	1.0000
8	0.9261	0.9704	0.9951	1.0000	1.0000	1.0000
9	0.6306	0.6752	0.7070	0.7325	0.7580	0.7834
mean	0.8803	0.9256	0.9492	0.9586	0.9650	0.9685
s.t.d.	0.0983	0.0980	0.0943	0.0884	0.0813	0.0729

Table 7.2: Results from the modified linear-chain CRF model on dance 1

Fold-*n*	**t +/- 0**	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
1	0.7105	0.7703	0.8110	0.8445	0.8589	0.8660
2	0.7443	0.8128	0.8356	0.8539	0.8676	0.8744
3	0.8083	0.8762	0.9102	0.9248	0.9320	0.9369
4	0.7400	0.7916	0.8220	0.8361	0.8431	0.8454
5	0.8527	0.9179	0.9420	0.9565	0.9638	0.9686
6	0.7143	0.7734	0.8079	0.8251	0.8374	0.8498
7	0.8082	0.8772	0.9079	0.9258	0.9284	0.9335
mean	0.7683	0.8313	0.8624	0.8810	0.8902	0.8964
s.t.d.	0.0547	0.0586	0.0558	0.0530	0.0502	0.0490

Table 7.3: Results from a normal linear-chain CRF model on dance 2

Fold-*n*	**t +/- 0**	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
1	0.8349	0.8923	0.9234	0.9426	0.9498	0.9545
2	0.8813	0.9361	0.9543	0.9658	0.9726	0.9795
3	0.7670	0.8374	0.8592	0.8689	0.8738	0.8786
4	0.7635	0.8220	0.8501	0.8712	0.8759	0.8782
5	0.8696	0.9469	0.9734	0.9879	0.9928	0.9952
6	0.7956	0.8695	0.8941	0.9113	0.9212	0.9261
7	0.8568	0.9361	0.9744	0.9872	0.9923	0.9974
mean	0.8241	0.8915	0.9184	0.9336	0.9398	0.9442
s.t.d.	0.0488	0.0505	0.0520	0.0508	0.0509	0.0513

Table 7.4: Results from the modified linear-chain CRF model on dance 2

t +/- 0	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
0.9298	0.9716	0.9861	0.9927	0.9964	0.9982

Table 7.5: Modified linear-chain CRF model for dance 1 (best model): Results from using the ground truth data (9 videos) for dance 1 with Gaussian noise $(\sigma = 5)$ to test

t +/- 0	**t +/- 1**	**t +/- 2**	**t +/- 3**	**t +/- 4**	**t +/- 5**
0.9298	0.9763	0.9855	0.9911	0.9942	0.9959

Table 7.6: Modified linear-chain CRF model for dance 2 (best model): Results from using the ground truth data (7 videos) for dance 2 with Gaussian noise $(\sigma = 5)$ to test

7.3 Summary

The proposed CRF model was able to achieve an accuracy 96.85% for dance 1 and 94.42% for dance 2 where the frame tolerance is set to +/- 5 frames. There was no confusion in the dance classification matrix (figure 6.2). A couple of classification plots can be seen in Appendix B. The confusion matrices between each dance role and the dancer can also be found in Appendix C

Chapter 8

Conclusion

This project set out to increase the capabilities of video analysis using a ceilidh dance as a foot stool. The outcome of this project has shown that a linear-chain CRF can be used to recognise a ceilidh dance. High accuracies of 96.85% for dance 1 and 94.42% for dance 2 were obtained. The proposed linear-chain CRF model outperformed the traditional linear-chain model by 8.79% for dance 1 and 4.78% for dance 2.

Though the goals of the project were accomplished a vital issue still remains. Further research is required on developing a suitable tracking algorithm so that the model can be tested using real world data.

For further development, this model can be compared with a model that does not have the intermediary step of learning features before learning the dance. This way, the model could be evaluated against a model that jointly trains the 10 dancers at a go.

Real world problems that require the recognition of spatio-temporal tasks can be experimented on. i.e. hand-gesture recognition, understanding animal behaviour, disease life-cycle

Appendix A

State Transition Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.9524	0.0476	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0.7778	0.2222	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0.9633	0.0286	0.0082	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.7586	0.2414	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.8571	0.1429	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0.9011	0.0989	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0.7273	0.2727	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0.9338	0.0662	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0.9189	0.0811	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0.8657	0.1343	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0.9557	0.0443	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0.9182	0.0818	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0.8269	0.1731	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9291	0.0709
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table A.1: State transition matrix for dance 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
1	0.9320	0.0680	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0.6818	0.3182	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0.7200	0.2800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.6957	0.3043	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.7083	0.2917	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0.7200	0.2800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0.6818	0.3182	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0.8333	0.1667	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0.7407	0.2593	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0.8409	0.1591	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0.8814	0.1186	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0.8939	0.1061	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0.8679	0.1321	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8986	0.1014	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7813	0.2188	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9698	0.0302	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9125	0.0875	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9195	0.0805	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9205	0.0795	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8955	0.1045	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9397	0.0603	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9514	0.0486	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7879	0.2121	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7742	0.2258	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6500	0.3500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5882	0.4118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7500	0.2500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8704	0.1296	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7083	0.2917	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6667	0.3333	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8542	0.1458	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8600	0.1400	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8889	0.1111	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8833	0.1167	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9136	0.0864	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.7813	0.2188	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9693	0.0307	0	0	0	0	0	0
38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9222	0.0778	0	0	0	0	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9079	0.0921	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9091	0.0909	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9054	0.0946	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9222	0.0778	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9320	0.0680
44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table A.2: State transition matrix for dance 2

Appendix B

Classification Plot

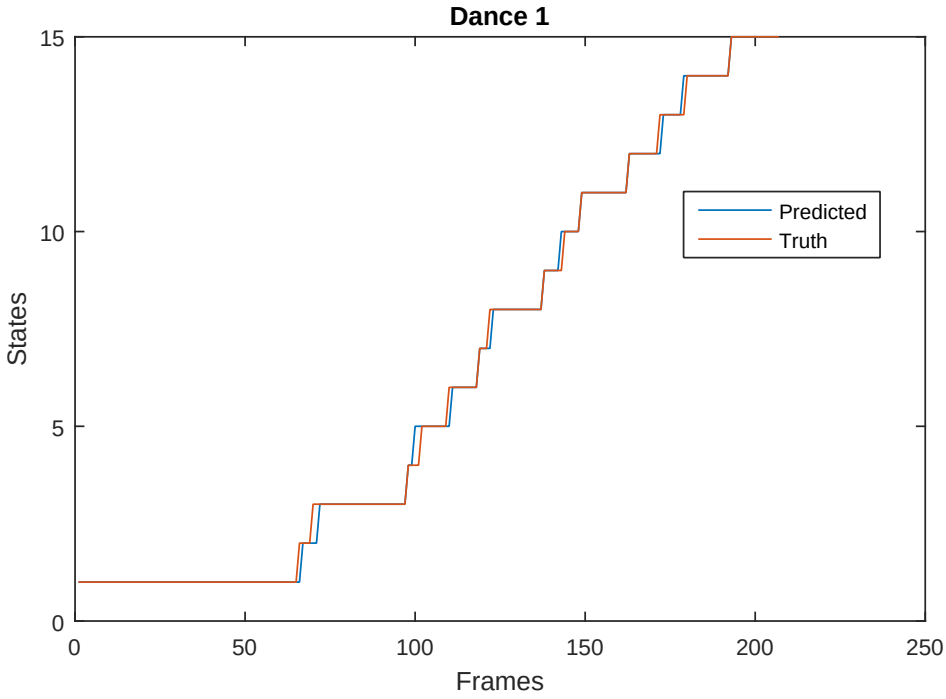


Figure B.1: Classification plot on the best model for dance 1.

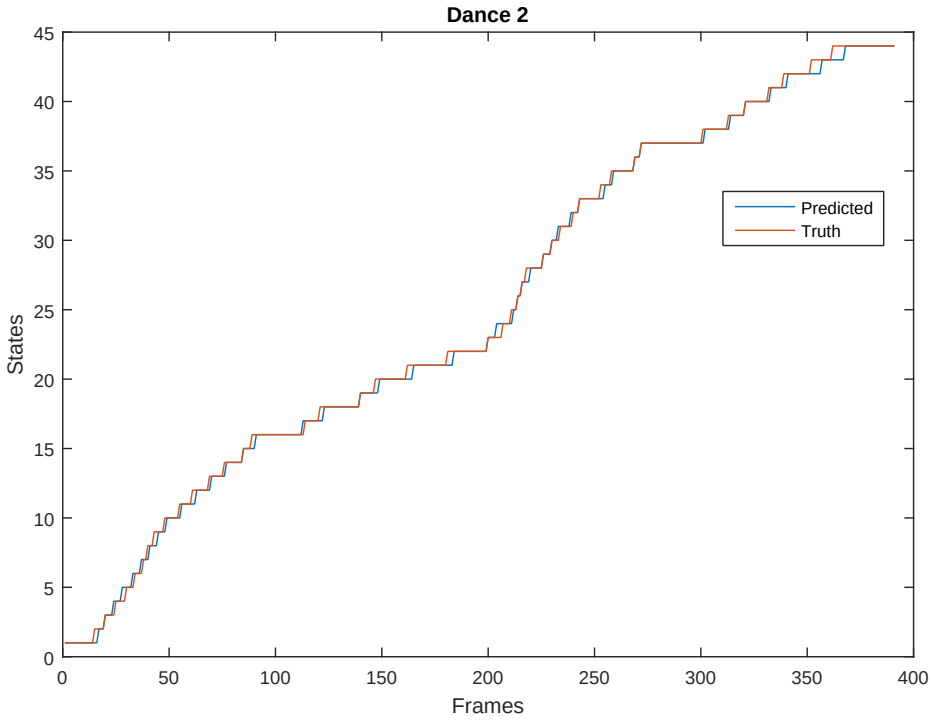


Figure B.2: Classification plot on the best model for dance 2.

Appendix C

Confusion Matrices

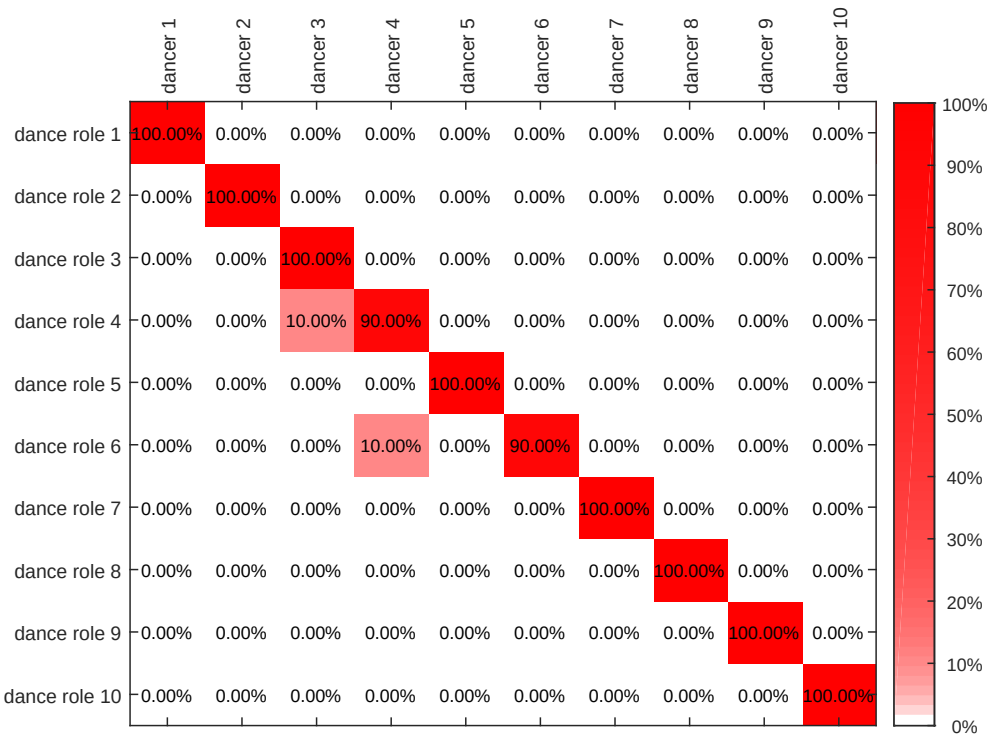


Figure C.1: Confusion matrix for dance 1: Using ground truth data (9 videos) for dance 1 as test data with Gaussian noise $(\sigma = 5)$

Figure C.2: Confusion matrix for dance 2: Using ground truth data (7 videos) for dance 2 as test data with Gaussian noise $(\sigma = 5)$

Figure C.3: Confusion matrix for dance 1: Using ground truth data (9 videos) for dance 1 as test data

Figure C.4: Confusion matrix for dance 2: Using ground truth data (7 videos) for dance 2 as test data

Bibliography

[1] Ammar, W., Dyer, C., and Smith, N. A. (2014). Conditional random field autoencoders for unsupervised structured prediction. *CoRR*, abs/1411.1147.

[2] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[3] Gayathri, S., Priyadharshini, A. S., and Bhuvaneswari, P. T. V. (2014). Multivariate linear regression based activity recognition and classification. In *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–6.

[4] Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2003). *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[5] Gough, J. (2001). *Hidden Markov models and their application to genome analysis in the context of protein structure*. PhD thesis, Cambridge University.

[6] Hossain, H. M. S., Roy, N., and Khan, M. A. A. H. (2016). Active learning enabled activity recognition. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9.

[7] Ilic, V. M., Mancev, D. I., Todorovic, B., and Stankovic, M. S. (2010). Gradient computation in linear-chain conditional random fields using the entropy message passing algorithm. *CoRR*, abs/1011.1478.

[8] Isard, M. and Blake, A. (1998). Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28.

[9] JI, L., Piper, J., and Tang, J.-Y. (1989). Erosion and dilation of binary images by arbitrary structuring elements using interval coding. *Pattern Recognition Letters*, 9(3):201 – 209.

[10] Kothiya, S. V. and Mistree, K. B. (2015). A review on real time object tracking in video sequences. In *Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on*, pages 1–4.

[11] Lee, K., Lee, C., Kim, S. A., and Kim, Y. H. (2012). Fast object detection based on color histograms and local binary patterns. In *TENCON 2012 - 2012 IEEE Region 10 Conference*, pages 1–4.

[12] Majecka, B. (2009). Statistical models of pedestrian behaviour in the forum. Master's thesis, University of Edinburgh.

[13] Morioka, K., Ando, N., Lee, J.-H., and Hashimoto, H. (2003). Robust tracking of multiple objects using color histogram in intelligent environment. In *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, volume 1, pages 533–538 vol.1.

[14] Nongmeikapam, K., Shangkhunem, T., Chanu, N. M., Singh, L. N., Salam, B., and Bandyopadhyay, S. (2011). Crf based name entity recognition (ner) in manipuri: A highly agglutinative indian language. In *Emerging Trends and Applications in Computer Science (NCETACS), 2011 2nd National Conference on*, pages 1–6.

[15] Prasad, G., Fousiya, K. K., Kumar, M. A., and Soman, K. P. (2015). Named entity recognition for malayalam language: A crf based approach. In *Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference on*, pages 16–19.

[16] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16.

[17] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[18] Schmidt, M. and Swersky, K. (2008). crfchain: Matlab code for chain-structured conditional random fields with categorical features. http://www.cs.ubc.ca/ schmidtm/software/crfchain.html.

[19] Setitra, I. and Larabi, S. (2014). Background subtraction algorithms with post-processing: A review. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 2436–2441.

[20] Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373.

[21] Yalcin, H. (2016). Human activity recognition using deep belief networks. In *2016 24th Signal Processing and Communication Application Conference (SIU)*, pages 1649–1652.

[22] Yang, X. and Liu, J. (2014). Deep belief network based crf for spoken language understanding. In *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, pages 49–53.

[23] Zhao, H. and Kit, C. (2007). Scaling conditional random field with application to chinese word segmentation. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 5, pages 95–99.