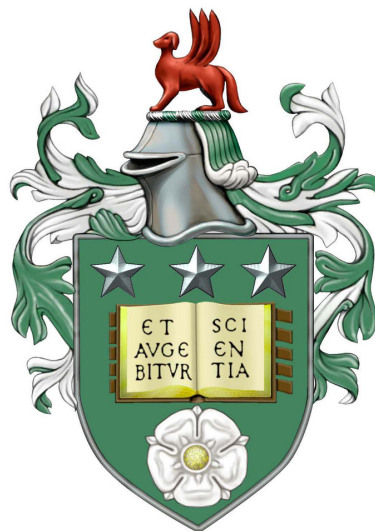


**Hierarchical Modelling and Recognition of Activities of
Daily Living**

Jawad Tayyub

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy



The University of Leeds

School of Computing

May 2018

The candidate confirms that the work submitted is his/her own, except where work which has formed part of a jointly authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Some parts of the work presented in Chapters 3, 4 and 5 have been published in the following articles:

Tayyub, J., Hawasly, M., Hogg, D. C. and Cohn, A. G.

Learning Hierarchical Models of Complex Daily Activities from Annotated Videos. In *2018 IEEE Winter Conference on Applications of Computer Vision, (WACV)*, 2018.

Tayyub, J., Hawasly, M., Hogg, D. C. and Cohn, A. G.

CLAD: A Complex and Long Activities Dataset with Rich Crowdsourced Annotations. In *arXiv:1709.03456, (arXiv)*, 2017.

Tayyub, J., Tavanai, A., Gatsoulis, Y., Hogg, D. C. and Cohn, A. G.

Qualitative and Quantitative Spatio-Temporal Relations in Daily Living Activity Recognition. In *Asian Conference on Computer Vision, (ACCV)*, 2014.

The above publications are primarily the work of the candidate.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

©2015 The University of Leeds and Your name

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors Professor Anthony Cohn and Professor David Hogg for their utmost support, constant encouragement, motivation and immense knowledge. I could not have imagined anyone better to offer me with such invaluable guidance throughout my PhD.

I would also like extend a special thank you to Dr. Majd Al-Hawasly for the brainstorming sessions, in-depth discussions and novel ideas.

I would also like to thank my colleagues Dr. Muhanad Al-Omari, Dr. Paul Duckworth and Dr. Aryana Tavanai.

I would also like to thank my father for his uncanny support, my mother for her unlimited love and lastly my beloved sister.

Finally, I would like to thank Giulia Pederzini for her undying affection and support throughout this journey.

Abstract

Activity recognition is becoming an increasingly important task in artificial intelligence. Successful activity recognition systems must be able to model and recognise activities ranging from simple short activities spanning a few seconds to complex longer activities spanning minutes or hours. We define activities as a set of qualitatively interesting interactions between people, objects and the environment. Accurate activity recognition is a desirable task in many scenarios such as surveillance, smart environments, robotic vision etc. In the domain of robotic vision specifically, there is now an increasing interest in autonomous robots that are able to operate without human intervention for long periods of time. The goal of this research is to build activity recognition approaches for such systems that are able to model and recognise simple short activities as well as complex longer activities arising from long-term autonomous operation of intelligent systems. The research makes the following key contributions:

1. We present a qualitative and quantitative representation to model simple activities as observed by autonomous systems.
2. We present a hierarchical framework to efficiently model complex activities that comprise of many sub-activities at varying levels of granularity.

Simple activities are modelled using a discriminative model where a combined feature space, consisting of qualitative and quantitative spatio-temporal features, is generated in order to encode various aspects of the activity. Qualitative features are computed using qualitative spatio-temporal relations between human subjects and objects in order to abstractly represent the simple activity. Unlike current state-of-the-art approaches, our approach uses significantly fewer assumptions and does not require any knowledge about object types, their affordances, or the constituent activities of an activity. The optimal and most discriminating features are then extracted, using an entropy-based feature selection process, to best represent the training data.

A novel approach for building models of complex long-term activities is presented as well. The proposed approach builds a hierarchical activity model from mark-up of activities acquired from multiple annotators in a video corpus. Multiple human annotators identify activities at different levels of conceptual granularity. Our method automatically infers a ‘part-of’ hierarchical activity model from this data using semantic similarity of textual annotations and temporal consistency. We then consolidate hierarchical structures learned from different training videos into a generalised hierarchical model represented as an extended grammar describing the overall activity. We then describe an inference mechanism to interpret new instances of activities. Simple short activity classes are first recognised using our previously learned generalised model. Given a test video, simple activities are detected as a stream of temporally complex low-level actions. We then use the learned extended grammar to infer the higher-level activities as a hierarchy over the low-level action input stream.

We make use of three publicly available datasets to validate our two approaches of modelling simple to complex activities. These datasets have been annotated by multiple annotators through crowd-sourcing and in-house annotations. They consist of daily activity videos such as ‘cleaning microwave’, ‘having lunch in a restaurant’, ‘working in an office’ etc. The activities in these datasets have all been marked up at multiple levels of abstraction by multiple annotators, however no information on the ‘part-of’ relationship between activities is provided. The complexity of the videos and their annotations allows us to demonstrate the effectiveness of the proposed methods.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Terminology	3
1.3	Goals and Challenges	4
1.4	Thesis Structure	7
2	Related work	11
2.1	Previous Surveys	11
2.2	Simple Activity Recognition	13
2.2.1	Pixel-based Representations	14
2.2.2	Object-based Representation	16
2.2.3	Qualitative Representation	17
2.3	Complex Activity Recognition	24
2.3.1	Graphical Approaches	25
2.3.2	Grammar Approaches	26
2.4	Summary	29
3	Qualitative and Quantitative Modelling of Simple Activities	31
3.1	Introduction	31
3.2	Qualitative Representation	33
3.2.1	Motivation	34
3.2.2	Representation	37

3.2.3	Qualitative Spatial Features using Region Connection Calculus	43
3.2.4	Qualitative Spatial Features using Qualitative Trajectory Calculus	49
3.2.5	Qualitative Temporal Features	50
3.2.6	Smoothing Fast Changing Relationships	54
3.3	Quantitative Representation	55
3.3.1	Motivating Example	56
3.3.2	Representation	57
3.4	Objects Representation	62
3.5	Feature Selection	62
3.6	Concluding Remarks	65
4	Hierarchical Modelling of Complex Activities	67
4.1	Introduction	67
4.2	Annotation Collection Process	70
4.2.1	In-house Annotations	71
4.2.2	Crowdsourced Annotations	71
4.3	Clustering Intervals Input Data	73
4.3.1	Motivation	73
4.3.2	Extended Example	74
4.3.3	Agglomerative Clustering of Annotations	75
4.4	Learning Hierarchical Structure of Activities	82
4.4.1	Challenges	82
4.4.2	Assumptions	84
4.4.3	Cost-based Optimum Activity Hierarchy Inference	85
4.4.4	Encoding Temporal Ordering of Sub-activities	98
4.5	Abstracting a Generalised Activity Model	102
4.5.1	Grammar Induction in Natural Language Processing	102
4.5.2	Building General Activity Models using Grammar Induction	105
4.6	Recognition	110
4.6.1	Low-level Action Recognition	111

4.6.2	Extended Earley Parser for Activity Hierarchy Recognition	113
4.7	Conclusion	120
5	Experimental Procedure	123
5.1	Introduction	123
5.2	Evaluation Measures and Ground-truth	125
5.2.1	Accuracy	125
5.2.2	Recall	125
5.2.3	Precision	125
5.2.4	Normalised Mutual Information	126
5.2.5	V-Measure	126
5.2.6	Correctly Parsed Terminals (CPT)	127
5.2.7	Qualitative Analysis	127
5.3	Datasets	127
5.3.1	Cornell Activity Dataset-120 (CAD-120)	128
5.3.2	Leeds Activity Dataset (LAD)	131
5.3.3	Complex and Long Activity Dataset (CLAD)	133
5.3.4	Summary	136
5.4	Evaluation of Qualitative and Quantitative Representation	137
5.4.1	Experiment 1: Evaluation of Simple Activity Recognition	138
5.4.2	Experiment 2: Evaluation of Classification of Mirror Activities	148
5.4.3	Discussion	151
5.5	Evaluation of Complex Activity Recognition	154
5.5.1	Experiment 3: Evaluation of Clustering Annotations	155
5.5.2	Experiment 4: Evaluation of <i>Parent-child</i> Relationship Inference	158
5.5.3	Experiment 5: Evaluation of Recognition of Complex Activity Hierarchies	166
5.5.4	Conclusion	171
6	Discussion and Conclusion	173
6.1	Summary	174
6.2	Contributions	176

6.3 Future Work	177
6.4 Final Remarks	181

List of Figures

1.1	An example of a hierarchy of high-level activities inferred from detections of low-level activities. High-level activities are shown using red boxes whereas low-level activities are represented by green boxes. Recognising such a hierarchical structure of activities over a test video is our key goal.	5
1.2	A flowchart showing different components of the presented frameworks in this thesis. Also, the interaction between the different frameworks is shown. Pink boxes represent training data. Yellow boxes represent testing videos. Green boxes represent recognised results.	8
2.1	An overview of the approaches of simple to complex activity recognition. Image taken from [Turaga et al., 2008]	12
2.2	Example of a pixel-based representation to generate templates of activities. Image taken from [Bobick and Davis, 2001]	15
2.3	A conceptual neighbourhood graph showing the relationships of the RCC-8 calculus. Any one of the eight relationships shown can represent the spatial arrangement of any two objects, a and b, at any one time. For example, ‘DC’ denotes a <i>disconnect</i> relationship, ‘PO’ denotes a <i>partially overlapping</i> relationship and so on. For a complete list of relations, refer to [Randell et al., 1992].	19

2.4	Conceptual neighbourhood graph of the basic qualitative trajectory calculus (QTCb) shown. The solid circles represent static objects, hollow circles represent moving object with the surrounding semi-circles indicating their range of motion. The corresponding QTCb relation is shown atop each arrangement of objects [Van de Weghe et al., 2006].	21
2.5	A subset of Allen’s temporal relations shown in the left column. The right column shows the INDU calculus, which can be seen as an extension of the Allen’s relations with the addition of relative durational knowledge.	23
3.1	A flowchart of the qualitative and quantitative spatio-temporal representations (QQSTR) activity recognition system.	32
3.2	Simple example of a ‘turning off the alarm’ activity shown in terms of positional observations of a hand and a clock entity at each frame. The corresponding RCC-3 relational string between the entities is also shown. A simple run-length encoding of the relational string also presented. The relations in the string present qualitative relations between the entities where ‘D’ refers to discrete and ‘PO’ refers to partially overlapping.	36
3.3	Conceptual neighbourhood graphs of RCC-5 and the simplified RCC-3 qualitative spatial relationships.	38
3.4	This illustration shows all possible states of motion between two objects in our 2-D representation. Atop each state, the corresponding SQTC relation is shown. The arrows demonstrate the transitions allowed between relations. Solid circles represent static objects whereas hollow circle represent moving objects. The semi-circle represents the range of motion and the size of the semi-circles represents velocity.	41
3.5	Example of an ‘unstacking boxes’ and ‘stacking boxes’ activity illustrating the motivation for using the SQTC calculus to discriminate amongst them.	43

3.6 An extended example of a ‘receiving a call’ activity represented at each time step. The corresponding relational matrix \mathbf{M} generated for each entity pair for every frame is shown. The two compression encoding mechanisms are presented. Global encoding represents a holistic picture of activity compared to local compression. However, local compression achieves a higher degree of compression. . . . 44

3.7 Feature space representing a vocabulary of codewords represented as a histogram. Note that codewords are represented by the set of relational changes as bins in this histogram. That is to say, each bin denotes a single codeword which may be ‘D’, ‘D,PO,D’ or ‘P,P,P,P’ and so on. Since there are N_E entity pairs within an activity video, the bins set is repeated N_E times to capture the relational changes for each entity pair. For example for entities 1 and 2, the corresponding bin set is denoted by $e_{1,2}$ in this figure. 48

3.8 Three learned clusters shown here represent the three qualitative relations which describe temporal knowledge. Note that these closely resemble the three extensions offered by the INDU calculus over symbols of the Allen’s interval algebra. The three clusters are labelled as either short, equal or long, reflecting the relative lengths of the two intervals. An example of a pair of intervals classified into each of these three clusters is presented. 52

3.9 The feature space F_3 represented as a histogram. The bins of the histogram correspond to the codewords representing the counts of specific durational relations held between two intervals. The bins are repeated for every entity pair in the activity. Note that another strength of our framework is that it does not utilise exact entity labels, which are often noisy and difficult to recognise for computer vision systems. 54

3.10 Simple example of a ‘turning off the alarm’ activity shown in terms of positional observations of a hand and a clock entity at each frame number. The corresponding Euclidean distance string between entities is also shown. Finally, the underlying distribution of distances represents the motion pattern between the entities. Statistically descriptive features can be extracted to represent the distribution. 57

3.11	The Feature space for F_4 shown as a feature vector. These features capture four moments of a distribution of Euclidean distances amongst each entity pair in the activity to achieve a quantitative representation of the activity.	61
4.1	A flowchart of our hierarchical modelling and recognition of complex activities system presented. The process comprises of learning hierarchical structures of activity derived from human descriptions. Automatic low-level action recognition on a test video, using the QQSTR system, is then performed to infer the remaining hierarchical activity structure over the test video.	69
4.2	A screen shot of the web-interface used to collect crowdsourced annotations of activities.	72
4.3	Agglomerative clustering over an example set of activity annotations for a single video of ‘using the phone’ activity. Annotations of a single video from three annotators are shown on the top. These are concatenated together to produce a saturated activity annotation describing activities at various granularities. An agglomerative clustering, with a custom distance function, is performed and the resulting clustering is shown to cluster all redundant labels while keeping sub-activity structures separated. Each cluster is referred to as an activity node ω	74
4.4	Complete linkage for agglomerative clustering proximity measure illustrated as the maximum distance between elements of two clusters.	81
4.5	Clean and noise-free annotations shown on the left, notice the full subsumption of activity intervals within one another. The right figure shows a real-world example of annotations. Due to subjective temporal boundaries and noise, the red circles show <i>parent-child</i> activity nodes with partial subsumption. This makes the task of inferring the correct <i>parent-child</i> relationships non-trivial.	83

4.6 The example shows matching of two activity nodes using weighted bi-partite graph matching concepts. Inter-label similarity is computed using the off-the-shelf linguistic semantic similarity measure λ^l . The Hungarian algorithm is used to solve the assignment problem between the two node lists, where blue edges represent assigned edges. The red edges represent unassigned edges which are used to penalise the overall score. 88

4.7 Sample interval data shown in a). The corresponding interval graph representation of the data shown in b). The trivially perfect graph in c) represents the inter data in a hierarchical format based on full subsumption criteria. d) The directed acyclic graph is a transitive reduction of the trivially perfect graph. This graph most closely represents the activity hierarchy structure we aim to learn. 92

4.8 Justification of the terms of the cost function are shown in this figure. Notation used to represent overlaps and lengths between any two interval ω_i and ω_j shown in a). The cost function is presented along with sample examples justifying each term. ω_c denotes a child activity interval and ω_{p*} denote all candidate parent activity nodes for that child. The red parent intervals show the correctly chosen interval as the parent for each of the child nodes in the three example arrangements. 96

4.9 This figure represents a sample set of sibling activities for the ‘preparing tea’ activity. The corresponding activity cluster denotes the set of activity nodes with their temporal information. The is represented using an interval graph shown in graphical form (left) and in notation (right). Note that, for simplicity, only one label is shown within each activity node. In real-data, multiple labels comprise an activity node. 99

4.10 This figure presents an example hierarchy tree, and the temporalised hierarchy. Each set of siblings (nodes on the same level) have been converted into interval graphs encoded with Allen’s temporal relations. Note that these hierarchies represent the activity nodes learned from the ‘using the phone’ example in figure 4.3. 101

4.11 Example of input sentence parsed using the language grammar shown on the right. 103

- 4.12 Example of ambiguity in parsing sentences using grammar rules. Parse tree a) represents the correct parsing of the words where ‘with regret’ is correctly identified as a prepositional phrase (PP) relating to a verb (V). Parse tree b) shows that ‘chocolate with regret’ is all a single noun phrase (NP), which is an incorrect classification. 104
- 4.13 Example of the incremental abstraction of temporalised hierarchies shown. Starting at the root node, activity clusters are merged based on their similarity. Concurrently, probability values associated with each cluster are also learned from their frequency of observation. The final model is represented as an extended grammar where the right hand side of each rule is augmented with a relations matrix \mathbf{R} encoding the inter-symbol Allen’s interval relations between the elements of \mathbf{Q} 109
- 4.14 This flowchart represents the extraction of low-level action terminals from the generalised hierarchical model. The green nodes represent the terminals/leaf nodes. The classes of these nodes are learned from extracting data from raw videos. Given a single test video, these actions are classified to generate a temporally complex activity stream. Using the multi-threaded parsing method, the complete activity hierarchy describing the input terminals is recovered. 112
- 4.15 A sample learned activity grammar presented along with a sample recognised low-level action terminal stream. Note that this example omits temporally complex activity grammar and input action stream for simplicity. The parsing output shows the output from a standard Earley parser executed on the input stream. From such a parsing output, multiple solutions with partial parse trees describing any sub-set of the input stream can be retrieved. Finally, the probability of the parse trees, the coverage (number of terminals described by the trees), and number of parse trees are shown for each solution. 114

4.16 This figure shows a temporally complex input stream on the left. Using an activity grammar with the Earley multi-threaded parser, we are able to recover all partial parses over many subsets of the input terminal stream. T_1 describes the orange and purple input actions, T_2 describe the green and blue actions etc. Note that the lowest level nodes, (hollow circles outlined with the same colour as their activity node) correspond to detected actions as seen in the action terminal stream. 117

4.17 This example illustrates the correct selection of partial parse tree over an input action terminal stream. The input stream is shown to be sequential in this example for simplicity, however, the same requirements apply for a temporally complex input stream. The triangles refer to the set of terminals that are described by the parse tree denoted by T . Note that we aim to minimise the number of trees selected therefore the bottom solution is preferred over the top one in a). In b), our aim is to avoid selection of trees which describe the same terminal; therefore the bottom solution is preferred to the top solution here as well. 118

5.1 Sample images of the CAD dataset. Various activity classes presented including ‘Microwaving Food’, ‘Stacking Objects’ etc. Bottom row shows the automatic skeleton tracks and object detections overlayed over the images. Image taken from [Duckworth, 2017] 129

5.2 Sample images extracted from some videos of the LAD dataset. Bottom row shows automatic skeleton tracks and automatic object detections overlaid over the images. 131

5.3 Sample images from videos of the three main classes of activities: ‘having breakfast’, ‘lunch in a restaurant’ and ‘working in an office’ presented. Each column shows activities from each one of the top-level activities. Bottom row shows automatic skeleton tracks overlaid on the sample images. 134

5.4 Framework of our QQSTR system for simple activity recognition. All three datasets are used to test the full system end-to-end. 138

5.5	Confusion Matrices showing classification results from high-level activities and sub-level activities using ground-truth object detections and automatic object detections, in the CAD dataset.	143
5.6	Confusion matrices presenting classification performance of activities in the LAD dataset.	146
5.7	Confusion matrices presenting classification performance of discriminating mirror activities in the CAD dataset. The red rectangle highlights the confusion and shows improvement across columns.	149
5.8	Confusion matrices presenting classification performance of discriminating mirror activities in the LAD dataset. The red rectangle highlights the confusion and shows improvement across the rows.	150
5.9	A bar chart showing accuracy values achieved on sub-level activity recognition across all datasets with different combinations of feature spaces. F_1 refers to the RCC-3 QSRs feature space, F_2 refers to the SQTC QSR feature space, F_3 refers to the learned temporal QSRs feature space and F_4 refers to the quantitative feature space. The error bars represent the standard deviation for each accuracy value.	152
5.10	A bar chart presenting best accuracies achieved with and without use of automatic entropy-based feature selection. The error bars represent the standard deviation for each accuracy value.	153
5.11	A flowchart of our complex activity recognition approach. The CLAD dataset is used to evaluate all parts of the system whereas the CAD and LAD datasets are used to evaluate all parts except the semantic clustering.	154
5.12	Sample output of clusters formed through our clustering mechanism. Note that linguistically close clusters are formed where semantically similar labels are merged into the same cluster.	158
5.13	Example of temporalised hierarchies inferred from two training videos of the ‘Taking Medication’ and ‘Cleaning Object’ activities in the CAD dataset.	162
5.14	Example of temporalised hierarchies inferred from two training videos of the ‘Having a meal’ activity in the LAD dataset.	163

5.15 Example of temporalised hierarchies inferred from two training videos of the
‘Having Breakfast’ and ‘Lunch in a Restaurant’ activities in the CLAD dataset. . 165

List of Tables

5.1	Overview of the properties of every dataset used during experimentation.	137
5.2	Performance measurements with and without assumption of ground-truth temporal segmentation based on accuracy, precision and recall of the higher-level activities in the CAD dataset.	140
5.3	Performance measurements with and without assumption of ground-truth temporal segmentation based on accuracy, precision and recall of the sub-level activities in the CAD dataset.	141
5.4	Performance measurements based on accuracy, precision and recall of the high-level activities in the LAD dataset.	145
5.5	Performance measurements based on accuracy, precision and recall of the sub-level activities in the LAD dataset.	146
5.6	Performance measurements based on accuracy, precision and recall of the sampled sub-level activities in the CLAD dataset.	148
5.7	Performance of our clustering method, which uses the proposed distance measure between activity annotation intervals, compared to other commonly used clustering methods is presented. The metrics used are normalised mutual information (NMI), homogeneity (HOM), completeness (COM) and v-measure (VM).	156

5.8	Performance of our hierarchy learning approach which automatically infers the <i>parent-child</i> relationship between activity nodes is presented for all datasets. Our method is compared to a well-defined baseline which uses trivially perfect graph to infer the <i>parent-child</i> relationship. Also, the use of semantic activity node matching and the use of multi-label representation of activity nodes is evaluated and results reported.	159
5.9	Performance of inferring activity parse tree on test videos in each dataset shown. Correctly parse terminals (CPT) metric is used to compute the performance values. Our method is incrementally equipped with proposed features and compared with a well-defined baseline which uses a relatively trivial mechanism. The table presents results with using automatic recognition of low-level action stream. . . .	167
5.10	Performance of inferring activity parse tree on test videos in each dataset shown. The correctly parsed terminals (CPT) metric is used to compute the performance values. Our method is incrementally equipped with the proposed features and compared with a well-defined baseline which uses a relatively trivial mechanism. The table presents results with using ground-truth recognition of low-level action stream.	168

List of Algorithms

1	Agglomerative Clustering with Complete Linkage	79
2	Temporal Knowledge Augmentation	100
3	Abstraction of Hierarchies	107

List of Symbols

\mathfrak{R}	Set of RCC-3 relationships
N_F	Number of frames in a video
\mathcal{E}	Set of entities within an activity video
N_E	Number of entity pairs in set \mathcal{E}
\mathbf{M}	Matrix of RCC-3 qualitative spatial relations between pairwise entity pairs for each frame
\mathbf{d}	Vector of data elements from a run-length encoding of a RCC-3 relation strings
\mathbf{c}	Vector of counts of elements in \mathbf{d} from run-length encoding of RCC-3 relation strings
I	Representation of matrix \mathbf{M} after encoding with either local or global compression
χ	Parameter defining maximum length of QSR relational changes that represent a codewords
V_l	Set of all permutations of QSR relations up to length χ
F_1	The extracted feature space from qualitative spatial representation using RCC-3 relations
\mathbf{X}	A Euclidean distance matrix capturing distance between entity pairs for every frame
\mathbf{x}	A row vector of \mathbf{X}
β	Parameters of a linear regression model
F_2	The extracted feature space from qualitative spatial representation using SQTC relations
\mathbf{r}	A vector of ratios of relative lengths of RCC relation intervals
\mathcal{C}	Set of learned temporal relationships which capture durational information
K^c	Set of cluster centroids resulting from k-means clustering
F_3	Extracted feature space from qualitative temporal representation using learned relations
\mathcal{D}	The distribution which models the Euclidean distances between entity pairs
m	Index of the moment of a distribution
S	Set of first four moments computed for a vector of distances drawn from a distribution
F_4	Extracted feature space from quantitative representation using <i>distance</i> statistics over time
F	Combined feature set for the QQSTR approach as a concatenation of F_1 , F_2 , F_3 and F_4
N_T	Number of annotators of a single video
\mathcal{A}	A set of activity annotation answers of all annotators for a single activity video
A	A set of annotation entries from a single annotator for a single activity video
ι	A single instance of an activity annotation interval
s	Start frame of the annotation interval ι
e	End frame of the annotation interval ι
l	English label describing the activity in the annotation interval ι
λ^l	Semantic similarity function between two English language labels l_i and l_j
δ	Semantic and temporal consistency based distance between two annotations ι_i and ι_j
η	A normalisation function to bound the value in the range $[0, 1]$
\mathbb{A}	A set of combined annotation entries from all the annotators for a single video
N_D	Number of annotated activity intervals in a combined annotation $ \mathbb{A} $
Θ	Output from our clustering method as a set of clusters assignments
θ	Any one cluster assignment in Θ
\mathcal{D}	Dendrogram output from our agglomerative clustering method
C	Set of clusters in agglomerative clustering algorithm
\mathbf{D}	Distance matrix between data points in agglomerative clustering
ω	Activity Node $\omega(\bar{s}, \bar{e}, L)$
\bar{s}	Average start time, computed from all start times of activity intervals of this clusters
\bar{e}	Average end time, computed from all end times of activity intervals of this clusters
L	Set of labels in the cluster $L = \{ll \in C\}$ which is referred to as ‘activity node’

N_A	Number of activity nodes in a video
Ω	Set of all N_A activity nodes in a video
G	A Graph
V	Set of vertices of a Graph
v	Any one vertex of a graph $v \in V$
E	Set of Edges of Graph
ϵ	Any one edge of a graph $\epsilon \in E$
$w(\epsilon)$	Weight of an edge in a graph
X	A subset of vertexes $v \in V$ in a graph such that $X \cap Y = \phi$
Y	A subset of vertexes $v \in V$ in a graph such that $Y \cap X = \phi$
\mathbf{B}	A bi-adjacency matrix between two subsets of vertexes of a graph
λ^L	Semantic matching between two activity nodes' label sets L_i and L_j
o	Function to compute temporal overlap between two activity nodes ω_i, ω_j
\mathcal{C}	Cost function which gauges <i>parent-child</i> relationship between two activity nodes
H	A learned hierarchy tree of activities from a single video
\tilde{H}	A learned temporalised hierarchy tree from a single video
Ψ	Set of activity clusters in a temporalised hierarchy
ψ	One single activity cluster
Q	Set of activity nodes in an activity cluster
\mathbf{R}	Temporal relations matrix of an activity cluster
Π	Set of parent activity nodes in a temporalised hierarchy
π	One single parent activity nodes in a temporalised hierarchy
Φ	A function that maps a parent activity node to its child activity clusters
P	A probability distribution over the set of activity clusters of a parent node
\mathcal{V}^{Train}	Set of training videos
N	Number of training videos
\mathcal{V}^{Test}	Set of test videos
M	Number of test videos
\mathcal{H}	Set of temporalised hierarchies for a top-level activity
J	Left hand side of a language grammar rule
K	Right hand side of a language grammar rule
ρ	Probability of a language grammar entity for example a grammar rule or a parse tree
τ	Parse tree over a set of words in a sentence
\mathcal{G}	A language grammar
\mathcal{M}	A generalised hierarchical model
\mathbb{G}	An activity grammar
\mathcal{L}	Set of low-level actions
L	A single low-level action
N_L	Number of low-level action terminals
\mathcal{T}	Set of partial parse trees over an input test video
T	A single activity parse tree
$N_{\mathcal{T}}$	Number of activity parse trees recovered
\mathbf{s}	A vector of ILP sum across the selected column of optimisation
\mathbf{p}	A vector of ILP probabilities across all rows or trees of the optimisation
\mathbf{A}	A matrix of ILP encoding coverage parse tree \times low-level actions, values in range $[0 - 1]$
i, j, k	Counter variables or indexes
x, y	Cartesian coordinates in 2D space

Chapter 1

Introduction

1.1 Motivation

One of the most challenging areas of research in computer vision is understanding data from visual input received by intelligent systems. Visual sensors are increasingly being used as an inexpensive modality to capture vast amounts of visual data. Humans possess a very distinct ability to observe their surroundings and understand the activities that occur within their complex environments. Even though activities present themselves as highly subjective and complex streams of continuously changing visual data, the human mind is able to extract the most distinctive and abstract information and organise it into activity classes. Developing an automatic learning system that is able to simulate the human brain to identify activities from visual data has been a popular research direction in computer vision [Turaga et al., 2008]. Automated activity recognition presents with many challenges. One main challenge in developing such a learning system is that different humans perform similar activities with high variation making the task of abstractly representing activities difficult [Kim et al., 2010]. Another challenge is the limited field of view granted by visual sensors, such as video cameras, on-board an artificial intelligent system. This results in a partial view of the environment which causes occlusions, incomplete and noisy human observations etc. The resulting perception of the human activity becomes limited and as a consequence the task of representing such activity challenging.

There are many applications for an activity recognition system, perhaps the most obvious being in the security and surveillance domain. Video cameras capture large amounts of visual data in the form of continuous stream of images during surveillance. Automatic detection of suspicious behaviour/activity by an intelligent system is a desirable feature that would help to create a smarter and more cost effective surveillance system. Taking a video from cameras as input and processing this continuous stream of images to learn patterns and features of human behaviour, is the key task at the core of activity recognition research.

In recent years, activity recognition has gained popularity within autonomous systems, such as robots or self-driving cars. Advances in robotics has equipped robots with greater computational power allowing robotic platforms to capture large amounts of real-time visual data and perform computationally intensive human intelligent tasks. Allowing a robotic system to use on-board sensors, such as depth video cameras, to automatically recognise activities within its environment, would bring about a greater range of applications. For example, a robot deployed in a hospital can be equipped to patrol the area and automatically recognise activities and detect situations where staff help is required. Similarly, an autonomous car can be equipped with an intelligent system which detects activities such as ‘children running on the street’, so that automatic braking can be applied and accidents avoided. Having autonomous systems operate for longer times and capture longer lengths of visual data would allow for learning human behaviour over longer periods which would in-turn help intelligent systems gain further utility. Areas such as human robot interaction will also benefit since robots will be better equipped to navigate and interact with humans in modern environments.

Therefore, a successful autonomous system requires the ability to understand human behaviour over long lengths of time. The major challenge deriving from this requirement is how to best represent human activities and human behaviour to abstractly capture the most important and discerning aspects and disregard redundant information from visual input data. This is an on-going research problem in the field of computer vision. The research problem is aggravated as human activities extend over longer periods of time. This is because longer activities tend to involve more objects, more behavioural changes and larger number of interactions resulting in complex spatio-temporal structures. Much work has been done on performing activity modelling and recognition on simple activities that span over short periods of time. However,

relatively little research tackles the problem of representation and recognition of activities which span over longer periods of time.

In this thesis, we tackle the problem of how an intelligent system can learn and understand human behaviour which spans over varying lengths of time. The research problems are stated thus: can we learn and model simple primitive activities which span short periods of time, as well as complex human activities expand over longer periods of time; can representations be found which effectively capture the underlying structures of these activities such that this knowledge can be generalised to recognising similar activities with varying details? We propose a framework for representation and recognition of simple and complex activities in this work.

1.2 Terminology

In the activity recognition literature, there is large disagreement and inconsistency in the terminology used to describe various aspect of activities [Lavee et al., 2009]. The term ‘activity’ has been interchangeably used with ‘event’, ‘action’, ‘behaviour’, ‘gesture’, ‘primitive’ and ‘movement’ [Cohn et al., 2002, Turaga et al., 2008, Bobick, 1997, Zelnik-Manor and Irani, 2006]. This confusion stems from the ambiguity in natural language words used to describe activities. A general definition of activities has been proposed by [Lavee et al., 2009] in an attempt to unify the disparity between the terminology used. According to this definition an ‘event’ or ‘activity’ must satisfy the following three conditions:

1. They must occupy a period of time,
2. They are built of smaller semantic building blocks of other events,
3. They can be described using the salient aspects of the video sequence input.

Following this definition, in this thesis, we use the following terminology to refer to different types of activities. A ‘high-level activity’ refers to an activity which specifically follows the second condition, stating that this activity must be built of smaller semantic building blocks. ‘Low-level activities’ are then referred to as activities which cannot be further broken down into smaller semantic building blocks within our representation. The ‘low-level activities’ term

is interchangeably used with ‘actions’, ‘terminals’ and ‘primitives’. ‘Top-level activities’ refer to those activities which do not have any parent activities, i.e. they are not part of any other ‘high-level activity’. We also use the terms ‘complex activities’ and ‘simple activities’. ‘Simple activities’ refer to activities that exhibit only a small amount of interactions and span only short periods of time. ‘Low-level activities’ are often also called as ‘simple activities’. ‘Complex activities’ are analogous to ‘high-level activities’ with the following added attributes: these activities span longer periods of time as compared to ‘simple activities’ and they exhibit sub-activities at many levels of granularity.

Specific to our activity recognition approach presented in this thesis, we define the term ‘entity’ referring to any element within the activity scene that is detected by the computer vision tracking systems. In the datasets used within this work, ‘entities’ refer to tracked skeleton joints of humans and tracked objects within the activity videos.

1.3 Goals and Challenges

In this thesis, our general goal is to present an effective approach to perform activity modelling and recognition of 1) simple activities and 2) complex activities. Activity recognition of each of these activity types presents its own goals and associated challenges.

Our goal for simple activity recognition is to find a representation of activities which abstractly captures the interactions between entities within the activity. Then, given an unseen video clip of a simple activity, our system must be able to compare the new video with learned models of activities to classify the activity within the new video. This goal presents common challenges found in activity recognition and computer vision research. The challenge from activity recognition is to find a good representation of activities which should abstract only the relevant information from video. At the same time, the challenge from computer vision is to handle noisy observations stemming from low-level image processing tasks such as object and skeleton recognition and tracking. Our chosen representation needs tackle these challenges by being robust to spatio-temporal variations which present due to either natural variations in the activity or noisy observations from image processing. Activity videos comprise a long sequence of still images containing a lot of information and abstracting a good representation from such

complex data is a challenging aspect of our goal.

Our framework addresses these challenges by modelling simple activities using qualitative and quantitative representations. Qualitative representations make use of *qualitative spatial relationships* (QSRs), which abstract continuous quantitative data, stemming from noisy visual observations of entity locations in an activity video, into qualitative categories designed to capture interesting aspects of activities. In this way a higher level of abstraction is achieved. Furthermore, we use quantitative representations to model the continuous quantitative data using highly descriptive statistics. Together the collective representation is able to address the challenges presented.

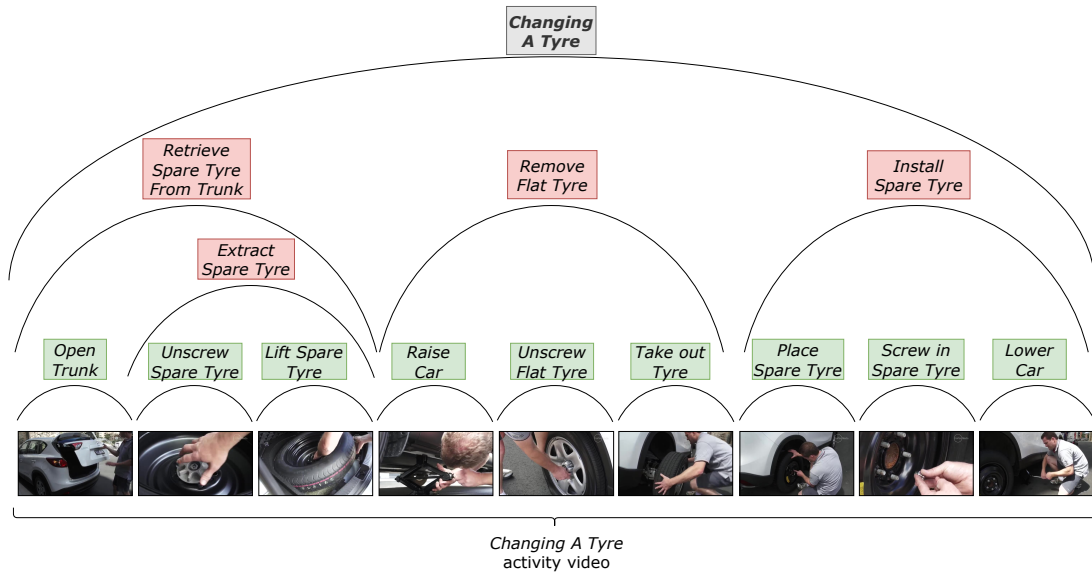


Figure 1.1: An example of a hierarchy of high-level activities inferred from detections of low-level activities. High-level activities are shown using red boxes whereas low-level activities are represented by green boxes. Recognising such a hierarchical structure of activities over a test video is our key goal.

Our goal for complex activity recognition is highly challenging. We aim to build an intelligent system which automatically represents a complex activity in a syntactical structure which describes all of its composed sub-activities in a hierarchical format. We show an example of this goal on a sample video in figure 1.1. In this figure, an activity video of a sample activity of ‘changing a tyre’ is given as input. Then each segment of the video is classified as a low-level activity class, seen in green. Note that there may be any arbitrary number of levels which define

the activity hierarchy. The classification process of the low-level activity classes is performed by our simple activity recognition method mentioned above. However, note that higher level classes are recognised as conglomerations of the low-level action detections, shown in red. These higher-level classes represent the higher-level activities and they comprise of task dependent set of sub-activities. Repeated combinations of these classes describe larger and larger segments of the input videos until the top-level activity class is detected which describes the activity over the entire video. The structure can be read as a hierarchy of detection of activities at various levels of granularity. Inferring such a rich structure that progressively describes activities over many different segments of the activity video is our goal for complex activity recognition.

This goal presents many challenges. The biggest challenge is to learn the hierarchical structure of activities and represent activities in such a structure. This is necessary to avoid learning explicit representations directly from the raw video for each activity class at different time spans, independently of other classes. This method is inefficient as redundant information will be represented within higher-level classes. For example, the explicit raw video representation of the activity class ‘extract spare tyre’ will be highly similar to the representation of its component classes ‘unscrew spare tyre’ and ‘lift spare tyre’. It is therefore our goal to model higher-level activities as a combination of its sub-level activity classes. Learning to represent a complex activity as a hierarchy of sub-activities without explicitly modelling each class from raw video, is therefore a challenging problem.

Our goals presented here lend towards the new and upcoming field of explainable AI [Gunning, 2017, Voosen, 2017]. Not only do we wish to model complex activities as hierarchical structures of component activities, as show in figure 1.1. We also aim to build models which are human-understandable. Our approach therefore differs from the popular deep-learning approaches [Ibrahim et al., 2016, Ordóñez and Roggen, 2016, Yang et al., 2015] which are able to learn extremely accurate representations of complex activity, but are hard to interpret and explain the recognised complex activity as a human-understandable structure of sub-activities. Recent research addressing the issue of explainable AI (XAI) [Zhang et al., 2017, Zhang and Zhu, 2018, Selvaraju et al., 2017] has begun to tackle the problem of interpret-ability of deep-learning architectures however this area is still in its infancy and complete understanding of intermediate layers within a deep network, sometimes consisting of hundreds of layers, is a

challenging problem. Moreover, for example, in the domain of image classification, interpreting intermediate layers within a deep network has revealed features which capture edges, high pixel gradients and so on [Zeiler and Fergus, 2014]. Similarly, in video analysis, intermediate nodes within a deep architecture are likely to capture short tracks of motion of various objects and entities within the scene. This is evident from an automated activity grammar learning approach by [Zhang et al., 2011]. Also mapping a descriptive linguistic label to tens or hundreds of nodes in intermediate layers within a deep architecture is cumbersome. By contrast, in our method, each intermediate node within our proposed complex activity hierarchy succinctly defines a sub-activity represented in natural language as described by humans as seen in figure 1.1. This makes such a representation highly understandable.

In order to address the challenges presented in complex activity recognition, we utilise mark-up of activities acquired from multiple annotators in a video corpus to learn human-understandable structures of complex activities. Multiple human annotators tend to identify activities at different levels of conceptual granularity. Our method automatically infers a ‘part-of’ relationship between intervals and generate a hierarchical activity model from this data using semantic similarity of textual annotations and temporal consistency. We use the resulting model to interpret previously unseen videos in terms of the conceptual categories of the acquired model, thereby providing a layered/compositional description that is naturally understandable by people. Our method is robust to noise and is able to represent complex activities in an efficient compositional structure.

1.4 Thesis Structure

The aim of our work is to model and recognise simple activities and complex activities. We aim to build a representation of simple activities as a combination of qualitative and quantitative features extracted from entity locations in raw activity video. We also aim to automatically learn and represent complex activities as compositional structures of its component activities. The flowchart shown in figure 1.1 shows the two frameworks and how they are tied together.

The remaining structure of this thesis is as follows:

- Chapter 2: We present a review of related approaches for general activity recognition

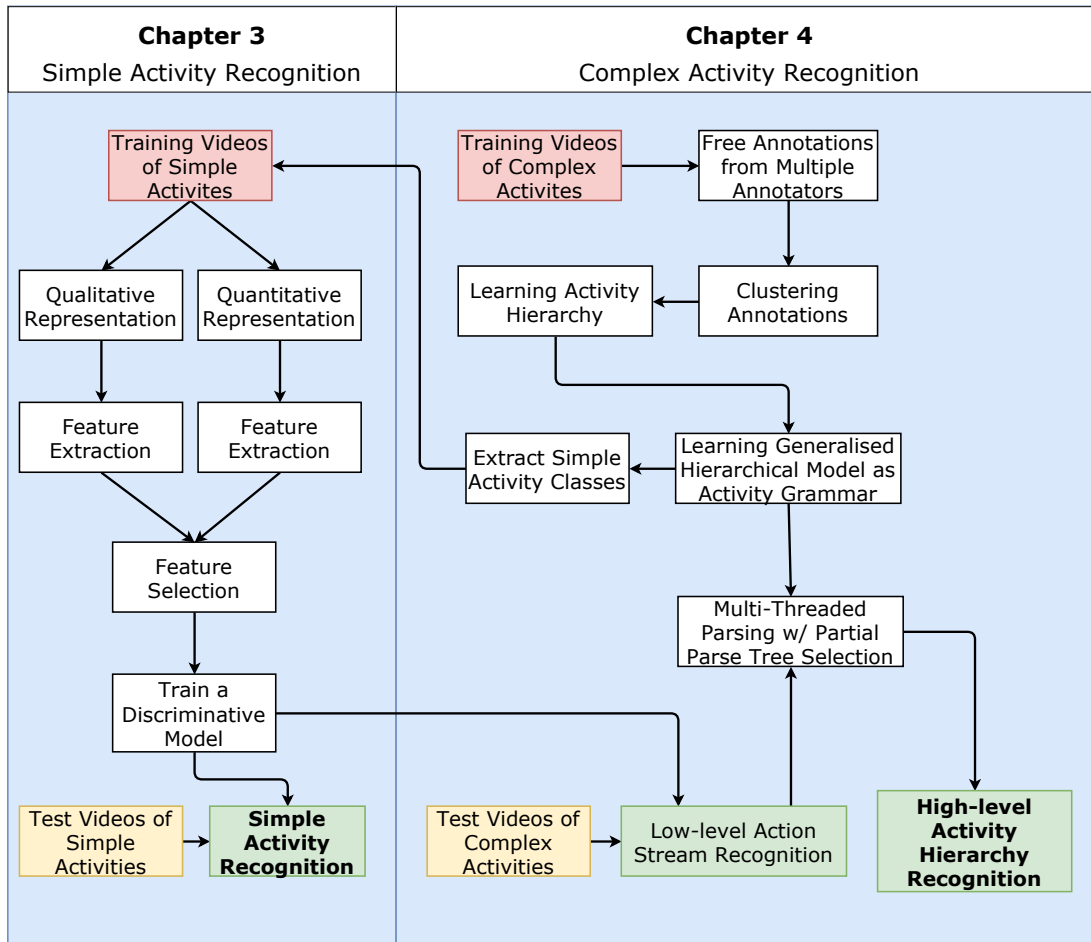


Figure 1.2: A flowchart showing different components of the presented frameworks in this thesis. Also, the interaction between the different frameworks is shown. Pink boxes represent training data. Yellow boxes represent testing videos. Green boxes represent recognised results.

in this section as well as a brief history of the evolution of the research area. We also present related work on activity representation that inspired us to develop our approaches. Furthermore, recent state-of-the-art approaches that tackle similar challenges as the ones mentioned before, are also presented.

- Chapter 3: In this chapter, we introduce our simple activity recognition framework. Note that in figure 1.2, the framework comprises of a first generating a qualitative and quantitative representation from activity training videos. Relevant features are then extracted from the generated representations. These features are pruned through a selection process

and finally a discriminative model is trained to learn the activity classes. Test videos are then classified into learned activity classes using the discriminative model.

- Chapter 4: In this chapter, we present our framework to perform complex activity recognition. Training videos are first freely annotated by multiple annotators. The collective annotations from multiple annotators for a single video are then clustered to remove repeated annotations. A cost-based optimisation is then used to learn the *parent-child* relationships between annotations and build an activity hierarchy per video. Multiple such hierarchies across all training instances are combined to learn a generalised hierarchical model of activity, represented as an extended grammar. Low-level action classes are then extracted from the learned general activity model. Our simple activity recognition framework is employed here to learn a discriminative model of the low-level activity classes present in the generalised hierarchy model, shown as the ‘Extract Simple Activity Classes’ module generates training videos for simple activity recognition in figure 1.2. Then, given a test video, the newly learned discriminative model is first used to detect a stream of low-level actions over the test video, green boxes in figure 1.1. The learned generalised hierarchical model, represented as activity grammar, is then used to infer the complex activity hierarchy, red boxes in figure 1.1, comprising of all its components activities up until the detected low-level actions.
- Chapter 5: We evaluate our approaches using multiple dataset in this section. Three datasets are used exhibiting increasing levels of complexity and longer videos. We experiment and evaluate each section of our frameworks, figure 1.2, for both simple and complex activity recognition.
- Chapter 6: Finally, we present conclusions drawn and summarise the contributions of this thesis. Also possible future directions of research are presented in this section.

In summary, figure 1.2 shows a framework of our approach. Broadly speaking, our key contributions are presenting novel approaches for activity modelling and recognition at varying levels of granularity, namely simple activity recognition and complex activity recognition. Specifically, in the simple activity recognition framework, we propose a new representation ex-

tracted from raw videos using qualitative and quantitative features jointly. we then present a method for performing feature extraction from the proposed representation. We use these features to learn a discriminative model of simple activities and show state-of-the art performance on the simple activity recognition task. For complex activity recognition, a novel mechanism is presented to learn latent hierarchical structure within complex activities by automatically inferring the *parent-child* relationships between various sub-activities. This is followed by presenting an approach to abstractly learn a generalised model from such syntactical structures. We then propose an end-to-end recognition mechanism which begins by first extracting the low-level action classes from the learned generalised model. These low-level action classes are then modelled using our simple activities framework. Given a test video, simple activities are detected using the pre-learned model. Finally, we propose a mechanism for parsing the stream of low-level action detections using the learnt generalised activity model in order to infer the complex activity hierarchy which best fits over the low-level actions. We compare the recognised complex activity hierarchies against well-defined baselines and show significant performance increases using our methods.

Chapter 2

Related work

2.1 Previous Surveys

A vast number of approaches have been developed which process raw visual data and extract semantically meaningful information about the underlying activities. Many survey papers have been proposed that group the activity recognition research literature into relevant and meaningful categories. However, often these categorisations are motivated by specific goals stemming from the area of interest of the authors. For example, authors in [Vishwakarma and Agrawal, 2013] present a survey of general activity recognition approaches motivated by applicability within the surveillance domain. They evaluate approaches with a focus on environment modelling, occlusion handling, unusual activity detection and other such surveillance related challenges. In a separate survey, the authors in [Lara and Labrador, 2013] present a survey of human activity recognition aimed at sensor-based approaches applicable to wearable sensors. These are mobile sensors and data collected from this modality exhibits higher amount of occlusion and partial views as well as noise due to the motion of the wearer. In another survey [Poppe, 2010], approaches to activity recognition using low-level vision such as gait representation, optical flow etc. are presented. This survey also focuses on activities of short length (a few seconds) which often exhibit little multi-human interactions. Perhaps one of the most general and semantically well categorised surveys is presented by [Lavee et al., 2009]. This

survey categorises approaches by the techniques employed to model activities rather than being driven by specific applications.

In this thesis, we are motivated by modelling of activities based on their complexity. We therefore seek to categorise approaches which are distinguished by their ability to model and represent activities along a spectrum of complexity. The authors in [Turaga et al., 2008] and [Aggarwal and Ryoo, 2011] have presented surveys which categorise approaches based on their complexity spectrum. Figure 2.1 shows the survey overview of categorisation presented in [Turaga et al., 2008].

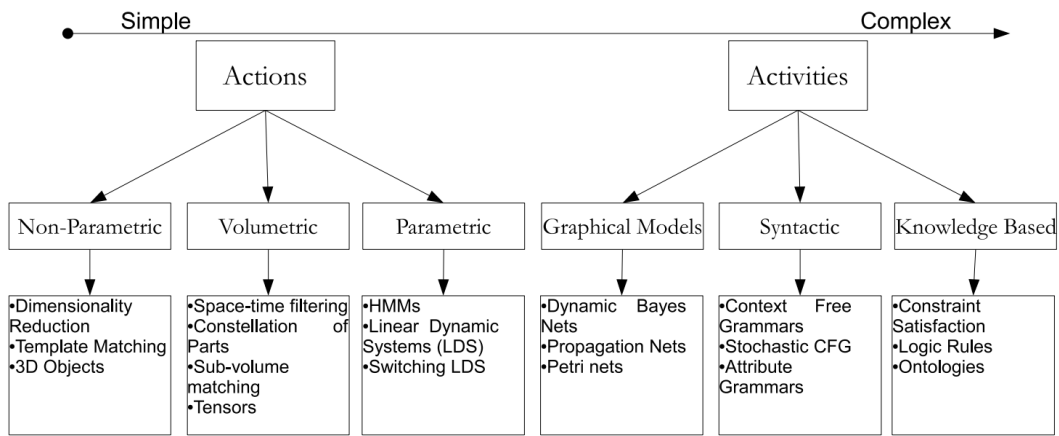


Figure 2.1: An overview of the approaches of simple to complex activity recognition. Image taken from [Turaga et al., 2008]

Note that approaches are categorised into groups, from simple activities to complex activities. This categorisation also loosely represents the chronological order in which activity recognition methods have been researched and developed. Briefly, earliest methods tackled simple activity recognition problem and attempted to model primitive actions alone by using non-parametric approaches such as template matching [Bobick and Davis, 2001, Polana and Nelson, 1994, Ke et al., 2007]. Parametric approaches were then developed to encode the temporal sequence of motions resulting in an action. This leads to the vast literature on activity recognition using Markovian models [Yamato et al., 1992, Starner and Pentland, 1997, Oliver et al., 2000]. As activities become more complex, graphical models were introduced such as DBNs [Subramanya et al., 2006], Petri nets [Albanese et al., 2008] etc. These methods model complex

activities by abstractly encoding their various sub-activities and their relationships within a graph structure. Syntactic methods [Dick and Cerial, 1990, Ivanov and Bobick, 2000, Ryoo and Aggarwal, 2006, Zhang et al., 2011] gained popularity to model highly complex activities which comprise of many levels of sub-activities of varying granularity. Finally, knowledge-based methods used logic rules to model complex activities. Our work is broadly categorised into simple activity recognition and complex activity recognition. Our simple activity recognition method falls into a new category of qualitative representation of actions, whereas our complex activity recognition method falls into the syntactic category within the taxonomy presented in figure 2.1.

Authors in [Aggarwal and Ryoo, 2011] propose a similar categorisation motivated by activity complexity, but with different sub-categorisation of approaches by non-hierarchical and hierarchical models. These surveys closely relate to our aim in this thesis and therefore we use a similar categorisation of the presented related work in those categories in this chapter.

2.2 Simple Activity Recognition

Activity recognition systems have a long and rich history of research. Approaches can be broadly categorised into vision-based activity recognition systems [Aggarwal and Cai, 1999, Turaga et al., 2008] and sensor-based activity recognition systems [Choudhury et al., 2008, Lara and Labrador, 2013, Chen et al., 2012]. Vision-based approaches rely on video and depth cameras to capture activity data. Computer vision techniques are then used to process this data and extract information about the activity occurring within the video. Contrastingly, sensor-based approaches rely on various non-vision sensors, such as motion sensors, RFID sensors etc. which are often placed on human subjects and/or objects. Data from these sensors provide a representation of the movement of various entities within the activity. Both approaches of activity recognition have their pros and cons. Sensor-based approaches are gaining popularity with the introduction of deep learning techniques [Ravi et al., 2016, San et al., 2017]. These approaches can obtain accurate detections from sensors with little noise. Sensors can be attached to human subject and surroundings to provide a full view of the activity. Sensor data is also less computationally intensive to process. However, sensor data does not provide rich information

content as video.

Vision-based approaches suffer from noise, computational complexity and limited camera frame for the activity to be observed. However, unlike sensor-based approaches, vision-based systems yield rich information of an activity in form of a video. Furthermore, in the domain of artificially intelligent systems such as robots, autonomous cars, surveillance systems, it is often not feasible to equip multiple people with intrusive sensors, such as IMU sensors or other markers, in order to gather data for performing activity modelling and recognition. This is particularly valid in case of observing long-term complex activities of daily human behaviour. Therefore, we assert that vision-based approaches, which utilise video and depth cameras, are more suitable for observing naturally occurring human activities over long periods of time. In this related work section, we provide an overview of the previous research performed on vision-based approaches.

2.2.1 Pixel-based Representations

These approaches operate on pixel-level information extracted from images in videos. Information from images can be extracted in form of colour information, textures, gradients, optical flow etc. Activities are represented through capturing the relationships between neighbouring pixels within the images in the video. Conceptually speaking, capturing the flow of colour or the changes in pixel values is useful to capture some notion of motion and thereby the latent activity. Previous work has focused on capturing these changes within neighbouring pixels by using histogram of gradients (HOG) between pixels [Dalal and Triggs, 2005, Yang et al., 2012]. The authors in [Laptev, 2005] introduced the use of Spatial Temporal Interest Points (STIP) to achieve abstraction of activities from videos. This process captures *interesting* patches of images across different frames in the video and models the change between patches. One way of identifying interesting patches in images is using Harris corner detector [Harris and Stephens, 1988].

Pixel-based representation have been successful in recognising simple activities, primarily those which exhibit repetitive or periodic motion such as ‘jumping jacks’, ‘running’ etc. Methods often capture the variations in values of interesting image segments across time to represent

these activity's structures. Histogram of gradients (HOG) and histogram of optical flow (HOF) features have been popularly used to capture this variation, in a fixed length feature spaces [Dollár et al., 2005, Laptev et al., 2008]. The authors in [Schuldt et al., 2004] used STIP features to produce robust models of activities which were trained using an SVM.

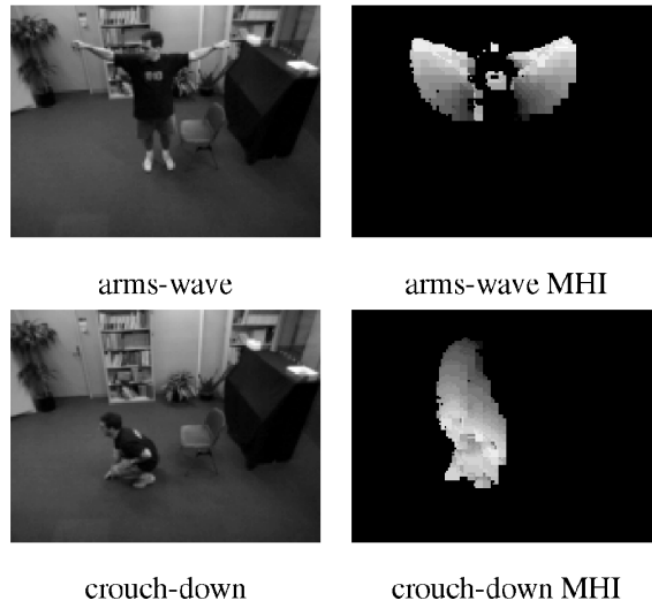


Figure 2.2: Example of a pixel-based representation to generate templates of activities. Image taken from [Bobick and Davis, 2001]

Template-based approaches are an early set of approaches classified as pixel-based approaches. In activity recognition, template-based approaches extract templates of motion from activity videos. Activities are first converted to low-level shape representations referred to as templates. A template for each activity is extracted as an abstract representation of the shape a human take when performing that activity, see figure 2.2. The authors in [Polana and Nelson, 1994] proposed a method to detect human posture using a periodicity measure. In this approach image patches encompassing human subjects are extracted from the the activity video. These are segmented into cycles, which capture the repetition within the activity. The template represents a repeating set of body postures which encode that periodic activity. Recognition of new activities is simply performed by matching the extracted flow features to existing templates to classify activities. The authors in [Bobick, 1997, Bobick and Davis, 2001] have expanded

on this idea by introducing a ground breaking approach which includes a notion of temporal knowledge within the activity representation. They proposed a real-time activity recognition system using the template matching approach. They represented each activity with a set of images that acted as templates. These images were created using binary cumulation of motion images called motion history images and motion energy images, as shown in figure 2.2. Images were assigned weights according to their location in the video thereby capturing some aspect of temporal knowledge within the template. By using template matching techniques, simple activities such as ‘sitting down’ or ‘waving arms’ could now be detected. Other approaches have further expanded on using template matching [Ke et al., 2007, Rodriguez et al., 2008] by either automatically finding human segment volumes through over segmentation [Ke et al., 2007] or using maximum average correlation height filter from object recognition research and applying this to capture characteristics of activity volumes to perform activity recognition [Rodriguez et al., 2008].

An undeniable benefit of these approaches is their ease of implementation, since they require no pre-processing of object tracks or skeleton track within the video. These approaches also benefit from simplicity. They have proven to be beneficial and successful for activities which can be represented as a well-defined set of arbitrary actions. However, more complex activities involving multiple humans and objects present a challenge for these approaches. Therefore, work on object-based representation is explored next.

2.2.2 Object-based Representation

Activities which exhibit daily living scenarios often involve interaction with other objects and/or other humans. Pixel-based approaches work well in scenarios where the motion of a person can be isolated from their surroundings and represented explicitly. In a natural dynamic environment, many entities around a person are often constantly in motion. For example, consider a ‘lunch in a busy restaurant’ activity where many people and objects can be seen moving within the camera frame. Reasoning about the complex world is often easier by first using computer vision algorithms to detect objects of interest within the scene. Recall that we introduced the term ‘entity’ to refer to any physical object and tracked skeleton joints of a human. In

the current context, objects refer to any interesting entities including human skeleton joints. Furthermore, reasoning about objects also allows to extract interesting interactions and motion patterns without needing to manage the image specific information such as colour or pixel values which are often prone to noise from external factors such as lighting changes, camera shaking etc. This is however based on the premise that the object tracking and detection produces accurate results which is often a big ask.

In object-based representation, activity videos are first abstracted into entities along with their properties such as location, size, linguistic label etc. Objects are commonly represented as minimum bounding boxes (MBR), blobs or (x,y) locations as approximations of their shapes. This representation obviously loses some information about the object's exact shape but high-level reasoning can be successfully performed using such representations [Cohn et al., 2012, Hongeng and Nevatia, 2001, Sridhar et al., 2008, Dubba et al., 2015]. Interactions of entities within an activity are key elements of the activity [Chen et al., 2015] and an approximated representation of objects allows for easily encoding their spatial and temporal knowledge to aid in extracting interactions within the activity. Trajectories are another popular method to perform object-based representation [Piciarelli and Foresti, 2006, Piciarelli and Foresti, 2007, Porikli, 2004]. Objects are tracked through the environment across many frames of the video and trajectories are extracted. Interactions of these trajectories are modelled and activities are represented using trajectory information. [Duckworth et al., 2016a] performed trajectory-based activity recognition by modelling interactions between trajectories of humans in an environment and interesting objects.

2.2.3 Qualitative Representation

Following from the idea of representing interactions and positional information of entities within an activity, in this section, we describe work related to qualitative representation of spatial and temporal information about entities within an activity. An activity can be naturally described in terms of qualitative description [Amorapanth et al., 2010]. Such a description abstracts away from precise quantitative information which may capture unnecessarily too much information to describe the underlying activity. For example, an activity of 'pick up phone' can be described in

a natural qualitative manner as ‘hand reaches phone followed by hand and phone approach the head’ etc. This description sufficiently captures the information needed to represent the said activity. Quantitative knowledge such as the speed of the hand towards the phone or the precise (x,y,z) location of the phone etc. are all examples of unnecessary quantitative information which add little value to the representation this activity. However, it is possible that such information may be necessary to model other activities to be learned therefore the qualitative representation is either engineered according to the domain of activities or automatically learned from training data [Bleser et al., 2015]. In any case, excessive representation of an activity could result when using quantitative descriptions. Moreover, a different instance of an activity is unlikely to share specific quantitative details whilst qualitative representation will result in high similarity between instances of similar activities. Therefore, using qualitative abstraction allows for discretising the continuous quantitative space and time into meaningful categories which are aimed to be relevant in order to capture activities.

Qualitative representation comprises of a set of jointly exhaustive and pairwise disjoint (JEPD) relations. Many different qualitative spatial and temporal calculi have been proposed in literature each aimed at modelling some aspect of space or time. Examples of these include topology, direction, distance, time etc. [Chen et al., 2015]. The authors in [Ferryhough et al., 1998] used these relationships to demonstrate a process aimed at removing noise from image sequences. The authors in [Cohn et al., 2012] list many other real-world applications of these representations. In this thesis, we utilise a set of calculi which captures notions of topology, motion and time. Next, we provide a brief outline of some of the popularly used calculi which capture these aspects within an activity.

Topology

These calculi describe qualitative relationships between entities that is immune to topological transformations in space. Given entities in an activity video, the commonly used MBR representation can be used to compute representations of entities qualitatively. Two main topological calculi include RCC (region connection calculus) [Randell et al., 1992] and n-intersection model [Egenhofer and Franzosa, 1991].

Region connection calculus uses regions in space as entities and represents the topological

relations held between any pair of regions at any time. Regions in space can refer to the MBRs of various objects, as done in our work. Given two regions of space, a and b, the topological relation between these regions at any point in time is one of a set of possible relations. Figure 2.3 shows eight arrangements of two regions and the corresponding topological relation between them in each arrangement. As regions (entities) move through continuous space, they may transition to different relationships. This is shown by the arrows between relations indicating possible transitions. This diagram is known as a conceptual neighbourhood graph [Chen et al., 2015] and the relationships shown make up the RCC-8 calculus.

In practice, systems often reduce the number of relationships needed according to the type of visual input, the level of granularity within low-level entity detections and domain knowledge [Sridhar et al., 2010a]. RCC-5 and RCC-3 have been used, which are smaller sets of relationships. In activity recognition, many different techniques have utilised such relationship. The authors in [Dubba et al., 2010] used these relationships to learn about arrangement of objects within a scene such as table plate settings, and reason about the events that take place in this setting [Dubba et al., 2011]. The authors in [Sridhar et al., 2008, Sridhar et al., 2010a, Sridhar et al., 2010b] used these relationships to build activity graph which compactly represented activities.

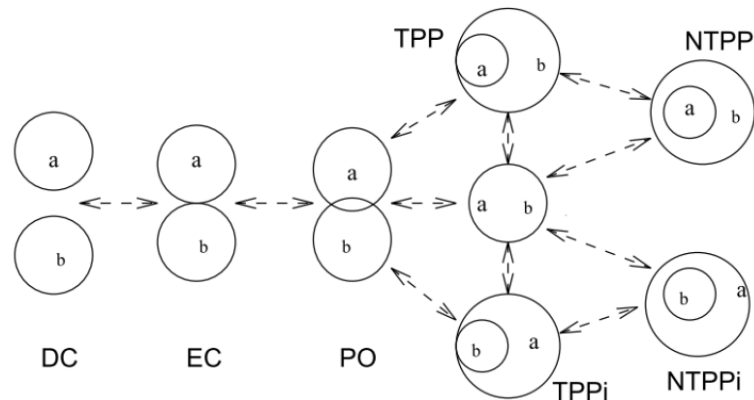


Figure 2.3: A conceptual neighbourhood graph showing the relationships of the RCC-8 calculus. Any one of the eight relationships shown can represent the spatial arrangement of any two objects, a and b, at any one time. For example, ‘DC’ denotes a *disconnect* relationship, ‘PO’ denotes a *partially overlapping* relationship and so on. For a complete list of relations, refer to [Randell et al., 1992].

In our simple activity recognition system, we utilise the RCC-3 calculus to represent topological information within activity videos. We motivate this choice and present more details in chapter 3.

Motion

There has been a strong line of research targeted toward development of calculi which explicitly represents relative motion of disconnected objects. The qualitative trajectory calculus (QTC) was specifically developed to represent and reason about objects in free space. In typical activity scenarios, objects refer to physical objects such as ‘cup’, ‘tyre’, ‘printer’ etc. or human joints such as ‘hand’, ‘head’ etc. These objects often exhibit complex trajectories of motion through the scene during an activity. These complex and continuous trajectories can be discretised and represented as a set of qualitative symbols using the QTC calculus. Representation of objects and human skeleton joints used in our work are considered rigid and therefore represented in 2-dimensional space by their centre points and/or MBRs, described in Cartesian coordinates. This allows us to adopt such a calculus. QTC was first proposed by [Weghe et al., 2004] to qualitatively capture the motion of the objects. Many variants of this calculus were later introduced. QTC basic (QTCb) was proposed by [Weghe et al., 2004], followed by double-cross QTC (QTCc) [Van de Weghe et al., 2005, Van de Weghe et al., 2006] which was built on the double-cross calculus presented by [Zimmermann and Freksa, 1996], a network type QTC (QTCn) [Bogaert, 2008] etc. This calculus has been used to represent dancing activities where motion is an important aspect of the activity in [Laube et al., 2005], authors in [Young and Hawes, 2015] have used this calculus in a robotics domain and provided comparison of its performance with many other calculi.

We utilise QTC basic in our work, a description of this particular calculus is provided here. QTC basic is a simple calculus where the positional information of objects is determined by their orientation and distance relations [Downs and Stea, 1974]. In the classic QTCb calculus, at each time-step, the movement between any two objects (O1,O2), is qualitatively represented as a two-tuple, for example (+,0), using the following rules:

1. O1 is moving towards O2, represented by (+) for O1,

2. O1 is moving away from O2, represented by (-) for O1,
3. O1 is neither moving towards or away from O2, represented by (0) for O1.

The same relationships can be applied for the converse arrangement of objects. Figure 2.4 shows the QTC conceptual neighbourhood diagram where the nodes represent all possible QTC states between two objects and the arcs represent the transition between these states. Within each node, a hollow circle represents a moving object while a solid circle represents a static object. Finally, the QTC state is represented by the two-tuple set atop each node. Note that since QTC represents motion of an object relative to another object, there exist two symbols atop each node representing the motion of each of the two objects below as either moving away (-), moving towards (+) or stable (0). This diagram shows the representation in two-dimensional space, but it can be extended to an n-dimensional space. We use a two-dimensional representation since our datasets comprise of detections of objects and skeletons embedded in the 2-dimensional image plane within an activity video.

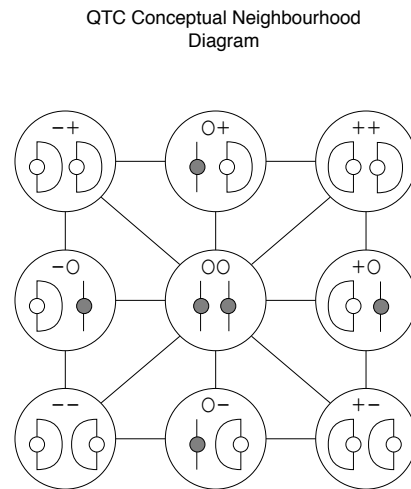


Figure 2.4: Conceptual neighbourhood graph of the basic qualitative trajectory calculus (QTCb) shown. The solid circles represent static objects, hollow circles represent moving object with the surrounding semi-circles indicating their range of motion. The corresponding QTCb relation is shown atop each arrangement of objects [Van de Weghe et al., 2006].

Time

Temporal ordering of events within an activity is undoubtedly an important aspect of any activity. Similar to qualitative discretisation of space, there exist representations that discretise time in meaningful segments. Using such a representation of time allows to encode long sequences of frames of an activity video into a low-dimensional feature space. Two main qualitative time intervals are described here. By far the most widely used temporal relations are Allen’s Interval Algebra (IA) [Allen, 1983]. This algebra expresses a unique temporal relation between any pair of intervals of time. For example, take an activity of ‘have breakfast’. The components of this activity are, for example, ‘mix sugar’ then 15 frames later ‘stirring’. IA allows us to represent the time relation between the two activity intervals, in this example case the relation would be *before*. Figure 2.5 shows a subset of IA relationships on the left between two activity intervals i and j . In an activity, there are events that may appear in a complex temporal stream rather than a sequence. Temporal relations between intervals within these complex streams can be effectively represented using Allen’s temporal algebra. IA provides for qualitative representation of temporal intervals whereby the relationship between each pair of intervals is represented by one of 13 relationships such as *before*, *after*, *meets* etc. This representation has been used in many different areas of research. The authors in [Duckworth et al., 2017, Sridhar et al., 2010b] have recently used these relations to encode the temporal ordering of events within activity graphs or intervals graphs. We utilise this representation to model the temporal ordering of component sub-activities, of each parent activity, within our complex activity recognition system.

Capturing duration of intervals is another aspect of representation of time which been studied in the INDU calculus [Pujari et al., 1999]. This is an extension of some relations of the Allen’s temporal algebra to include a notion of duration between intervals within the calculus. Inspired by the point algebra presented by [Vilain et al., 1990], which uses three basic relations namely: $\{<, =, >\}$ to describe intervals, the INDU calculus extends a subset of Allen’s temporal relations to further describe whether the intervals share a shorter ($<$), equal ($=$) or longer ($>$) relation with each other, as shown in figure 2.5 on the right. Some of Allen’s temporal relations implicitly holds information about the duration of the intervals for example Allen’s temporal


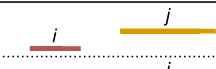




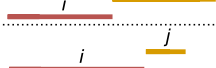



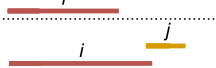
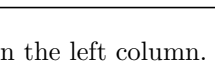
Allen's Relations	INDU Calculus Extensions
<i>before</i> (i,j) 	<i>before</i> < (i,j) 
	<i>before</i> = (i,j) 
	<i>before</i> > (i,j) 
<i>meets</i> (i,j) 	<i>meets</i> < (i,j) 
	<i>meets</i> = (i,j) 
	<i>meets</i> > (i,j) 
<i>overlap</i> (i,j) 	<i>overlap</i> < (i,j) 
	<i>overlap</i> = (i,j) 
	<i>overlap</i> > (i,j) 

Figure 2.5: A subset of Allen’s temporal relations shown in the left column. The right column shows the INDU calculus, which can be seen as an extension of the Allen’s relations with the addition of relative durational knowledge.

interval *equals* relation also implies that the duration of the intervals is of equal length. Therefore, INDU calculus only extends a subset of Allen’s temporal relations. In our simple activity recognition system, we learn temporal relations which closely resemble a simplified version of these relations. We use the INDU calculus as a validation mechanism for our learned relations which represent duration through incorporating short, equal or long relations for the ‘meets’ IA relation.

Learning Qualitative Relations

So far, we have presented qualitative relations that have been hand crafted and pre-set. However, there has been research on learning these relations from observational data. This approach presents the advantage of automatically discovering highly suitable and domain specific relationships. However, such relationships may not be as generalisable. The authors in [Mohnhaupt and Neumann, 1991] learned spatial relations between objects represented as trajectories while the work by [Fernihough et al., 1998] identified object which are likely to interact and learned spatial relations at different levels of granularity in a surveillance domain capturing videos of moving vehicles on a highway. Primitive spatial relations were first learned such as *behind*,

right, ahead etc. Following this, composite relations were inferred such as *pulling out, pulling in* etc. It is clear that these learned relations are ‘driving’ specific since they are learned from ‘moving vehicles’ training videos. This work, though proves that qualitative relations can be learned from data, was however limited to only two levels of granularity.

Authors in [Behera et al., 2012] learned non-topological relations using distance and velocity between two moving objects to create a relative feature vector. These vectors were then clustered to obtain low-level actions which were used to describe the more complex manufacturing-like activities from an egocentric point of video. This method was limited to a fixed number of objects within the scene. The work by [Al-Omari et al., 2017] contributed to this field by learning relations in an incremental fashion. This is a more suited methodology in an autonomous robot domain. Qualitative representations were incrementally learned in this system by using natural language commands which aided in segmenting various continuous spaces such as distance, colour etc. At the same time the learned representation was simultaneously being used to describe new observations made by this system.

In our work, we learn the temporal relations needed to describe the relative lengths of pairs of activity intervals, we show that the learned representation is beneficial in discriminating visually similar activities for example ‘taking a bite of an apple’ versus ‘sniffing an apple’.

2.3 Complex Activity Recognition

As activities become lengthier, they begin to exhibit higher complexity in terms of interactions occurring within the activities. Naturally, complex activities exhibit a hierarchical structure i.e. complex activities can often be described as a set of simple component activities which can further be broken down into even simpler activities. This natural structure of complex activities gives rise to research which presents activities in a graphical structure [Aggarwal and Ryoo, 2011, Turaga et al., 2008]. One of the first instances that makes use of hierarchies in computational models was introduced by [Fukushima, 1988]. Hierarchical models gained popularity as it was noticed that many naturally occurring phenomena exhibit hierarchical properties and can be modelled as such. The research by [Serre et al., 2005] uses a hierarchical architecture found in the visual cortex of the human brain to build a structure for rapid object

recognition. Work by [Ahuja and Todorovic, 2008] proposes hierarchical representation which is based on segmented image regions that are modelled by arranging them semantically into a hierarchy. The authors in [Fleuret and Geman, 2001] provided a coarse-to-fine strategy where edgelets act as the leaf nodes or terminals and are progressively built up to top-level activities. The authors in [Fidler and Leonardis, 2007] use an unsupervised learning approach which employs hierarchies to represent and categorise objects. They proposed a model for hand-written activity recognition by introducing a hierarchical multi-layered neural network. In this section, we further explore the related research in representation of complex activities by presenting two broad categories of related work, namely graphical approaches and grammar-based approaches.

2.3.1 Graphical Approaches

Graphical approaches are commonly employed to capture activities through encoding sub-activity structures into graphical models. Common graphical models comprise of trees, networks, interval graphs etc. Graphical approaches have been previously used to model simple activities. For example, hidden Markov models (HMMs) capture simple activities as a sequence of hidden states. Each state corresponds to one component of the overall activity. Also, each state generates an observation as a feature vector which can be extracted from input visual data. The model essentially captures activities as component set of sequential states and learns probabilities of transitioning from one state to the next along with learning probability of the mapping between observations and states. The work by [Yamato et al., 1992] is an early instance of an approach which utilised HMMs to model activities. Following their success, authors in [Starnner and Pentland, 1997] also used HMMs to recognise sign language. In this work, the location of hands in activity videos were extracted and each sign language word was modelled as an HMM whereby observations are represented as a sequence of hand motions and the corresponding hidden states represented words. Many approaches used such models to capture increasingly complex activities [Park and Aggarwal, 2004, Oliver et al., 2000]. Coupled HMMs introduced by [Oliver et al., 2000] was one of the first instances to model multi-person and human-human interaction. This work was followed by introduction of couple

hidden semi-Markov models [Natarajan and Nevatia, 2007] which augmented the model’s state with durational information. In recent years, these approaches have been extended to tackle more complex structures. For example, HMMs have been extended to include multiple layers to incorporate hierarchical structures of activities [Oliver et al., 2002, Nguyen et al., 2005]. The authors in [Subramanya et al., 2006] present a hierarchical dynamic Bayesian network (DBN) that jointly model activities along with their surrounding environment. DBN’s ability to deal with noise and uncertainty whilst capturing the temporal structure of complex activities make them an attractive approach.

Though these models have proven successful, they lack a deep hierarchical structure which is key for modelling complex activities. As these models increase in hierarchical levels and/or capture multi-agent interactions, they scale poorly since new latent variables are introduced and capturing the newly-introduced dependencies quickly becomes intractable. With the increasing number of latent variables, training the model requires more data making these models infeasible for complex activities [Turaga et al., 2008, Aggarwal and Ryoo, 2011]. Also, as with other approaches mentioned before, these models rely on direct inference of activity classes from raw video data or detected entities. Our premise in this thesis is that models of higher-level activities should stem from detection of lower-level activities and once low-level activities are found, higher level activities can be inferred [Aggarwal and Ryoo, 2011]. In the next section, we present grammar-based approaches which operate on this premise.

2.3.2 Grammar Approaches

As mentioned before, hierarchical structures can be found in many natural phenomena. One such area is within language. In language a sentence can be described in the form of a hierarchical structure called the parse tree. A parse tree recognises collection of words within a sentence as parts of speech. Many parts of speeches then form other higher structures such as nouns, verbs etc. Modelling language grammar and inference of the syntactical structure that describes a test sentence has been researched over a long time [Dick and Cerial, 1990]. Inspired by these models, approaches have been suggested in the activity recognition domain to model complex activities in a hierarchical manner. An advantage of using hierarchical approaches is

that much less training data is needed than other approaches. This is because only low-level activities are needed to be trained using many instances of raw visual data, the remaining high-level activities can be inferred from low-level detections. Also, redundant sub-structure which are common across multiple high-level activities can be easily modelled and recognised more efficiently. Finally hierarchical structure are able to incorporate human knowledge of how an activity is described by humans. This allows for constructing models which are semantically rich and explainable. For these reasons, in our work, we model complex activities as hierarchical structure in the form of an extended grammar.

Activity grammars have been previously used in literature to model complex activities. Considering complex activities as sentences, authors in [Ivanov and Bobick, 2000] used a stochastic context-free grammar (SCFG) to represent the activities, this was coupled with HMMs at the lowest level to recognise primitive actions and provide them as terminals to the grammar. In our complex activity recognition system, we similarly employ a low-level action detector to be used with our learned activity grammar. More complex activities were tackled by [Moore and Essa, 2002] who introduced more reliable error detection and recovery technique. Other researchers explored using extensions over the grammar by adding other properties to classic linguistic grammar structures [Joo and Chellappa, 2006]. They used an attribute grammar which was augmented with semantic labels and conditions over each rule making modelling of more descriptive activities possible. The authors in [Ryoo and Aggarwal, 2006] extended this work and modelled multi-agent activities but used a description-based approach to model grammar rules as programming language like syntax.

In the approaches mentioned above, rules of the grammar were manually designed and complex temporal relations between activities were not fully utilised. These approaches suffer from highly complex activities which tend to exhibit temporally complex structure such as parallel sub-activities. The authors in [Kitani et al., 2005] attempted to circumvent these problems by learning grammar rules from observation automatically. Other researchers attempted the problem of learning rules of grammar automatically namely [Zhang et al., 2011, Pei et al., 2013]. They used Allen’s temporal logic [Allen, 1983] to capture temporal relations within the right hand side expansions of the grammar rules. The authors in [Zhang et al., 2011] also presented a probabilistic grammar structure whereby each variation of a rule was associated with a proba-

bility which was learned through training. Furthermore, since grammar rules were represented as a set of right hand side symbols and a temporally complex order, the authors in [Zhang et al., 2011] proposed a multi-threaded parsing scheme which allowed for parsing a syntactical structure of activities as a temporally complex stream rather than a sequence. This was different to previous definitions of grammars and parsing which assumed a sequential input since that is the typical behaviour of sentences in language. Though these approaches took great strides in this research area, the generated models were opaque and not human describable. That is to say, a recognised parse tree was not easily understandable. Also these approaches had only been evaluated and shown to be successful on simplistic activities e.g. ‘jumping jacks’, ‘lifting arms’, etc. Moreover, atomic actions were modelled from trajectory motion data of various elements in the video and each primitive described a simple basic motion in terms of point coordinates and motion parameters. This representation is conceptually opaque and does not allow for interpretation of the learnt language at test time. The authors in [Aksoy et al., 2017] have applied hierarchical compositional structures to model complex activities. However, the hierarchies are bounded to fixed number of levels and are partially describable in the training language.

In this thesis our contributions are to learn activity models in the way humans perceive them by eliciting annotations of activity videos at different levels of granularity from multiple annotators. Our system uses semantic matching of activity components to a) unify variable description of the same activity and b) learn the *parent-child* pairing between intervals to generate an activity hierarchy. We also devise a method to learn an extended stochastic grammar representation capable of representing temporally complex parallel activities and parsing of automatically detected low-level actions into their most likely hierarchical interpretations. Multiple annotations are efficiently acquired using platforms such as Amazon Mechanical Turk, CrowdFlower etc. [Nguyen-Dinh et al., 2013]. Many fields of computing now employ crowd sourcing to obtain human intelligent’ responses. Amazon Mechanical Turk (AMT) is a well-known crowd sourcing system that has been used in NLP, HCI and ML researches [Kaisser and Lowe, 2008, Rashtchian et al., 2010, Sorokin and Forsyth, 2008, Heer and Bostock, 2010]. Crowdsourcing offers a large pool of participants which ensures variability in the annotations collected from participants, and sufficient coverage of possible interpretations. Furthermore, re-

sponses are unbiased since respondents have little or no knowledge of the research. This ensures that a wide human perspective of activities can be captured and utilised for model learning. The learned model is therefore capable of being semantically interpreted in the training language. The system is designed to model the human conceptual understanding of the activity and learn individual per-task optimum hierarchies by using a consolidation method to learn the overall model for the activity. The learned model therefore can be semantically interpreted in the training language. We then test our system on natural and complex multi-level activity datasets.

2.4 Summary

In this thesis, we present an approach which differs from other related work in the following aspects. Our simple activity recognition framework makes use of qualitative and quantitative representations to jointly model simple activities and actions. This is a distinguishing aspect since previous methods rely solely on qualitative or quantitative representation and little to no exploration has been performed on a combined representation. Our complex activity recognition approach differs from other approaches in its capability to model an activity using an arbitrary number of levels of granularity. Previous graphical approaches often have limited number of levels of granularity when hierarchically modelling complex activities. Furthermore, through our approach, language phrases are grounded to videos of activities, this grounding is rarely utilised within previous activity recognition methods. Another distinguishing aspect of our hierarchical modelling of complex activities is the generation of explainable and human-interpretable models. Previous state-of-the-art approaches presented models which exhibit deep hierarchical structures, however, the intermediate levels of these hierarchies are often represented by low-level feature spaces such as trajectories or optical flow. In contrast, our method models human activity using natural language labels of activities which enable the model to become explainable and interpretable. Differently from typical methods which use discriminative modelling to model each activity class, our method models activity classes as progressive combination of other sub-activity classes thereby avoiding redundant learning. Finally, our complex activity recognition method tackles the challenging problem of automatic inference of the *parent-child*

relationship amongst different activity classes within a single complex activity video. In past work, this relationship has often been hand-defined.

In this section, we described the most relevant related work to our approaches. We proposed a review of general activity recognition from early work which relied on template-based activity recognition and pixel-based representation to more recent work utilising object-based representation. We then covered on representations used in this thesis for simple activities, namely qualitative spatial and temporal relationship. Related work on complex activities was presented with the use of language like grammar to model intricate hierarchical structures within activities. Finally, we described an overview of our approach and its relation to the related work. In the next chapter, we present our simple activity recognition approach.

Chapter 3

Qualitative and Quantitative Modelling of Simple Activities

3.1 Introduction

In this chapter we present our qualitative and quantitative spatio-temporal representation (QQSTR) for simple activity modelling and recognition. Activities vary largely in complexity both in terms of temporal length and in the number of constituent activities. The goal of our QQSTR system is to model ‘simple’ activities that do not exhibit a deep hierarchy of constituent activities. Examples of such activities include ‘microwaving food’, ‘arranging objects’, ‘picking up’, ‘putting down’ etc. In order to represent activities accurately and completely, we must capture the spatial and temporal interaction changes that occur within the activity. Existing methods have been used to capture interactions within an activity using either qualitative or quantitative methods [Sridhar et al., 2010b, Duckworth et al., 2016b, Behera et al., 2012]. We assert that representing an activity through either capturing the qualitative or quantitative method alone is insufficient to fully capture the intricacies and variations within the activity. A simple example is interaction between the hand and the head of a person during two activities namely: ‘taking a bite’ versus ‘coughing’. In this example, suppose no information on the object is known and only skeleton tracks of the person performing the activity are provided.

In both activities, the set of qualitative description of the interaction between the hand and head is exactly the same: hand *not touching* head, hand *approaching* head, hand *touching* head and finally hand *retracting from* head. There is no notion of the length of time that the hand was in contact with the head. Adding a quantitative feature that models the distribution of Euclidean distances between the head and hand over time and representing this distribution with statistical measures such as mean, median, kurtosis etc. would encode enough additional information on quantitative changes so that a clear distinction between the two activities would become apparent. We present a system that finds a highly discriminating representation of an activity in terms of the optimal combination of qualitative and quantitative features.

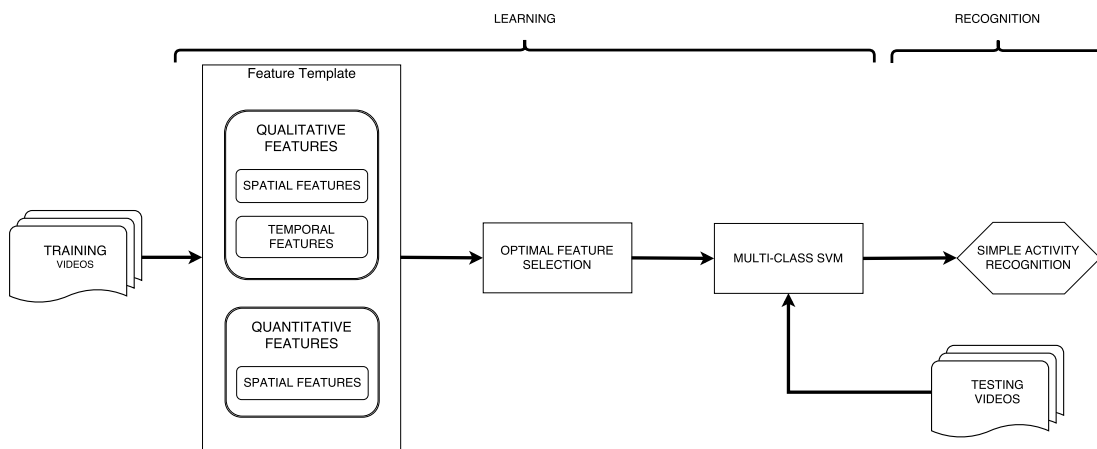


Figure 3.1: A flowchart of the qualitative and quantitative spatio-temporal representations (QQSTR) activity recognition system.

Our qualitative representation comprises of the use of well-established qualitative spatial relations and Allen’s temporal algebra to abstractly capture the interactions occurring within the video in space and in time. Complimentary to this, the feature space is augmented with a quantitative representation which uses common statistical measures to represent the changes in the interactions within the activity. We then combine the two feature spaces to build a combined feature space representation of the activity. Ground-truth training data is then used to automatically prune the feature space to the most representative features by maximizing entropy. A flowchart of this framework is presented in figure 3.1. This figure shows that the approach is split into two sections: learning and recognition. In learning, training videos with object and/or skeleton tracks are provided. These are used to learn a combined multivariate

feature space including qualitative and quantitative features. The feature space is then automatically pruned to produce a feature set that has the highest discriminative power to discern the different classes of activities given the training videos. In recognition, a multi-class SVM is trained to classify previously unseen testing videos. This approach is shown to outperform existing benchmarking methods. We explain these steps in greater detail as well as provide motivation for our choices of representation.

3.2 Qualitative Representation

Human activities are a complex set of motions and interactions of various entities, objects, human joints etc. in an environment [Lavee et al., 2009]. We argue that in order to characterise human activities effectively, qualitative spatio-temporal relations are fundamental in representing the interactions between various entities in an environment. Objects and human agents move freely through space and time, however the human mind tends to represent this motion through qualitatively interesting discrete states called qualitative relations. Most cognitive description of activities in space and time is qualitative in nature; this is evident through observing the natural language descriptions used to describe these activities for example ‘car *far* from garage’, ‘person *moving towards* door’, ‘cup to the *left of* kettle’ etc. [Cohn et al., 2002]. The use of qualitative representation is an attractive aspect since interesting events occur as interactions between entities which can be succinctly represented using qualitative representation. Take an example of an ‘eating apple’ activity. Interesting episodes within this activity can be thought of as ‘picking the apple’, ‘biting the apple’, ‘putting down the apple’ and so on. Notice that all these descriptions are defining interactions between two or more entities within the scene such as the interaction between the ‘hand’ and the ‘apple’ or the ‘hand’, ‘apple’ and the ‘mouth’ etc. A large volume of research exists on various qualitative spatial relationships [Cohn and Hazarika, 2001, Cohn et al., 1997]. In this section, we describe the motivation behind the use of qualitative representation along with our choice of qualitative spatial relationships and features used in our system.

3.2.1 Motivation

Given a stream of quantitative observations obtained from low-level sensor data analysis such as skeleton tracks or object tracks in a video, purely quantitative approaches produce precise and numerical information about different entities in the scene. For example, ‘cup 1 is at a distance of 2.2 meters from the left hand at frame 54’. This precise quantitative information is often noisy due to low-level sensor errors and lacks the natural way in which humans describe activities. When describing environments that are composed of dynamic objects, it is often common to omit precise numerical description such as the speed or relative distance per time-step of specific entities within the environment. Qualitative representations are usually sufficient and offer a more natural description of such environments. An example of the use of a qualitative representation is when capturing the spatial changes occurring within an environment over time. We can broadly divide the motivation for our choice to use qualitative representation in three sections: abstraction of concepts, low-dimensional representation and noise robust representation.

Abstraction of Concepts

An important aspect of any supervised activity learning and recognition system is its ability to draw general concepts about the underlying activity from training data whilst omitting unnecessary or redundant information. This is achieved through modelling qualitative relations between different entities in the scene over time rather than their exact coordinate positions in space over time. Through qualitative representations, the system learns the qualitative description of the motion pattern of entities. For example, a ‘drink’ activity would involve a person lifting a cup, drinking from it and setting it back down. This activity would vary when performed by different human subjects. Some may perform the lifting and setting down motion at different speeds, others may use the left or the right hand to grab the cup. The cup might travel different distances depending on its initial position and the posture/size of the subject. In activity recognition, it is desirable to generalise and represent the underlying activity of ‘drink’ regardless of specific and irrelevant information such as the posture, the speed of the motion of the cup etc. To achieve this goal, in all the above mentioned cases, a qualitative description

aims to produce highly similar representations of the activity. Attempting to compare exact coordinates of hands, cups, head etc. through the motion across multiple subject would be more challenging. We are therefore able to abstract concepts much more effectively using a qualitative representation.

Low-dimensional representation

Another motivation of using qualitative representation is that such a representation is often low-dimensional while encoding enough discriminatory information to discern many human activities [Chen et al., 2015]. Low-dimensionality is an attractive feature for any representation since it relieves computational burden and promotes rapid processing of input test data. Abstracting away from minute and unnecessary details of an activity allows for capturing the most important aspects of an activity through the use of qualitative spatio-temporal relations which can be encoded in a much smaller dimensional space compared to storing full details of all moving entities through time. For example, the observed space when performing activity recognition is a continuous environment of moving entities such as body poses of people and tracks of objects. Using qualitative spatial representations (QSRs), the continuous tracks of objects and body poses can be mapped into a discrete set of qualitative symbols representing qualitatively interesting events within the activity. For example, qualitative symbols might be comprised of ‘moving towards’, ‘part of’, ‘departing’ etc. These symbols are capable of describing many elements of the scene such as topology, distance, orientation etc.

Depending on the calculi used, the qualitative features generated from these relations are of much lower-dimensionality. Further in this chapter, we show that choosing a very simple and concise calculus is sufficient to encode much of the discriminatory information about simple everyday human activities. In our work, we also employ a feature selection mechanism to identify and select most discriminative features to further lower the dimensionality of our feature space. Much work in the literature has used qualitative representation mainly motivated by its low-dimensional representation. [Sridhar et al., 2010a, Sridhar et al., 2010b] uses QSRs to build three-layered activity graphs representing interactions within objects in terms of qualitative relations. Temporal orderings of these relations are captured by temporal nodes between each pair of qualitative relations. [Young and Hawes, 2015] similarly leverage the low-dimensional

and high generality features of qualitative representation to ‘learn by observation’ for an intelligent agent replicate behaviour in a complex environment. [Behera et al., 2012] builds a qualitative egocentric representation of the scene for either navigation or activity recognition.

Noise robust representation

Qualitative spatial relations (QSRs) have been frequently used in the past to discretise space into qualitative descriptions. Consider the activity ‘approaching’ where the distance between two entities is gradually reducing. Quantitatively, we can represent this activity as a set of distance values between the entities per time step. This set of distances, however, provides a less abstracted representation of approaching since it is too specific to one example and cannot be generalised. In order to abstract away from such a representation, categorising distance into distinct relations such as: ‘far’ followed by ‘near’ to capture ‘approaching’ provides a much higher level of abstraction describing that activity, see figure 3.2. Conversely, the ‘approaching’ action can itself be introduced as a qualitative state defining the relative trajectory of an object [Van de Weghe et al., 2005]. Notice that noisy observations observed in input data such as radically varied position of an entity, would have a dire effect on the quantitative distance value whereas the qualitative representation is likely absorbing such turbulence in observations.

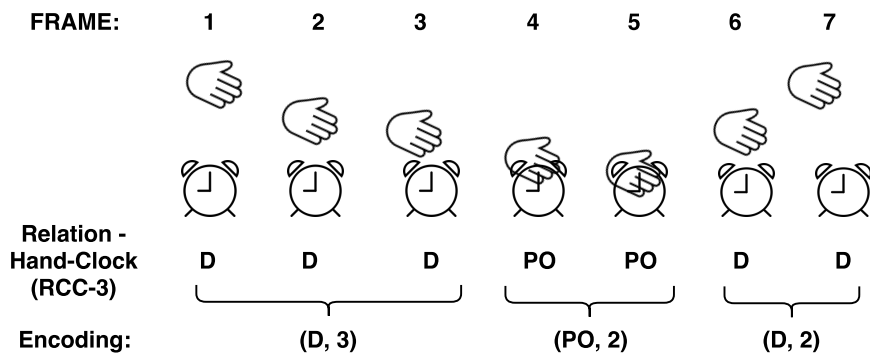


Figure 3.2: Simple example of a ‘turning off the alarm’ activity shown in terms of positional observations of a hand and a clock entity at each frame. The corresponding RCC-3 relational string between the entities is also shown. A simple run-length encoding of the relational string also presented. The relations in the string present qualitative relations between the entities where ‘D’ refers to discrete and ‘PO’ refers to partially overlapping.

3.2.2 Representation

In this section, we describe the qualitative spatial relations used in our work. Many different calculi exist to model different aspects of space such as topology, direction, relative speeds etc. [Cohn and Hazarika, 2001]. The use of different types of QSRs to model space, have been explored in literature for example topology [Sridhar et al., 2010a], direction [Duckworth et al., 2016b], relative trajectories or direction [Ferynhough et al., 1998, Duckworth et al., 2016b], relative sizes [Gerevini and Renz, 2002] and relative speed [Delafontaine et al., 2011] etc. The task of choosing the best QSRs or an optimum combination of different QSRs is non-trivial and is often driven by domain knowledge and requirements [Wöflf and Westphal, 2009]. In this thesis, real video data comprises of skeleton and object tracks embedded in two-dimensional space extracted from RGBD images. Since objects in video data are commonly represented by their minimum bounding rectangles (MBR) rather than exact shapes, we decide to use a simplified spatial representation to represent the topology using RCC-3 as shown in figure 3.3b. This representation is shown to have been used with much success in the previous literature to represent specifically video data of activities in 2D space captured through RGB images [Sridhar et al., 2010a]. Though human activities are naturally performed in 3D space, using a 2D projection of these activities in form of images from video is often a good approximation of the overall activity. Furthermore, the noise and jitter caused due to inaccurate object and skeleton detectors is somewhat suppressed in the 2D projection thus highlighting true interactions within the activity more prominently. This helps to model the activity with higher accuracy. Finally, our chosen calculus RCC-3 is natively designed for 2D projections, for example the ‘part-of’ relationship between two disjoint objects can only hold in 2D projections of the scene. We describe this representation in further detail in the next section.

Whilst topology is an important aspect to capture in video data, modelling the trajectories of objects within the scene is also a discriminating aspect of many activities. We employ a simplified transformation of the QTC calculus [Van de Weghe et al., 2006] to succinctly capture the global trend of each entity’s relative trajectory within the scene. Both these calculi are further motivated and discussed next.

Region Connection Calculus (RCC-3)

We make use of three simple spatial relations in our representation as shown in equation 3.1;

$$\mathfrak{R} = \{D(\text{Discrete}), PO(\text{Partially overlap}), P(\text{Part Of})\} \quad (3.1)$$

These relations are a simpler version of the RCC-5 calculi proposed in [Randell et al., 1992]. Figure 3.3 shows the conceptual neighbourhood graphs for the RCC-5 and the derived RCC-3 relations. Note that the ‘PO’ relationship in physical object is rare since objects do not often naturally merge into each other. But since we are projecting 3D space into 2D image planes, the ‘PO’ and ‘P’ relations become commonly observable. Two simplification in the RCC-5 calculi are made to derive RCC-3:

1. The equivalence ‘EQ’ relation is removed. In our data, two entities are unlikely to have same sizes resulting an EQ relation. Moreover, there is a low-probability of entity arrangements in an EQ relation for an extended period of time in a highly dynamic environment like a video of a daily living activity. We therefore omit this relationship in order to avoid unnecessary dimensionality trade-off in our final feature space.
2. The ‘PPI’ and ‘PP’ relationships are inverses of one another; in our feature representation, this duality is implicitly captured. Therefore, these two relationships are merged into just ‘P’. We describe the precise feature space used later in this chapter.

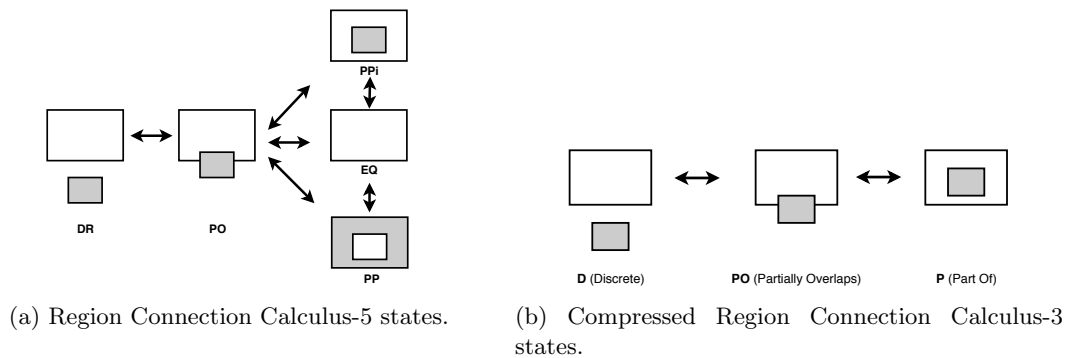


Figure 3.3: Conceptual neighbourhood graphs of RCC-5 and the simplified RCC-3 qualitative spatial relationships.

The choice of the RCC-3 QSRs is motivated through two examples 3.2 and 3.6. The example 3.2 shows a sequence of trajectories of a hand and a clock for a sample ‘turning off the alarm’ activity. It is obvious that most interesting events within this activity tend to happen at the interaction between the hand and the clock. We can represent the entire sequence as a compressed set of relational changes as shown in the compressed column. Using the RCC-3 representation, it is apparent that interesting interaction points in time are well captured. Later in the chapter, we present the ‘receive a call’ activity in example 3.6. The first part of this example represents trajectories between three entities: a phone, hand and head of a person. Notice that all entities hold a discrete (D) relation for the first 3 frames since none of the entities are in contact with each other contact. The relation changes to part-of (PO) during partially overlapping contact between entities in 2-D space and finally reverts to (D) for the remaining sequence. We process the string of relations through a run-length encoder and omit the number of repetitions in the resulting encoding. Run-length encoding is a form of lossless compression of a sequence of data symbols where repeating symbols are represented as a single instance of the symbol and a number of repetitions of that symbol. Details are described later in this chapter. The resultant encoding represents a sequence of relations without consecutive repetitions (D-PO-P). This sequence sufficiently removes redundant and uninteresting parts of the relational sequence and captures only the interesting changes in the relations. This representation is able to model any type of activity where interactions occur between entities. Entities may refer to tracked human skeletal joints and/or objects. Researchers have previously modelled various types of activities successfully using this representation ranging from human activities such as ‘eating’, ‘cleaning’, ‘setting up a table’ etc.[[Duckworth et al., 2016b](#), [Dubba et al., 2015](#)] to non-human activities such as aircraft activities on an airport [[Sridhar et al., 2011](#)].

While our choice of QSRs here is RCC-3, it should be noted that our framework is general and is able to adopt any other calculi e.g. distance, relative speed etc. It is also possible to automatically learn an optimal set of QSRs [[Galata et al., 2002](#)] for representation from training data.

Simplified Qualitative Trajectory Calculus (SQTC)

Using the region connection calculus presented in the previous section, motion relationships between objects are not fully encoded. Some notion of their motion within the wider environment can still be drawn using RCC. It can be argued that implicit motion patterns through recording state changes are still captured. For example, consider a state change from discrete (D) to partially overlap (PO) between two objects, as shown in figure 3.2. This change implicitly implies that two objects are moving towards each other. Whilst this is true in the case when an *interaction* between two objects occurs, the same conclusion cannot be drawn in cases of no interaction i.e. objects are in a constant discrete (D) state. In other words, motion of disconnected objects is not represented using the RCC representation. [Weghe et al., 2004] asserts that describing this motion qualitatively is highly desirable in modelling activities in many domains. In a typical everyday activity, objects are mostly observed in the discrete state with occasional interactions. This further imposes the need for describing the motion of objects in a more representative qualitative manner in order to model the activity with higher detail.

In our work, we take inspiration from the calculus proposed by [Weghe et al., 2004], namely QTC basic, to produce a simpler calculus that sufficiently captures the motion within the domain of activities covered in this thesis. In the classic QTCb calculus, at each time-step, the movement between any two objects (O1, O2), is qualitatively represented as a two-tuple, for example (+,0), using the following rules:

1. O1 is moving towards O2, represented by (+) for O1,
2. O1 is moving away from O2, represented by (-) for O1,
3. O1 is neither moving towards or away from O2, represented by (0) for O1.

In our work, we propose a simpler calculus which represents the relative motion between each pair of objects using only one-tuple symbol rather than two-tuple symbol as is used within the traditional QTCb. This can be seen in the transformed conceptual neighbourhood diagram in figure 3.4. Conceptually, rather than representing the relations of an object relative to another object, we represent the relation between both objects globally using a single symbol. We can now define the relations as the following transformed relations:

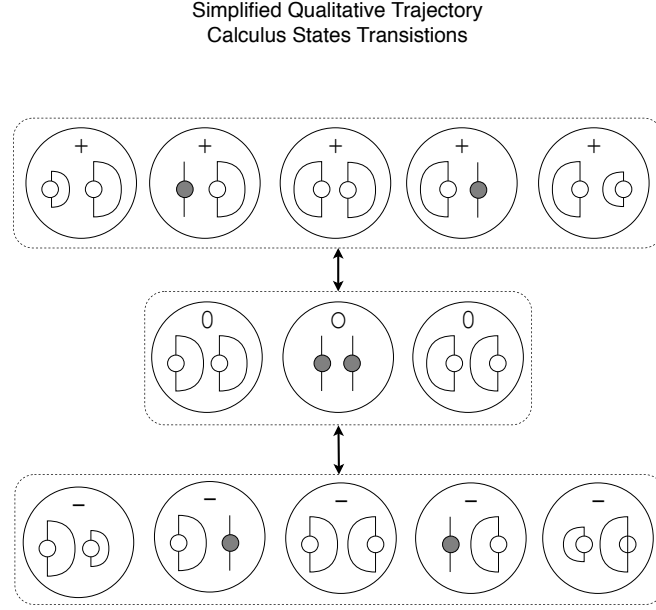


Figure 3.4: This illustration shows all possible states of motion between two objects in our 2-D representation. Atop each state, the corresponding SQTC relation is shown. The arrows demonstrate the transitions allowed between relations. Solid circles represent static objects whereas hollow circle represent moving objects. The semi-circle represents the range of motion and the size of the semi-circles represents velocity.

1. O1 and O2 are *approaching*, represented by (+),
2. O1 and O2 are *departing*, represented by (-),
3. O1 and O2 are *stable*, represented by (0).

Note that the new relations are independent of object order i.e. the relations represent whether the two objects are moving away from each other, closer to each other or exhibit a stable state. For example, an approaching relation between two objects O1 and O2 may occur in case when O1 is stationary and O2 is moving towards O1, O2 is stationary and O1 is moving towards O2 or O1 and O2 are both moving towards each other. All these three states are now represented by the approaching symbol (+). In a similar way, each of the *two* motion patterns are encoded with different sets of QTC symbols ($\{+,0\},\{-,+\},\{0,0\}$) hence creating a simpler calculus. Figure 3.4 shows all cases of motion of two objects and the corresponding encoded symbol using our proposed amendment to the calculus. The size of the semi-circle conceptually represents the speed at which the object is moving, therefore, objects moving in

the same direction are further split into three-state where first object is faster than the second, the second faster than the first and when both are at the same speed (top-left and bottom-right nodes). The remaining nodes are simply transformed to our proposed calculus symbols of approaching (+), departing (-) or stable (0). We refer to this transformed calculus as simple qualitative trajectory calculus (SQTC).

To motivate the simplification choices made on the traditional QTC calculi, we now present three reasons for our decision.

1. The primary goal of any feature engineering mechanism is to find the right balance between dimensionality and amount of distinct knowledge represented. Using a single symbol and discretising the trajectory patterns between a pair of objects into just three states, as opposed to nine states each represented with two-tuple symbols, helps towards representing the motion patterns in a lower-dimensional feature space. We further describe the precise feature representation in the next section.
2. Sufficiently capture enough knowledge about the motion patterns of objects to discriminatively represent the target set of activities. Our aim is to build accurate and effective models of simple activities that span short time frames. As explained before, these activities are commonly composed of simple actions lasting few seconds, such as ‘lifting’, ‘pushing’, ‘picking’ etc. Within such short time frames, it is reasonable to assume that most observed motion patterns amongst entities would be monotonic in nature i.e. they would be increasing or decreasing. As activities get more complex, we start to notice motion patterns exhibiting changes in direction, speed, rotation etc. Whilst in simple actions, it is rare to witness such sporadic changes. We therefore assert that using traditional QTC would create an over-representation whilst the proposed simplified calculus captures a more abstracted representation of the objects motion patterns.
3. This reason directly arises from observing sample activities from our datasets. One such class of activities are mirror activities. Examples of mirror activities include ‘pushing’ and ‘pulling’, ‘stacking’ and ‘unstacking’ etc., see figure 3.5. These activities tend to be very similarly represented in both qualitative and quantitative feature proposed. Mirror activities tend to generate a similar count of qualitative relations when using the RCC

calculus at the same time the distributions of distances between joints and objects, covered in next sections, is direction independent. In order to discriminate between such activities successfully, some notion of direction need to be introduced within the feature space. Using our simplified calculi, we are able to sufficiently represent whether objects within a scene are moving apart from each other (unstacking) or moving towards each other (stacking), as shown in figure 3.5, using a minimal set of symbols.

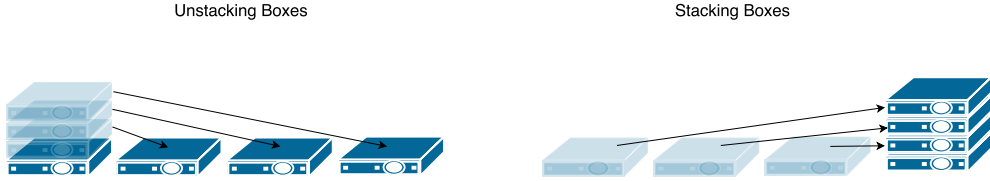


Figure 3.5: Example of an ‘unstacking boxes’ and ‘stacking boxes’ activity illustrating the motivation for using the SQTC calculus to discriminate amongst them.

3.2.3 Qualitative Spatial Features using Region Connection Calculus

As mentioned before, we make use of the well-established Region Connection Calculus to represent topology within our system. This is done by defining a representation that uses RCC relationships to encode the elements. In this section, we describe how the proposed representation is utilised to extract meaningful features and generate a feature space from training videos of activities.

Suppose there is an activity video comprising N_F frames, containing a set \mathcal{E} of entities, referring to objects and human subject skeletal joints. We compute the Region Connection Calculus-3 (RCC-3) QSRs between each pair of entities in \mathcal{E} for every frame and build a spatial-relations matrix called \mathbf{M} . The dimensionality of \mathbf{M} is $\frac{|\mathcal{E}|(|\mathcal{E}| - 1)}{2} \times N_F$ denoting the number of entity pairs per number of frames. An example of the \mathbf{M} matrix for a sample activity with 3 entities is shown in figure 3.6.

Given matrix \mathbf{M} , an i^{th} row of this matrix can be denoted by $\mathbf{M}_{i,*}$. Qualitatively interesting event occur at every position j where $\mathbf{M}_{i,j} \neq \mathbf{M}_{i,j+1} \forall j \in \{1 \dots N_F\}$ [Cohn and Hazarika, 2001]. Between these *interesting positions*, a single spatial relation holds for multiple consecutive frames. This intermediate period is considered as qualitatively uninteresting and can therefore

be compressed and represented by a single instance of the repeating spatial relationship. Run-length encoding is a standard simple form of lossless compression in which a sequence of data with consecutively repeating data elements is replaced with a single data element and a count of the repetition. For example, a data input string ‘AAABBBAAAACDDD’ will be encoded as ‘A3B2A4C1D3’. We represent this encoding as a 2-tuple (\mathbf{d}, \mathbf{c}) denoting the data elements vector \mathbf{d} and their corresponding counts vector \mathbf{c} . In the example above, the encoding will have $\mathbf{d} = (A, B, A, C, D)$ and $\mathbf{c} = (3, 2, 4, 1)$. We perform this encoding, using two different methods, to remove continuous repetition of relations and produce a maximally compressed representation of the changes in relations between entities.

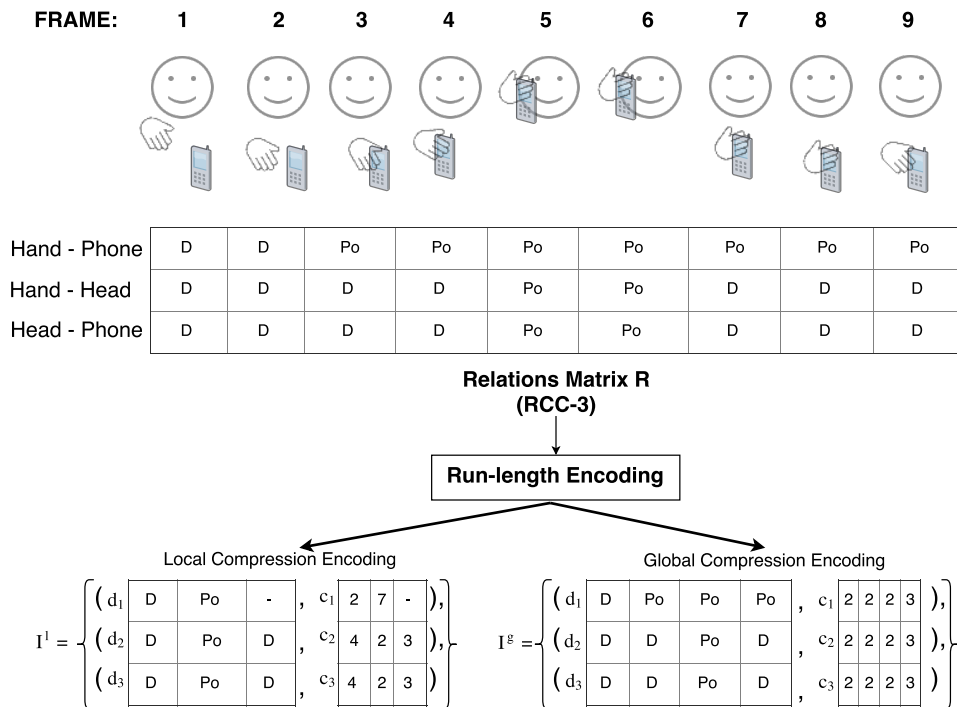


Figure 3.6: An extended example of a ‘receiving a call’ activity represented at each time step. The corresponding relational matrix \mathbf{M} generated for each entity pair for every frame is shown. The two compression encoding mechanisms are presented. Global encoding represents a holistic picture of activity compared to local compression. However, local compression achieves a higher degree of compression.

1. Local compression

In local compression, we encode the input relation strings by performing a run-length

encoding over each row i of the \mathbf{M} matrix independent from other rows. This has been previously done by [FERNYHOUGH et al., 1998]. For each pair of neighbouring data elements $(\mathbf{M}_{i,j}, \mathbf{M}_{i,j+1}) \forall j$ in the input stream, elements are compressed into a run-length encoding if the elements are equal $\mathbf{M}_{i,j} = \mathbf{M}_{i,j+1}$. Intuitively, local compression encodes an input relations string for an entity pair by compressing repeating relations while only considering the surrounding relations of that entity pair (row) only. Other entity pairs (other rows of matrix \mathbf{M}) may have ongoing interactions, i.e. changing relations, during this time but these would not be considered when compressing relations for that entity pair. Each row i of the \mathbf{M} matrix would be encoded using this process in local compression to produce an total encoding of the entire video sequence $I^l = \{(\mathbf{d}, \mathbf{c})_1, (\mathbf{d}, \mathbf{c})_2, \dots, (\mathbf{d}, \mathbf{c})_{N_E}\}$ where N_E is the total number of entity pairs in the activity or the number of rows in the \mathbf{M} matrix. Note that the resulting set of locally compressed intervals I^l might have varying lengths of encoded relational strings across different encoded rows or different i . This is because there might be more compressible areas in the relation string of one entity pair than another. Example 3.6 shows a sample encoding using local compression. Note that the encoding compresses relations from each entity pair maximally and independent of other entity pairs. This is obvious when observing the interaction between the hand and phone entities, frames 1 till 2 are compressed and encoded as just ‘D’, then frames 3 through 9 are collapsed and encoded as ‘PO’. Other entity pairs such as hand and head are interacting during frames 3 through 9 but this is not considered when compressing relations.

2. Global compression

Global compression achieves the same encoding by performing a run length encoding over each row i of the \mathbf{M} matrix however with a slightly varied comparison condition between consecutive elements of the relational string. Differently to local compression, the comparison of two data elements of a relational string depends on the same relation holding across all entity pairs $i \in \{1 \dots N_E\}$ rather than just the pair under consideration i . Formally, given two consecutive relation elements $(\mathbf{M}_{i,j}, \mathbf{M}_{i,j+1}) \forall j$, elements are compressed only if elements across all entity pairs (all columns) are equal $\mathbf{M}_{*,j} = \mathbf{M}_{*,j+1}$. Intuitively, considering entire columns of matrix \mathbf{M} for equality is equivalent to the assertion that

there needs to be no relational changes across all entity pair. That is to say a static scene where no interactions are taking place between any entity pair must be observed in order to suppress that part of the activity. This is different to local compression where the dormancy requirement is only applicable to a single entity pair, whereas the rest of the entity pairs are relaxed from this requirement and are free to interact. We demonstrate that in our domain of activities, global compression provides a holistic picture of the scene and captures more of the subtlety of activities especially when there are parallel occurring activities, for example drinking with one hand whilst writing with the other. Similar to local compression, each row i is encoded to produce the overall encoded set of episodes $I^g = \{(\mathbf{d}, \mathbf{c})_1, (\mathbf{d}, \mathbf{c})_2, \dots, (\mathbf{d}, \mathbf{c})_{N_E}\}$.

One drawback of this approach is that the resulting representation will have less compression than local compression however this will not affect the dimensionality of our feature space as shown later in this chapter. Lastly, different from local compression, note that the resulting encoding I^g will all have the same lengths of encoded relations since compression will only occur if repeating relations exist across all entity pairs. We experiment with both forms of compression and compare them empirically in the experiments section.

Example 3.6 illustrates the global compression encoded matrix. It is shown that compression is performed in locations where all consecutive columns of the relations matrix hold. The relations between hand and phone being PO from frame 3 through 9 will now be compressed from 3 through 4, 5 through 6 and 7 through 9 since the other relations in the rest of the entity pairs in those frames also hold the same relations. This results in the same length encoding for all rows in global compression and varying length encoding for different rows in local compression. During implementation, the global compression method just stores one row of the matrix of counts (\mathbf{c}_1 , \mathbf{c}_2 or \mathbf{c}_3) since all rows represent the same count values, in effect producing a more efficient encoding.

Feature Extraction

The feature space for our method is designed to capture the qualitatively interesting events that take place during a video. Our goal is to construct a fixed length feature space that is

able encode instances of activities represented using the RCC-3 representation described. We draw inspiration from the natural language processing and computer vision literature to use a bounded version of the popular bag-of-words approach [Sivic and Zisserman, 2003] to build our feature space.

A bag-of-words model in natural language processing treats text as a set of its constituting words. This approach disregards grammar, word order and other expressive details of the text but rather captures multiplicity of words in the documents. Training documents can be used to extract a fixed length vocabulary (codebook). Each document can then be represented as a histogram of its constituting words over the pre-learned vocabulary [Joachims, 1998, McCallum et al., 1998]. In the field of computer vision [Zhang et al., 2010], similarly, images are first broken up into image patches and expressive image patches are mined which become the bag-of-words (codebook). Training images are then represented as a count of these patches. Any image can then be represented as such a set of visual words counts with a fixed dimensionality.

Similar to the bags-of-words approach in computer vision, our bag-of-words approach represents each activity as a count of the qualitatively interesting relational sequences. Recall that the notion of ‘qualitatively interesting’ events are defined as interactions or relational changes amongst entities within the activity. As in text analysis, a vocabulary may comprise individual words and/or multiple words for example ‘Eat’ or ‘Eat sandwich’. Multiple words or sequences of words provide additional information about the context in which the word is expected to appear in a document. This is analogous to capturing a series of qualitative relational changes along with single changes to fully represent an activity. We therefore build our codebook/vocabulary from all possible qualitatively interesting sequences up to some length χ between all pairs of entities in the scene for example ‘D’ to ‘PO’ to ‘D’ (D,PO,D) for hand-cup or ‘D’ to ‘D’ (D,D) for head-hand etc. An activity compressed relational matrix can then be represented as a histogram of the qualitatively interesting primitives or relations sequences over this vocabulary.

Formally, the vocabulary is constructed from the set of RCC relations used within the framework. Recall that we use the RCC-3 calculi to represent our videos with the following relations $\mathfrak{R} = \{D, PO, P\}$. In order to build our vocabulary, We then take all permutations of the elements of set \mathfrak{R} of length $i \in \{1 \dots \chi\}$ while allowing repetitions. These permutations represent the different qualitatively interesting sequences of relations up to length χ . Permutations of

different lengths are concatenated together to form V_l as shown in equation 3.2.

$$V_l = \bigcup_{i=1}^{\chi} \mathfrak{R} P_i \quad (3.2)$$

Recall that we have multiple entities in the scene and each codeword needs to be repeated for all pairs of entities. We therefore produce the vocabulary, denoted by F_1 , as a union of all the permutations of relational changes for all entity pairs $i \in \{1 \dots N_E\}$ as detailed in equation 3.3.

$$F_1 = \bigcup_{i=1}^{N_E} V_l^i \quad (3.3)$$

This process produces $\sum_{i=1}^{\chi} |\mathfrak{R}|^i$ codewords. In our framework the selected value for χ is 4. We are able to build a qualitative feature space encoding these relations called F_1 with dimensionality $120 \times N_E$ as there are N_E entity pairs. The resulting feature set F_1 is shown in example 3.7. We are able to encode any activity with varying number of frames and entities, and therefore varying number of interactions, in a fixed length vector as defined above.

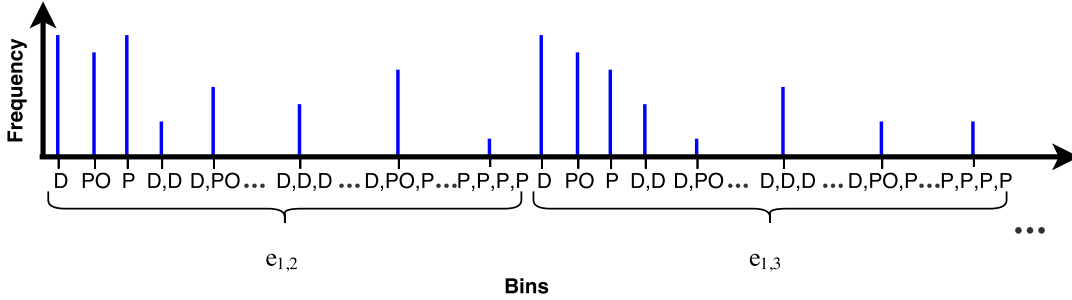


Figure 3.7: Feature space representing a vocabulary of codewords represented as a histogram. Note that codewords are represented by the set of relational changes as bins in this histogram. That is to say, each bin denotes a single codeword which may be ‘D’, ‘D,PO,D’ or ‘P,P,P,P’ and so on. Since there are N_E entity pairs within an activity video, the bins set is repeated N_E times to capture the relational changes for each entity pair. For example for entities 1 and 2, the corresponding bin set is denoted by $e_{1,2}$ in this figure.

3.2.4 Qualitative Spatial Features using Qualitative Trajectory Calculus

In order to build features to encode the SQTC representation, we compute distances between every pair of entities at each frame. The set of distances are globally discretised into either one of the three relations of the SQTC. Since the relations in this calculus captures whether two entities are approaching, departing or stable, a simple observation of the evolution of the distances between the entities throughout the activity video would indicate the qualitative relation of SQTC between the objects. Therefore we first compute the Euclidean distances $dist$ between every pair of entities which are represented by their mid-point (x,y) in the 2-D image plane.

These distances are computed for all N_F frames of the video between all N_E pairs of entities to build a distances matrix with dimensionality $\mathbf{X}^{N_E \times N_F}$. Each row of the \mathbf{X} matrix is then used to estimate the global direction of motion between those pairs of entities in the scene. Consider a single row vector \mathbf{x} of matrix \mathbf{X} corresponding to distances between any two entities $(e_i, e_j) \in \mathcal{E}$. Fitting a straight line through values of \mathbf{x} would indicate the general trend of motion between two entities e_i and e_j as being either increasing, decreasing or non-changing. This would indicate if the two entities are approaching, departing or stable. We can perform a simple linear regression [Seber and Lee, 2012], to determine the optimum line $y' = \beta_0 + \beta_1 x$ with gradient β_1 and y-intercept β_0 that best fits the data points defined by $\{(x_i, y_i), i = \{1 \dots N_F\}\}$ where x represents the frame number in vector \mathbf{x} and y represents the corresponding distance value between a pair of entities. Using a least squares approach [Leon, 1980], values for β_0 and β_1 can be analytically computed. The resulting optimum gradient β_1 is then mapped to a qualitative symbol representing either approaching, departing or stable using the following policy:

- *approaching* (+) : β_1 is positive $\wedge \beta_1 > \hat{T}$
- *departing* (-) : β_1 is negative $\wedge \beta_1 < -\hat{T}$
- *stable* (0) : $-\hat{T} \leq \beta_1 \leq \hat{T}$

A small threshold \hat{T} provides a minor range of gradient values around zero for the qualitative

stable (0) state to occur. This is done to account for noisy detection causing small jittery and sporadic positional changes through frames. This results in non-zero distance changes whilst the overall trend of the distances between entities remains, for example monotonic. In cases where entities within an activity exhibit near equal approaching and departing behaviour, the resultant gradient would encode this motion as stable, however, other instances of the same activity would yield a similar encoding and therefore consistent representation is achieved even though the actual motion is encoded simplistically as stable. Features extracted from this method are referred to as F_2 .

3.2.5 Qualitative Temporal Features

Qualitative spatial features described in the previous sections, discretise continuous (binary) spatial relations into a smaller set of pair-wise disjoint qualitative spatial relations. Similarly, in this section, we describe qualitative temporal features which discretise the time domain. Using the representation described so far, result in an abstract representation of activities where, though important aspects are captured, other necessary information related to time between intervals is lacking. Some notion of time is captured through the use of *n-gram* like vocabulary generation. The codewords (sequences of relational changes) implicitly capture some order of activities, however, no information on the duration of time for which the relationships hold, is encoded. The philosophical analysis presented by [Broad, 1938] puts forth the important characteristics of representing time. These are 1) temporal relation, 2) duration and 3) transitive aspects of temporal facts. We construct a feature space that captures a durational aspect by encoding some notion of the relative lengths of time that any single relation holds compared to the next relation in a relational string. For example, consider activities such as ‘picking an apple’ and ‘touching an apple’, the spatial relations introduced thus far between the hand and the apple will probably be identical i.e. the relational string is likely to be ‘D-PO-D’ between the hand and the apple in both cases. In order to discern between the two activities, we propose temporal features which encode the relative duration for which the spatial relations hold within the activity. In the above example, these durational features would encode the relative lengths of relations ‘D-PO’ and relations ‘PO-D’. In the ‘touching an apple’ activity, these relative

lengths of PO with respect to D are expected to be much shorter than in the ‘picking an apple’ activity thereby discerning the two activities.

We adopt the Allen’s Interval algebra (IA) in our later work when modelling complex activities in chapter 4. However, given the current domain of simple activities, it is noted that such activities often comprise of relatively sequential streams of events and intervals that occur one after the other. In our representation we focus on capturing the duration aspect of time by using a slightly different algebra than the Allen’s interval algebra to represent the temporal information in our domain.

Our calculus is somewhat similar to the INDU calculus discussed in chapter 2. We propose a calculus which captures the durational aspect of interval by observing the relative temporal lengths of the intervals. We assert that this is a better descriptor of temporal knowledge in simple activities since high discrimination is noticed by the durations of continuous spatial relations across an activity.

Formally speaking, recall that given a sample \mathbf{M} matrix of spatial relations between N_E pairs of entities for every frame in a video (N_F), a run-length encoding of each row of \mathbf{M} is performed using either local or global compression. Given the output from the encoding, I defines a set of episodes as $I = \{(\mathbf{d}, \mathbf{c})_1, (\mathbf{d}, \mathbf{c})_2, \dots, (\mathbf{d}, \mathbf{c})_{N_E}\}$ where \mathbf{d} and \mathbf{c} vectors represents the compressed relations and the corresponding counts of repetition of those relations. The counts in the \mathbf{c} vector represent the duration that the corresponding relations in \mathbf{d} held for before a different relationship was observed. Let us consider a single entity pair i and the corresponding encoding element $(\mathbf{d}, \mathbf{c})_i \in I$. We encode duration between every set of neighbouring intervals or repeated relations. Given elements of vector $\mathbf{d} = \{d_1, d_2, \dots, d_{|\mathbf{d}|}\}$ of relationships and corresponding vector $\mathbf{c} = \{c_1, c_2, \dots, c_{|\mathbf{d}|}\}$ of counts, we compute the relative durations between each pair of neighbouring elements of \mathbf{d} to generate a ratios vector $\mathbf{r} = \{r_1, r_2, \dots, r_{|\mathbf{d}|-1}\}$ as shown in equation 3.4.

$$\mathbf{r} = \frac{c_i}{c_{i-1}} \forall i \in \{2, \dots, |\mathbf{d}|\} \quad (3.4)$$

Intuitively this ratio of the counts between each neighbouring element represents the duration information between two spatial relations and provides a notion of quantitative length

between the spatial relations. In order to represent these quantitative durations \mathbf{r} into qualitative symbols, we perform a K-means clustering on all the ratios \mathbf{r} generated across the set of all training videos. Using the datasets available to us, in practice we found that discretisation of the continuous space of ratio values can be sufficiently captured with three clusters, represented in figure 3.8. As seen in figure 3.8, the clusters give a qualitative notion of relative duration as either being short, equal and long. These are represented as $\mathfrak{C} = \{\text{short, equal, long}\}$. This verifies our approach with the established INDU calculus where relations from Allen’s algebra are extended to a similar three qualitative duration variant $\{<, =, >\}$. Given a clustered model of durations denoted by $f : \mathbb{R} \mapsto \mathfrak{C}$, we classify each element $r \in \mathbf{r}$ using the clustered model as detailed in equation 3.5 to produce a qualitative representation of ratios $\hat{\mathbf{r}}$.

$$\hat{r} = \arg \min_i \text{dist}(K_i^c, r) \quad (3.5)$$

$\text{dist}(K_i^c, r)$ specifies the Euclidean distance between cluster centroid K_i^c and ratio r . \hat{r} is the cluster centroid that is closest to the ratio r . Through using clustered prototypes to classify relations we allow for a slightly fuzzy comparison of interval boundaries. For example, a set of intervals with ratios 0.9 instead of 1 is highly probable to be classified as equal duration as it likely fall close to the ‘equal’ cluster centroid. Further implications of this approach are discussed in the experiments chapter.

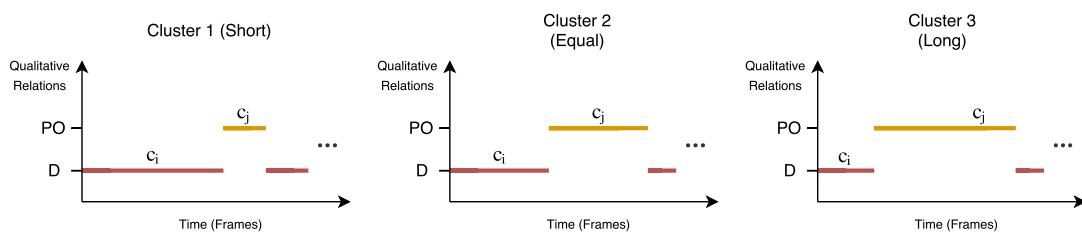


Figure 3.8: Three learned clusters shown here represent the three qualitative relations which describe temporal knowledge. Note that these closely resemble the three extensions offered by the INDU calculus over symbols of the Allen’s interval algebra. The three clusters are labelled as either short, equal or long, reflecting the relative lengths of the two intervals. An example of a pair of intervals classified into each of these three clusters is presented.

Feature extraction

To extract features from the above mentioned representation, we employ the same approach of generating an exhaustive vocabulary which represents all possible permutations of RCC relation changes of exactly length 2, and extending each relational change (codeword) by creating three additional codewords augmented with the duration clusters short, equal, long. Given a set of RCC relations $\mathfrak{R} = \{D, PO, P\}$, all permutations of the elements of set \mathfrak{R} of length 2 with repetition are computed. Since our temporal relations only hold between 2 intervals or relations, we only need to generate permutations of length 2, for example ‘D-PO’, ‘PO-P’, ‘D-P’ etc. Each element of the permutation set V_l is replicated into three elements representing each of the temporal relation clusters. For example, the ‘D-PO’ would be extended to ‘D-PO_{small}’, ‘D-PO_{equal}’ and D-PO_{long} representing the three possible durational relations. Finally, the permutation set needs to be repeated for all pairs of entities N_E . A vocabulary is generated that concatenates all permutations of relational changes with durations for all pairs of entities as detailed in equation 3.6.

$$F_3 = \bigcup_{i=1}^{N_E} V_l^i \quad (3.6)$$

The total number of codewords in the dictionary F_3 are $N_E \times |V_l| \times |\mathfrak{C}|$ where N_E is the number of all pairwise combination of objects, $|V_l|$ is the cardinality of the set of all relational changes and $|\mathfrak{C}|$ is the total number of durational relations. An activity is then represented as a fixed length histogram of its durational relations over this vocabulary. An example histogram is presented in figure 3.9 showing a sample part of the codewords in the bag of words as a histogram feature.

Let us justify the choice of this representation using an example of two common activities. Consider the following two activities: ‘playing a game on phone’ and ‘touching the phone’. The entities in this scenario are phone and hand. Firstly, we develop the qualitative relations between the hand and the phone. In ‘playing a game on phone’ scenario, our observed relational string might look like: (D,D,D,D,D,PO,PO,PO,PO,PO,...,PO,PO,PO,D,D,D,D). This generates from a scenario where a person picks up the phone (D to PO between hand and phone), holds the phone for a while (PO holds between hand and phone) and puts the phone down (PO

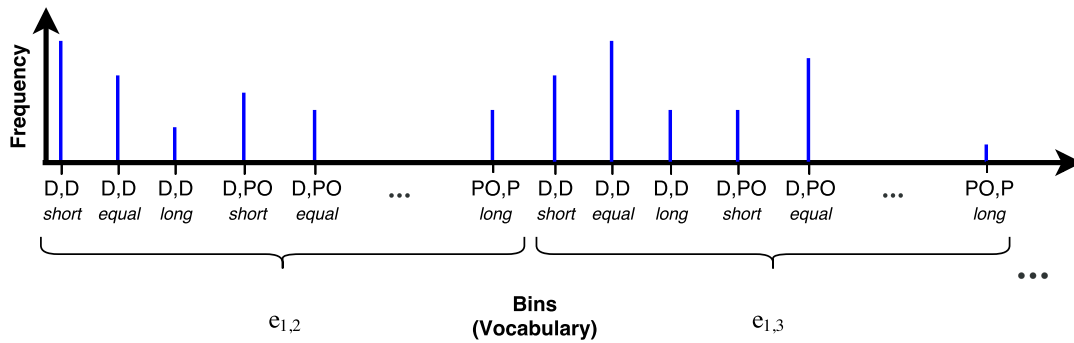


Figure 3.9: The feature space F_3 represented as a histogram. The bins of the histogram correspond to the codewords representing the counts of specific durational relations held between two intervals. The bins are repeated for every entity pair in the activity. Note that another strength of our framework is that it does not utilise exact entity labels, which are often noisy and difficult to recognise for computer vision systems.

to D between hand and phone). In the scenario of ‘touching the phone’, the observed string might be: (D,D,D,D,PO,PO,D,D,D,D). Notice the difference between the two activities is the number of the inner relation (PO). In both activities, run-length encoding will produce the same compressed string (D-PO-D) and the qualitative spatial feature will not discriminate between activities. However, using the counts from the run-length encoding represents a clear distinction between the two activities. The ‘playing game’ activity will increment the (D-PO-Long) histogram bin whereas the ‘touching the phone’ will increment (D-PO-short) bin.

3.2.6 Smoothing Fast Changing Relationships

A common problem when using a discretised representation of space and time is the often erratic and rapid changes of relations between time points. This is also known as *jitter*. Jitter occurs mainly due to noise in low-level observation such as object detections or skeleton tracking. This causes undesirable sequences of relations which, when compressed, reflect many imaginative interactions produced due to noisy motion of bounding boxes or joints. Since our method heavily relies on modelling interactions within the activity to represent said activity, having unnecessarily large noisy interaction, negatively affects our model. We therefore aim to pre-process raw input streams of relations to attempt and smooth out jitter or imaginative interactions that occur with the input stream to a large extent. [Sridhar et al., 2011] tackles

this issue by introducing a hidden Markov model that learns transition probabilities from one relation to the other. It therefore acts as a filter by avoiding rapidly switching relations by estimating likelihood of these switches. However, a large amount of ground truth data is needed to estimate the parameters of this model. We use a median filter that does not require training data and is shown to perform satisfactorily as verified through cross-validation on the datasets available to us; this is described next.

Given a relations matrix \mathbf{M} , each element of $\mathbf{M}_{i \times j}$ contains the spatial relation that holds at the j^{th} frame between entity pair corresponding to i^{th} row. We apply this filters on each row of \mathbf{M} matrix before relations are compressed. A median filter passes a sliding window (of a pre-set length of time points) over each element of the input stream and replaces it with the most common element in the window. This has the effect of smoothing out swift or sudden changes in the input data. For example, consider a single row of the \mathbf{M} matrix with 5 frames as the following vector: $[D, D, PO, D, D]$. A median filter will pass over each element with a window size larger than 1, this will result in smoothing the single changing relation at frame 3 into the most common neighbouring relations producing the following resulting vector: $[D, D, D, D, D]$. The size of the window length is an important parameter, denoted by \hat{w} , of this type of filter. We select our window length based on actual observations to estimate the minimum number of frames below which changes of relations are likely to be noise and not an actual interaction.

3.3 Quantitative Representation

Whilst qualitatively representing space and time within an activity achieves high level of abstraction, as shown in the previous section, it is, however, not possible to capture every aspect of an activity qualitatively and some information is bound to be lost. A qualitative representation has to be carefully chosen according to domain knowledge to account for scale and speed of normally observable activities within that domain. Experimentation with different QSRs for example, qualitative direction calculus (QDC) or qualitative trajectory calculus (QTC) to capture relative direction or relative velocity, or learning a qualitative representation that best describes the training data, are performed to minimise information loss in the final representation. Previous work has experimented with combining multiple qualitative representations

to maximally capture different aspects of an activity [Duckworth et al., 2016b, Duckworth et al., 2017, Young and Hawes, 2015]. Differently from previous approaches, we have attempted to combine qualitative and quantitative representations together to achieve highly descriptive strength when representing an activity.

As mentioned before, the use of a qualitative representation is attractive since interesting events occur at interaction points within entities. These points are identifiable through qualitative changes in spatial and temporal relations between entities. Our quantitative representation is aimed at capturing motion of entities relative to each other and encode motion patterns quantitatively as descriptive features of the activity. This idea has been widely used in optical flow analysis and activity recognition of repetitive actions such as ‘jumping’, ‘walking’, ‘waving’ etc. Descriptive statistics are applied on observations of quantitative measures, such as distance or direction, between entities to extract features describing an activity through the combined motion flow of its entities. We assert that combining such features with qualitative representations provides a richer representation of activities. We further show that some lost information through qualitative representation can be recovered when using the proposed quantitative representation. We motivate the use of this representation through describing an example emerging from our datasets next.

3.3.1 Motivating Example

The use of qualitative representation is effective, however due to the coarse discretisation of space and time, it is often not possible to discern similar looking activities that are performed at vastly different scales, speeds etc. Quantitative spatial representations, on other hand, are able to encode fine motion patterns in an activity. Consider activities with only hands and an object as entities, for example ‘picking up pot’ or ‘picking up television’, assuming that object types are unknown, the qualitative relations between the entities are the same in both activities. The inherent activity is similar, however, the motion of hands is different in terms of scale. Scale can cause very different activities to share the same qualitative representation. For example, two objects approaching each other on a coffee table or on an airfield will create similar relations. The distances between the object are significantly different, this difference can be leveraged to

discern such activities where the qualitative relations do not provide much discerning power. A similar argument can be applied to activities with different speeds, directions etc. An example of a quantitative representation is shown in figure 3.10.

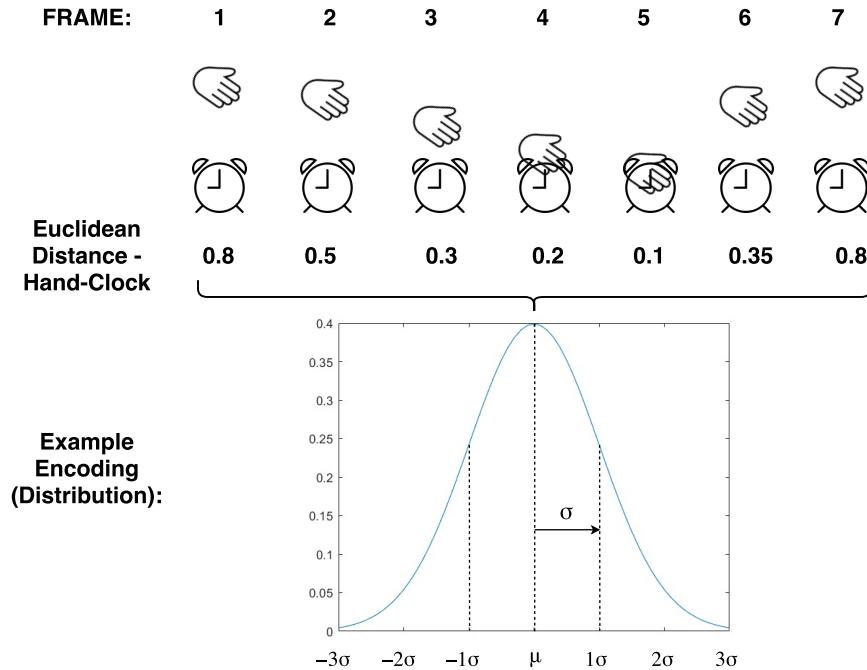


Figure 3.10: Simple example of a ‘turning off the alarm’ activity shown in terms of positional observations of a hand and a clock entity at each frame number. The corresponding Euclidean distance string between entities is also shown. Finally, the underlying distribution of distances represents the motion pattern between the entities. Statistically descriptive features can be extracted to represent the distribution.

3.3.2 Representation

In order to represent the activity quantitatively, we take inspiration from previous work [Behera et al., 2012] and the QSRs research, specifically qualitative distance calculus (QDC) [Clementini et al., 1997]. In both these approaches, the Euclidean distance between entities within a scene is used to derive a representation of the activity. Authors in [Behera et al., 2012] computes the distance between all pairs of entities and discretises the space into a set of learned relations. The QSR literature has proposed the QDC calculus which discretises distance into predefined spatial relations. For example, the ‘near’ or ‘far’ relations. We assert that discretising the distance

measure between entities into a small number of classes discretises the space excessively and thus the representation may suffer from over-abstraction. We propose a different method to represent the distances between entities using descriptive statistics in a fixed length feature vector.

Given an activity scene comprising of N_E entities \mathcal{E} and N_F frames. We compute the Euclidean distance *dist* between every pair of entities for all frames. This results in a distances matrix $\mathbf{X}^{N_E \times N_F}$. Given any row \mathbf{x} of distance matrix \mathbf{X} , we assume that the set of distances in this row are drawn from some distribution $\mathcal{D}(\text{mean}, \text{standard deviation})$. Given these observations, we model the distribution of distance values by extracting the moments of the distribution as the representative features of that distribution. Using moments to represent the distribution of a random variable is a common technique used in statistics to concisely and numerically represent the key qualities about a distribution. The use of moments further allows us to create a fixed length feature vector, using a fixed number of moments, which is of a low-dimensionality yet capturing the most important aspects of the distribution. Generically speaking, equation 3.7 shows the formula for computing the m^{th} moment of a distribution where x_i refers to an observation in our data comprising of N_F observations. The parameter a is a conditional parameter which defines that for the first moment, a is set to zero and for every other moment, a is set to the mean.

$$m^{\text{th}} \text{ moment} = \frac{\sum_{i=1}^{N_F} (x_i - a)^m}{N_F} \quad (3.7)$$

We represent the latent information using the first four moments of the distance distributions for each entity pair for a certain activity. These four moments are commonly known as mean, variance (var), skewness (ske) and kurtosis (kur). Higher order moments are often difficult to estimate and describe semantically. For example, the 5^{th} moment is a measure of the tails compared to the shoulders of a distribution. As higher order moments are computed, more specific aspects of the distribution are captured. We aim to avoid over-describing the distribution and therefore choose the first four moments which are most commonly used in statistics [Iserles, 1989]. We now formally present each feature extracted using these four moments.

- **Mean** The arithmetic mean is the first raw moment of a distribution ($m = 1$). It represents the centre tendency or centre location of the data. We compute the mean by using equation 3.8 which is in accordance with the general formula 3.7 for m^{th} moment:

$$\text{mean} = \frac{\sum_{i=1}^{N_F} (x_i - 0)^1}{N_F} \quad (3.8)$$

Whilst modelling activities, it is apparent that the mean of the distribution of distances between joints would be able to distinguish activities where the motion of limbs and objects is highly variable such as for activities ‘eating an apple’ and ‘kicking a ball’. However, for activities exhibiting closely related motion such as ‘eating an apple’ or ‘receiving a phone call’, the mean alone may not be sufficiently discriminating since the distance distribution’s mean values would be largely similar in both cases.

- **Variance (var)** The variance is computed using the second moment of the distribution. It is a measure of dispersion or the shape/spread of the data from the mean. It fits into the generic moment formula (equation 3.7) with $m = 2$ about the mean, this is seen in equation 3.9.

$$\text{var} = \frac{\sum_{i=1}^{N_F} (x_i - \text{mean})^2}{N_F} \quad (3.9)$$

Variance further represents the shape of distributions of distances from activities. An activity might have highly dispersed distances if the subject exhibits erratic and spatially wider motion such as jumping jacks. This would be different to a subject performing relatively spatially constrained activities such as ‘writing on paper’. Variance acts as a highly discriminating feature within these two activities. In order to establish consistent units with the mean, the square root of variance is often used and is known as the standard deviation (std).

- **Skewness (ske)** The third feature we use is skewness which is computed using the first and second moment of a distribution, the mean and standard deviation. Skewness is a

measure of symmetry or asymmetry of a distribution. A non-zero skewness represent a distribution that is distorted to the left or right from the normal distribution. We compute skewness of distributions using the third standardised moment of a distribution seen in equation 3.10.

$$\text{ske} = \frac{\frac{1}{N_F} \sum_{i=1}^{N_F} (x_i - \text{mean})^3}{\left(\sqrt{\frac{1}{N_F} \sum_{i=1}^{N_F} (x_i - \text{mean})^2} \right)^3} = \frac{\text{mean}^3}{\text{std}^3} \quad (3.10)$$

Most real-world application assume normal distribution of data and therefore do not value skewness as a highly important aspect of a distribution since skewness in normally distributed data is zero. However, our distance distributions of activities, observed from the datasets used, rarely show a perfectly normal distribution and is often skewed. This inspired the use of skewness as an important feature to capture in representing distributions of distances.

- **Kurtosis (kur)** The kurtosis is computed using the second and fourth moments of a distribution. Similar to skewness, kurtosis is a higher order moment of a distribution making it hard to intuitively explain. Using most recent definition, [Westfall, 2014, DeCarlo, 1997], we can define kurtosis as a measure of *tailedness* and *peakedness* of a distribution. The tails of the distribution tell us about outliers in data therefore interpretation of kurtosis is the representation of modelling such outliers. However, the tails of a distribution cannot be independently altered from the peak whilst keeping the variance unchanged, therefore, kurtosis is a measure defining both the peak and tails of a distribution [DeCarlo, 1997]. We compute kurtosis in our system using the fourth standardised moment of a distribution formula as seen in equation 3.11.

$$\text{kur} = \frac{\frac{1}{N_F} \sum_{i=1}^{N_F} (x_i - \text{mean})^4}{\left(\sqrt{\frac{1}{N_F} \sum_{i=1}^{N_F} (x_i - \text{mean})^2} \right)^4} = \frac{\text{mean}^4}{\text{std}^4} \quad (3.11)$$

Higher order moments could be used to further describe the data distributions however,

empirically, we have found that no significant descriptive strength is added by including further orders of moment in our feature space. Through feature selection, described in section 3.5, we ensure removal of non-descriptive features when training our activity models.

Features are extracted from the raw distance data matrix \mathbf{X} into a fixed-length feature space using the measures described above. Each row $\mathbf{x} \in \mathbf{X}$ of distances is used to compute the four moments described above. We denote the set of the resulting four moments computed for the i^{th} vector $\mathbf{x}_{i,*} \in \mathbf{X}$ as \mathcal{S}_i . These 4 moments represent the distribution of distances for a specific entity pair, as specified by the row of the \mathbf{X} matrix. This operation is performed across each row of the matrix and the results are concatenated in a feature space denoted by F_4 , shown in equation 3.12.

$$F_4 = \bigcup_{j=1}^{N_E} \mathcal{S}_i^j \quad (3.12)$$

Example 3.11 demonstrates the feature space generated through this process. Note that the dimensionality is equal to $N_E \times 4$, indicating four descriptive statistic values for each pair of entities in the activity video.

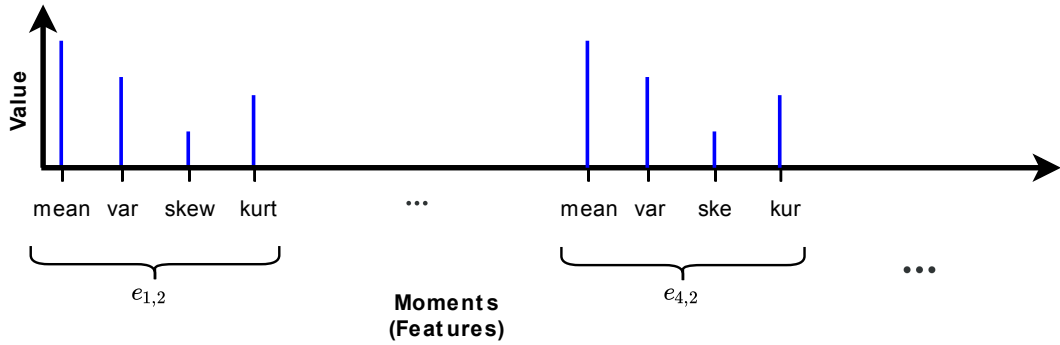


Figure 3.11: The Feature space for F_4 shown as a feature vector. These features capture four moments of a distribution of Euclidean distances amongst each entity pair in the activity to achieve a quantitative representation of the activity.

3.4 Objects Representation

Object detection in video is an open research problem. There is a vast amount of research present on object detection, however, accurate object detection and subsequent tracking is still an open problem. When modelling activities, it is obvious that the objects involved within the scene provide a lot of relevant information about the activity occurring. For example, ‘making tea’ activity would commonly involve objects such as tea bags, spoons, sugar, kettle, cup etc. Detection of these objects within the scene provide very accurate cues towards the activity being undertaken. However, it is rare that correct automatic recognition and accurate tracking of these objects is always present. We are therefore left with missing incomplete tracks and/or incorrect/missing recognition of objects in the scene. Activity recognition systems are therefore required to be robust enough to deal with such lack of object knowledge within the scene.

Our proposed system for simple activity recognition in this thesis is capable of achieving high accuracy in the activity modelling and recognition task without relying on a high amount of accurate object knowledge. We show that our system performs very well with object tracks only; no information on object affordances or classes is needed. We also show that our system performs well with skeleton tracks alone since it captures the most representative motion patterns stemming from various qualitative and quantitative representations utilised.

3.5 Feature Selection

Our final feature space is a concatenation of all the features extracted from the various representation described in previous section. We denote by F the final feature space where equation 3.13 represents the components of F . However, concatenation across many different feature spaces which represent various aspects of an activity, is bound to have redundant information across representation. We propose the use of a feature selection method which extracts the most discriminating features from our complete feature space. In this section we describe this method.

$$F = \langle F_1, F_2, F_3, F_4 \rangle \quad (3.13)$$

Feature selection is a process of selecting a subset of features that provide the most discriminative power to the resulting model for the prediction task. The goal of feature selection can be described as, given a feature set F , find a subset of those features $F' \subseteq F$ that maximises some criteria which eliminates redundant/irrelevant features. Feature selection is a commonly used as a pre-processing technique in machine learning due to its following benefits. With selecting a subset of the entire feature set, we ensure reduced dimensionality thus generating a more concise model which can be trained with less data resulting in shorter computation time. Higher performance can be achieved on the prediction task since non-discriminating and irrelevant features are removed, this is further discussed and empirically confirmed in experiments section. Finally, the model is interpretable since statistically most *important* features emerge which provides interesting insight into the representation used to describe the datasets [Chandrashekar and Sahin, 2014].

Many different techniques have been developed to perform feature selection that propose different criterion from accuracy/objective maximisation, incremental usefulness, inter correlation etc. to select the relevant feature [Chandrashekar and Sahin, 2014, Molina et al., 2002]. One such relevance criteria is to maximise entropic relevance using mutual information from information theory to generate a highly *informative* feature set [Wang et al., 1998]. [Peng et al., 2005] provides a popular framework that uses entropic measure for performing feature selection by simultaneously 1) maximising the relevance of selected features and 2) minimizing the inter-redundancy of the selected set. We will briefly describe the approach here.

Mutual information between two features \mathbf{a} and \mathbf{b} can be computed using equation 3.14 where $H(\mathbf{a})$ refers to the entropy of a feature \mathbf{a} .

$$I(\mathbf{a}, \mathbf{b}) = H(\mathbf{a}) - H(\mathbf{a}|\mathbf{b}) \quad (3.14)$$

Mutual information I is then computed between each individual feature f_i and the training class vector \mathbf{c} . The set of features that maximises the average mutual information values between individual features and class c are considered as most relevant. $D(S, c)$ captures this relevance as a measure called dependency which is computed between a feature set S and class

vector c . The search for the optimum set S is formally defined in equation 3.15.

$$\max D(S, c), D(S, c) = \frac{1}{|S|} \sum_{a_i \in S} I(a_i; c) \quad (3.15)$$

The selected set that has high relevance with the class vector c is likely to have redundant features since they all mutually agree largely with the class vector. In order to further prune the feature space to remove redundant features, the inter-dependency of features is computed by measuring the mutual information between features $I(a_i, a_j)$. $R(S)$ denotes the redundancy of a feature set S , similar to the dependence formalism, the redundancy is computed by selecting a sub-set of features that minimises the square average mutual information between features as shown in equation 3.16

$$\min R(S), R(S) = \frac{1}{|S|^2} \sum_{a_i, a_j \in S} I(a_i; a_j) \quad (3.16)$$

Finally, the process of mRMR combines the above two criteria in a single framework to simultaneously optimise both the relevance and redundancy. $\phi(D, R)$ denotes the operator which combines the maximum dependency D and minimum redundancy R as defined in equation 3.17.

$$\max \phi(D, R), \phi = D - R \quad (3.17)$$

Our presented feature space F comprises of a variety of different representations including quantitative features and qualitative feature which further comprise of features generated using different calculi. It is safe to assume that some overlap in knowledge represented is inevitable when using such a variety of modalities to represent the activities. Feature selection to remove redundant features whilst selecting the mode relevant features is therefore highly needed to prune our feature space. Using the approach defined above, we empirically find the effectiveness of feature selection included within our system. This is further discussed in the experiments chapter.

3.6 Concluding Remarks

In summary, we propose a framework to perform activity recognition of basic activities. The framework involves engineering a representation of activities using quantitative and qualitative representations. The qualitative representation uses a region connection calculus (RCC) and a modified qualitative trajectory calculus (QTC) to model spatial aspects of entities within an activity scene. Temporal information is captured using learned qualitative temporal relations. Quantitative representation is achieved by modelling the distribution of Euclidean distances between entities in the scene over time. Features extracted using these representations are concatenated in a single feature space, normalised and then optimally pruned to generate a set of highly discriminating features. A discriminative model (SVM) is trained to learn the various classes of activities in the training data.

Our key contribution in this chapter is to propose a feature space which jointly captures qualitative and quantitative aspects of an activity. We derive simple qualitative relations from existing calculi to represent space, learn temporal relations from data to capture temporal knowledge and propose a representation which captures entity motion in an activity quantitatively.

As shown in the experiments section 5, this system out-performs the state-of-the-art on simple activities which span short time frames. However, longer activities tend to attain lower performance. As activities become longer, more entities are introduced and larger feature spaces are generated. Activities spanning longer periods of time will result in incredibly large representations of interactions. Moreover, a high amount of inter-class information is repeatedly represented. When modelling activities at longer intervals as well as comprising activities at shorter intervals, representation of long activities would largely comprise of a concatenation of representation of inner shorter activities. This results in redundancy and hence unnecessary overlap within classes. We therefore require a different mechanism to model longer and complex activities. In the next chapter, we describe a framework to model complex activities. Short and simple activities are learned and recognised using the system presented in this chapter, while longer activities are inferred using the stream of these simple activities recognised. The framework is explained in detail in the next chapter.

Chapter 4

Hierarchical Modelling of Complex Activities

4.1 Introduction

In this chapter, we describe a framework developed for modelling and recognition of *complex activities*. We refer to complex activities as those that 1) span over an extended period of time, 2) compose of multiple sub-activities at multiple levels of granularity and 3) represent natural human behaviour. A challenging area of research in computer vision and artificial intelligence is the representation and recognition of such activities. One challenge is that different subjects perform the same complex activities with high amount of variation. Moreover, as activities extend for longer periods, their spatio-temporal structure of interactions tends to become more complicated. Also, inaccuracy exhibited by low-level visual trackers (such as skeleton/object trackers), creates recognition challenging due to noisy observations which cause errors in detections. Mostly noise occurs due to occlusion, change in scene lighting, erratic or quick motion etc. The method presented in chapter 3 is designed to deal with short clips of simple human activities. This task has been extensively researched and many methods have been conceived in literature which aim to model simple activities e.g. [Aggarwal and Cai, 1997, Aggarwal and Ryoo, 2011, Peng et al., 2016]. Despite the long history of this research, existing meth-

ods tend to become ineffective when dealing with activities over longer periods of time that exhibit greater complexity [Turaga et al., 2008, Aggarwal and Ryoo, 2011]. Latest methods which utilise deep architecture to automatically learn relevant features to model complex activities often require large amounts of training data to learn the parameters of the models [Herath et al., 2017, Wang et al., 2018]. With the introduction of faster computational ability in robotics, surveillance and many other applications, the demand to model and recognise complex activities is ever-increasing. One example of an application where efficient complex activity recognition is required is in long-term autonomous behaviour of mobile robots [Hawes et al., 2017].

To model complex activities, a simple idea previously explored is to build a hierarchy-like representation of such activities encoding the compositional structure within the activity [Aggarwal and Ryoo, 2011], however, existing approaches largely rely on hand crafting such models relying on experts to construct a hierarchical description of activities. Other methods employ graphical models, such as HMMs, which do not scale well with increasingly longer and more complex activities [Oliver et al., 2002, Oliver et al., 2000]. This is due to the added number of parameters which require higher amounts of training data to train the models. We propose a novel method for building a generalised hierarchical activity model from mark-up of activities acquired from multiple annotators in a video corpus. Multiple human annotators identify activities at different levels of conceptual granularity. Our method automatically infers the *parent-child* relationships between different activities from this data using semantic similarity of textual annotations and temporal consistency. We use the resulting model to interpret previously unseen videos in terms of the conceptual categories of the acquired model, thereby providing a layered/compositional description that is naturally understandable by people. Our method is robust to noise and can deal with missing observations resulting from mis-detections of low-level action recognition. Finally, our model is designed to scale with longer activities and requires relatively less data to train as compared to deep learned approaches. The flowchart of the framework is presented in figure 4.1.

As seen in figure 4.1, a generalised hierarchical model of complex activities is learned from training videos. We first augment annotations provided for these videos with further annotations such that each video has been annotated by more than one independent annotators.

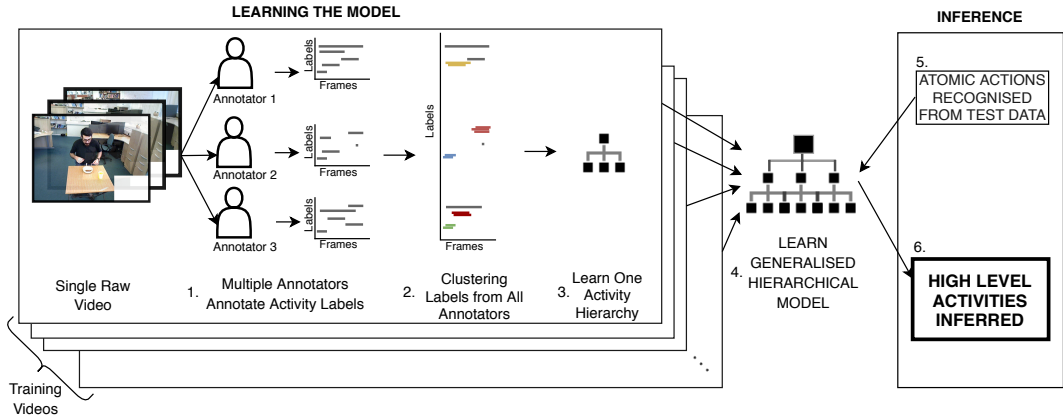


Figure 4.1: A flowchart of our hierarchical modelling and recognition of complex activities system presented. The process comprises of learning hierarchical structures of activity derived from human descriptions. Automatic low-level action recognition on a test video, using the QQSTR system, is then performed to infer the remaining hierarchical activity structure over the test video.

Multiple annotators naturally tend to describe activities at different levels of granularity. We describe our annotation collection process in further detail in upcoming sections. Then the hierarchical activity structure for each video is extracted as follows. Using the annotations which comprise of activity labels along with their temporal intervals (figure 4.1 part 1), we first reduce repeated identification of activity annotations and noisy annotations by clustering (figure 4.1 part 2). Since a single video is described by multiple annotators, it is inevitable that a single activity or action may be identified multiple times with slight variations in temporal extents and linguistic labels used. We reduce this redundancy by annotation clustering using a distance measure that takes into account the temporal boundaries of the intervals and the semantic similarity of the corresponding natural language labels. Note that there is no information provided by the annotators as to which activity annotation is a child of which other activity annotation (*parent-child* relation). These *parent-child* relations among the annotation clusters are then automatically learned by optimising a cost function which examines the configuration of the clusters and their label’s common theme. This results in a single hierarchical model (figure 4.1 part 3) that represents the complex activity occurring within one annotated video in a hierarchical format. Multiple hierarchies learned from different training videos reflect multiple training instances for the complex activity. These training hierarchies are then merged

to produce a probabilistic generalised hierarchical model of all activities in the training data (figure 4.1 part 4).

The model is then represented as an activity grammar which naturally captures the variation in activities along with probabilistic likelihood of each variation. Finally, given an unseen test video, the framework for simple activity recognition, presented in chapter 3, is used to recognise the low-level actions in the test video (figure 4.1 part 5). These recognitions are then used as an input stream for our learned activity grammar to infer the higher-level activities as an activity parse tree. The grammar allows for tractable inference and interpretation of new observations (automatically detected low-level actions) using a multi-threaded parsing algorithm inspired by the well-known Earley Parser [Earley, 1970], similar in some aspects to [Zhang et al., 2011]. This algorithm yields the exhaustive set of possible activity parse trees generated over every subset of input actions. Finally, high-level activities are inferred by searching this set of parse trees to find the optimal sub-set which describes the input actions with maximal likelihood (figure 4.1 part 5).

In the remainder of this chapter, we present details of our approach starting with the clustering mechanism aimed at clustering annotations from multiple annotators of the same video in section 4.3. Then, in section 4.4, we describe the process through which the optimum *parent-child* relations are established for each pair of annotation clusters to build a single hierarchy of activities. We then describe our approach to merge multiple hierarchies from different training videos to form an abstracted generalised hierarchical model which is represented as an activity grammar in section 4.5. Finally, in Section 4.6, we show how to perform complex activities inference given low-level action observations.

4.2 Annotation Collection Process

In this section, we describe the process through which activity annotations can be efficiently collected by multiple annotators. Typical datasets for activity recognition are usually only annotated by a single annotator per video. In order to infer the inherent hierarchical structure within an activity videos, we require these videos to be annotated by multiple annotators. As mentioned before, we notice that different people tend to describe activities at different levels

of granularity. We leverage this variation in human description to attempt to learn the latent structure of the activity hierarchy. Note that the annotations do not include any explicit notion of the *parent-child* relationships. Also, requiring annotators to annotate the entire syntactical structure of an activity is time consuming and highly prone to variations across annotators. Rather, we attempt to keep the annotation process simplistic so that it can be performed quickly; deeper hierarchical structures of activities can be extracted, and activity descriptions are independent of any one specific annotator. We describe the two main methods which are used for collecting additional annotations for our purpose.

4.2.1 In-house Annotations

Firstly, we employ research students for additional annotations. The annotators are given basic instructions requiring them to freely identify any activity within the video sequence then, using a pre-set list of activity labels, to select an activity label defining the identified activity and finally input the start and end time for that activity. Repeat this process for as many times as necessary to label all identified activities within a video. If a label is not present in the pre-set label list, the annotator is free to add it to the list for all future annotations. In this way, no bias is present towards the choice of identifying an activity and consistent labelling is achieved across annotators. No further instructions on granularity level of the annotation is provided. The in-house annotators have some knowledge of the research area and therefore the collected annotations are linguistically consistent across annotators and the overall annotation set is relatively noise free. The CAD and LAD datasets are labelled with in-house annotations.

4.2.2 Crowdsourced Annotations

Even though in-house annotations provide a noise-free annotation set and consistent labelling, the process is slow for large datasets which comprise of long activity videos. We therefore utilise a different mechanism for eliciting independent annotations. We use a crowdsourcing platform, namely Amazon Mechanical Turk (AMT), to build an interface whereby a world-wide pool of annotators is employed and hundreds of annotations gathered in minutes. The full description of our design and details on parameter decisions when designing this interface are presented in

[Tayyub et al., 2017]. However, a brief overview is provided here. Note that the CLAD dataset used in our experimentation process is labelled using crowdsourced annotations.

Activity Annotation Tool

Instructions:

Please watch the video below and then (1) answer the two questions below; and (2) use the annotation tool to identify as many activities as you can from the perspective of one of the actors. You are encouraged to identify activities that overlap in time and form constituents of one another. For example, in a kitchen scenario, "dicing onion" might occur as part of "cooking dinner".

To identify an activity:

- Use the controls to move to the frame at which the activity starts and press "Set Start Frame", and likewise for "Set End Frame".
- Enter a short label (1-3 words) describing the activity from the perspective of the chosen actor - for example, "cooking dinner", "dicing onion", "picking up spoon".
- Press "Add Activity" to add into the table on the right-hand side. If you wish to remove an identified activity, you can delete it from the table.

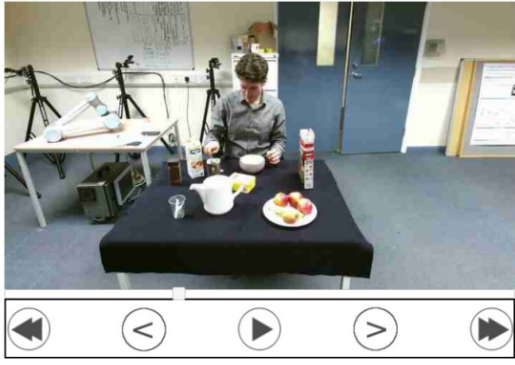
You will be paid ?? for each activity appearing in the table, and a bonus of ?? per activity if you provide concise and appropriate labels, accurate start/end times and a spread of overlapping activities that are constituents of one another.

Allow up to a minute for the video to load.

Section 1


How many times does the subject use the teapot? (Answer as a single digit) Does the subject check their phone? (Y/N)

Current Frame Number: 2864



Activity Label	Start Frame	End Frame	Delete
Pick Milk	2453	2480	delete
Open Lid	2513	2577	delete
Pour Milk	2578	2748	delete
Close Lid	2749	2819	delete
Add Milk To Bowl	2453	2864	delete

Timeline



Section 2

Start Frame

End Frame

Activity Label

Add Activity

Figure 4.2: A screen shot of the web-interface used to collect crowdsourced annotations of activities.

A web interface, as shown in figure 4.2, is first developed to allow for annotation collection through crowdsourcing. Annotators are given a time limit and a small monetary incentive to identify and label as many activities as possible. A minimum amount of labelling is set to help prevent non-diligent annotators from submitting inaccurate answers since a certain number of entries must be input before the annotation answers are accepted. Other filters are also used such as spell checks, validation questions, inter-annotator agreement etc. to further filter out spam answers. A table on the right show annotations submitted so far and a time-line below displays the position of each activity label. The time-line visualisation is in place to encourage annotators to label parallel activities and not just sequentially occurring activities. Labels are set at a 2-3 words limit, and annotators are free to use any short phrase to describe the

activity. Different to in-house annotation collection where an amendable pre-set list of labels was provided, with this method greater freedom of choice is allowed. However free labelling introduces further complexity when comparing labels for assessing similarity. We describe how these issues are addressed in sections 4.3 and 4.4. An average of 5 annotators annotate each video using this interface. The data is aggregated and a clustering mechanism is employed to remove redundant labels described in the next section.

4.3 Clustering Intervals Input Data

In this section, we describe the process in which activity annotations from different annotators are combined into a unified set of activity intervals each of which describe a unique activity.

4.3.1 Motivation

As described in section 4.2, we gather activity annotations from multiple annotators for a single training video. We then merge annotations from all annotator in order to produce a single annotation list that describes activities at different levels of granularity in a single video, from the perspective of many different annotators. Recall that annotators are simply shown the entire video and are free to label activities at any level. They further have the freedom to use any English phrase to define the activity. Though this creates a flexible and unbiased labelling of the video with no research-relevant intervention, it does present the following major challenges in the resulting combined annotation:

- Different annotators label the same activity segment using different English phrases, this introduces redundancies within the annotation data.
- Annotators identify spurious activities which usually are not part of the main top-level activity of the video. For example, ‘scratching head’, ‘staring at camera’ etc. within a ‘Having breakfast’ activity video.
- Annotators identify parallel activities, for example ‘Reading Newspaper’ at the same time as ‘Having Breakfast’.

These challenges are highlighted through a working example of a sample combined annotation acquired from multiple annotators for a single video in section 4.3.2.

4.3.2 Extended Example

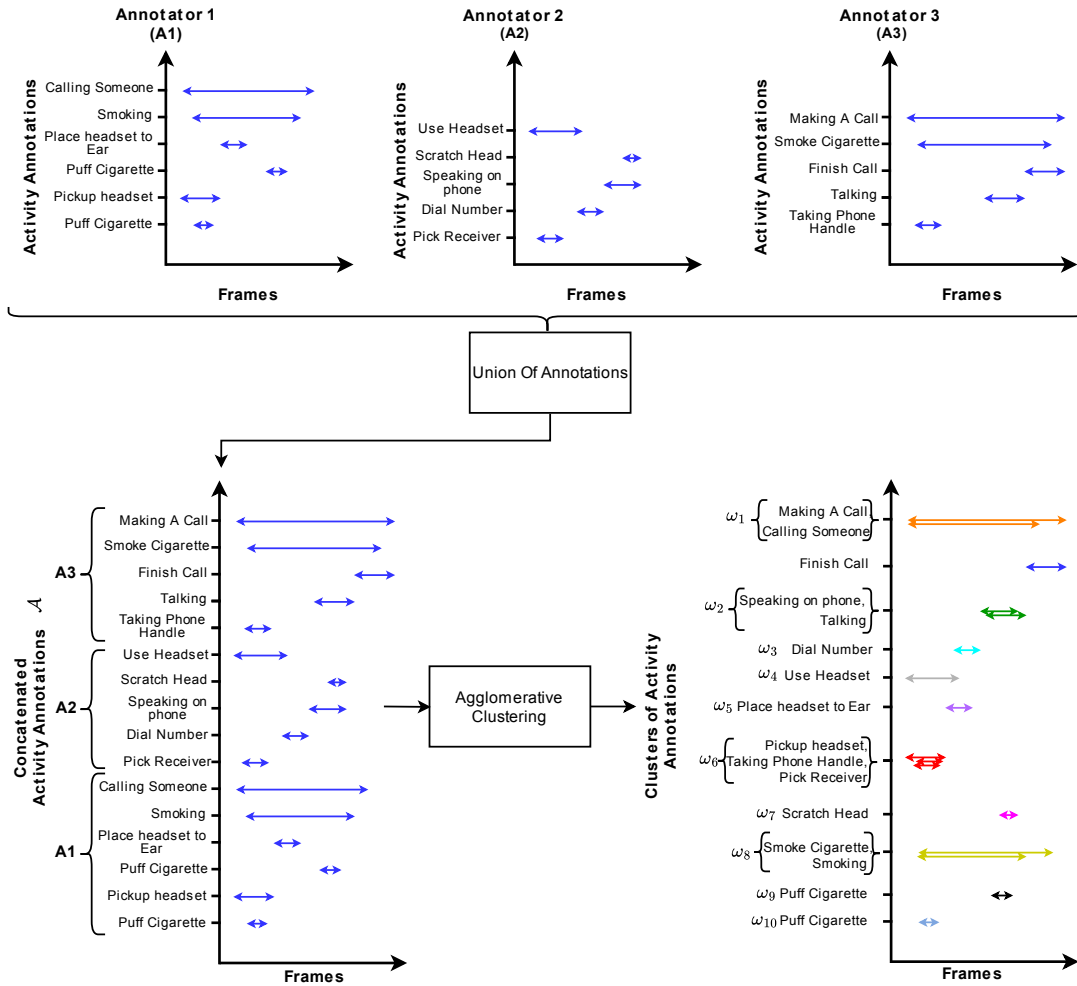


Figure 4.3: Agglomerative clustering over an example set of activity annotations for a single video of ‘using the phone’ activity. Annotations of a single video from three annotators are shown on the top. These are concatenated together to produce a saturated activity annotation describing activities at various granularities. An agglomerative clustering, with a custom distance function, is performed and the resulting clustering is shown to cluster all redundant labels while keeping sub-activity structures separated. Each cluster is referred to as an activity node ω .

Figure 4.3 shows a sample annotation of a single video acquired from three annotators. These

annotators independently and freely identify activities within the video. Note that there is high variation in granularity, linguistic labels and length of the identified activities across different annotators. The figure also highlights the challenges presented in section 4.3.1. Further note that the three annotation sets also contain activities that are irrelevant to the top-level activity. Finally, activities executed in parallel are also identified. We will continue to use this example to demonstrate our methodology through the remainder of this section. In order to unify the annotations from different annotators to build a single set of annotations without redundancies and noise, we present an agglomerative clustering approach next.

4.3.3 Agglomerative Clustering of Annotations

In order to unify annotation data from various annotators, we automatically cluster annotations. Concretely, given a collective list of annotations from all annotators for a video, the task is to identify annotations that are *conceptually close* in order to cluster them together. Consider a video annotated by N_T number of annotators. Annotations for this video are denoted as a set $\mathcal{A} = \{A_1, A_2, \dots, A_{N_T}\}$ annotations. Each annotation $A \in \mathcal{A}$ is a set of annotated activity intervals $\iota \in A$. Each activity interval ι is a 3-tuple representation of an annotated candidate activity starting at time s , ending at time e , and carrying the label l making $\iota = \{s, e, l\}$. The goal of clustering is to partition the combined annotations from all $A \in \mathcal{A}$ into subsets (clusters), each of which exhibit a high degree of similarity between their elements [Gowda and Krishna, 1978]. This necessitates a carefully chosen similarity measure to judge the *closeness* between data-points. We introduce a distance measure which evaluates the similarity between two annotated activity intervals (ι_i, ι_j) in terms of the similarity between their start and end times as well as the linguistic similarity between their respective labels. For example, take two annotation intervals (20, 40, ‘pick up phone’) and (22, 35, ‘answer a call’), note that their start and end times are approximately similar on a video sampled at approximately 25-30 frames per second. Moreover, their labels exhibit high linguistic similarity as well. These annotations are therefore assumed to be referring to the same activity and thus given high numerical similarity value. Standard similarity measures, such as Euclidean distance, are often metrics, our measure requires the inclusion of a non-metric linguistic similarity which motivates a design of a new

similarity measure detailed in the next section.

Annotated Activity Interval Distance Measure

To define a similarity measure we propose a distance function between two annotations that takes into account three aspects of the annotation namely start times, end times and linguistic labels. Formally, take $\iota_i(s_i, \epsilon_i, l_i)$ and $\iota_j(s_j, \epsilon_j, l_j)$ be two annotated activity intervals. As mentioned before, the distance measure is constructed based on the 1) linguistic similarity of labels and 2) temporal boundary similarity. To measure the linguistic similarity of any two labels written in form of short English phrases, we adopt the benchmarking linguistic semantic similarity system developed by [Han et al., 2013]. Using this off-the-shelf system, we can define a similarity function denoted by $\lambda^l(l_i, l_j)$ that returns the similarity between any two label (l_i, l_j) in the range $[0,1]$ where high values denote extremely high semantic similarity and low values indicate the contrary. The semantic similarity value between phrases is computed by an *align and penalise* process. First the inter-word similarity between words of two phrases is computed. Comparing two words involves first learning a word co-occurrence matrix using a large text corpus. This matrix conceptually captures the frequency of a pair of words appearing close to each other. It is a common assertion in NLP that words that appear close together are likely to be related. Taking the singular value decomposition of the matrix results in low-dimensional word vectors and the cosine similarity between any two vectors denotes the semantic score between words. This scoring mechanism is then extended to phrases by first computing the pairwise score between the words of the two phrases. Then an overall score term is computed by finding a word pairing scheme which maximising the scores of individual word pairs. This score is then penalised for mismatching or highly dissimilar terms to produce the semantic similarity between phrases. More details on the exact computation of these various measures can be found in [Han et al., 2013].

Computing the distance between temporal boundaries in classic interval data has been previously researched and many distance measures proposed to quantise proximity between interval data [de Souza and de A.T. de Carvalho, 2004, Gowda and Ravi, 1995]. [Cha, 2007] provides a comprehensive set of commonly used distances between a set of arbitrary points in space. Differently from previously proposed distance measures, We define a distance function

$\delta(\iota_i, \iota_j)$ to capture not only the temporal overlap of the two intervals (based on the start and end times), but also the semantic similarity of their labels, based on linguistic closeness computed using $\lambda^l(l_i, l_j)$. Equation 4.1 shows the computation of the distance between two annotated activity intervals.

$$\delta(\iota_i, \iota_j) = \begin{cases} \eta(|s_j - s_i| + |\epsilon_j - \epsilon_i|) & \text{if } \lambda^l(l_i, l_j) > \hat{\lambda}^l \\ 1 & \text{otherwise} \end{cases} \quad (4.1)$$

Note that the difference in start and end frames of intervals are computed using the city-block distance while η normalises this distance and forces δ to stay in the range $[0, 1]$, where 0 represents high similarity (low inter-distance) between annotated intervals and 1 represents complete disparity (high inter-distance). We use the city block distance over other distance measure for the following two reasons:

1. Since city block distances are computed as the first norm, this distance examines the absolute difference between pairs of coordinates. This tends to preserve the semantics of time in terms of difference of number of frames between the intervals as their distance. Other distance measures, such as Euclidean, cosine or other higher norm Minkowski distances, would produce a more convoluted distance value.
2. Having an interpretable distance measure, in terms of frame numbers, further allows for easily selecting justifiable parameter values within the clustering framework. More details are presented later in this section.

Recall that the semantic similarity of linguistic labels computed using the λ^l function is not a strict metric, i.e. it follows the reflexive and symmetric property however it does not obey the triangular inequality. Therefore, the resulting value from this function cannot be arithmetically included within the distance metric δ computation. We therefore influence the distance measure δ by the semantic similarity of labels through imposing a precondition. The condition specifies the following: if the interval's labels are found to have a high semantic similarity value, only then consider start and end frame similarity to compute the final distance between the intervals, otherwise set a high value as the final distance indicating high dissimilarity. In other words,

set δ to 1 for the *linguistically different* pair of intervals. $\hat{\lambda} \in [0, 1]$ is the cut-off threshold of semantic similarity at which activity intervals do not relate semantically enough which results in setting δ to the maximum value 1. Having defined a distance measure to assess *closeness* of two annotation intervals, we now present the clustering method employed to utilise this measure and cluster the intervals.

Clustering Algorithm

Given a single video's annotations $\mathcal{A} = \{A_1, A_2, \dots, A_{N_T}\}$ comprising of annotation responses from N_T multiple annotators. We first concatenate the annotations intervals from each annotator's annotations $A \in \mathcal{A}$ according to equation 4.2 to produce a merged annotation list \mathbb{A} of annotations from all annotators. As seen in figure 4.3, it is obvious that many intervals have been re-identified and annotations of activities at multiple levels of granularities are emergent.

$$\mathbb{A} = \bigcup_{i=1}^{N_T} A_i \quad (4.2)$$

To cluster the repeated and redundant intervals, we use an agglomerative clustering method on elements of \mathbb{A} which learns a set of disjoint clusters comprising of annotation intervals that refer to the same activity. Many clustering algorithms can be used such as K-means, spectral clustering etc. Our choice of the clustering algorithm is driven by the form of our data. Since we attempt to represent each data-point (annotation interval) by its start, end time and a linguistic label, a clear embedding in a n-dimensional numeric space is not obvious due to the inclusion of a linguistic label dimension. However, linguistic similarity is still captured through influencing the pairwise distances between data-points to build a dissimilarity matrix as detailed in the previous section. Agglomerative clustering is a standard mechanism to perform clustering in a bottom-up hierarchical manner which takes as input a similarity or dissimilarity matrix. In agglomerative clustering procedures, data points are clustered together successively. The process involves identifying two most *similar* samples and merging them into a single cluster by assigning them the same cluster label. In subsequent iterations, previously merged clusters act as the building blocks for further clustering. This repeats until all data-points or clusters are assigned in the same cluster [Gowda and Krishna, 1978]. The implementation of agglomerative

clustering algorithm is described in algorithm 1.

Algorithm 1 Agglomerative Clustering with Complete Linkage

- 1: **Input:** Combined Activity Intervals $\mathbb{A} = \{\iota_1, \iota_2, \dots, \iota_{N_D}\}$, Cut-off threshold $\hat{\delta}$.
 - 2: **Output:** Cluster assignments $\Theta = \{\theta_1, \theta_2, \dots, \theta_{N_D}\}$, \mathfrak{D}_k for $k = 1$ to N_D .
 - 3: Initialise:
 - 4: $\mathbf{D}_{ij} = \delta(\iota_i, \iota_j), \forall i, j$.
 - 5: $C_i = \{\iota_i\}, \forall i$.
 - 6: **for** $k = N_D$ **to** 1 **do**
 - 7: $d(i, j) = \mathbf{D}(C_i, C_j), \forall i, j$.
 - 8: $\mathfrak{D}_k = \{\{C_1, \dots, C_k\}, \min_{a,b} d(a, b)\}$.
 - 9: **if** $\min_{a,b} d(a, b) \geq \hat{\delta}$ **then**
 - 10: $\theta_i = \{\nu : \iota_i \in C_\nu\}, \forall i$.
 - 11: **end if**
 - 12: $l, m = \arg \min_{a,b} d(a, b)$.
 - 13: $C_l = \text{Join}(C_l, C_m)$.
 - 14: $\mathbf{D}(C_l, C_i) = \max_{x \in C_l, y \in C_i} d(x, y), \forall i$.
 - 15: Remove C_m and renumber indices of C .
 - 16: **end for**
-

The input to algorithm 1 is the set of merged annotation intervals \mathbb{A} as well as a threshold $\hat{\delta}$ which specifies the maximum inter-cluster distance beyond which no further clustering is performed. Conceptually, this value corresponds to the maximum number of frames that two annotation intervals are apart after which they are not clustered. Since we have used a distance measure that preserves the time semantic in terms of frame numbers, the choice of this parameters boils down to deciding how far apart two annotation intervals need to be in terms of number of frames, to assign them different clusters with high certainty. More on this is covered in the experiments chapter 5. The algorithm outputs the cluster assignments for all intervals as well as a dendrogram \mathfrak{D}_k which is a standard representation of the overall hierarchical clustering [Gowda and Krishna, 1978]. In the algorithm, a distance matrix $\mathbf{D}^{N_D \times N_D}$ where $\mathbf{D}_{ij} = \delta(\iota_i, \iota_j) \forall i, j \in \{1, \dots, N_D\}$ is initialised using interval similarity measure δ defined in the previous section. Moreover, each interval in \mathbb{A} is clustered in a single cluster where C represents a set of these initial clusters. Then, given C , we iteratively cluster together the two closest clusters until all elements have been clustered into a single cluster. Within each iteration, a list of all pairwise distances between clusters is first generated and stored in d . Then the closest pair of clusters l, m are identified and merged into one cluster C_l . The distances between this new

cluster and all other clusters/data-points is recomputed. Finally, the indexes of the clusters are incrementally renumbered to allow for correct numeric fitting of the new cluster. A dendogram \mathfrak{D} maintains a list of all clusters at each iteration along with the minimum inter-cluster distance identified at that cluster. Whilst the minimum inter-cluster distance is larger than the given threshold, cluster assignment is performed by assigning all interval l_i , its corresponding cluster index ν retrieved from the current cluster list $\{C_1, \dots, C_k\}$. We stop this assignment, and in effect the clustering, once δ between two clusters exceeds the threshold $\hat{\delta}$.

Defining distance/proximity between new clusters

Establishing a method to compute the distances between newly formed clusters at each iteration after the first is an important design decision within the agglomerative clustering framework. Since new clusters consist of multiple data-points, this creates multiple candidates to compute the distance from other entities (data-points or clusters). Computing distance from newly formed clusters to other entities is needed to update the distance matrix \mathbf{D} for the next iteration. The choice of this method is what differentiates different agglomerative clustering techniques [Tan et al., 2005]. The task translates to selecting a prototype or a representative point of a new cluster. This point is used to compute proximities to other clusters. This point can either be a) one of the existing data-points or b) a new point for example the mean of all the data-points within the cluster. However, option b) requires the assumption that the distance measure used to compute similarity or dissimilarity between data-points is a strict metric [Arkhangelski and Fedorchuk, 1990]. Euclidean distance is a popular example of such a measure. We are unable to utilize methods that generate new points as prototypes as in option b) since our distance measure is not a strict metric. This was because the devised distance measure, equation 4.1, loses the metric property when influenced by the linguistic similarity measure. We therefore have to use option a) whereby the intra-cluster distance measures, which select one of existing data-point as the cluster prototype, is used.

Three main types of cluster proximity measures exist that are compatible with non-metric distance measure and only follow the symmetric and non-negative properties. These proximity measures are referred to as single, complete or average linkages. Keeping in mind that that our clustering mechanism utilises dissimilarities, we explain the three proximity measures in this

context. Average linkage takes the mean of distances between elements of two different clusters. Single linkage considers the closest distance between elements of two clusters. This method is a conceptually strict paradigm where the shortest distance between two points from two different clusters represent the cluster proximity. Conversely, complete linkage is a conceptually liberal method since it uses the largest distance or the distance between the farthest points from different clusters as the intra-cluster distance, see figure 4.4.

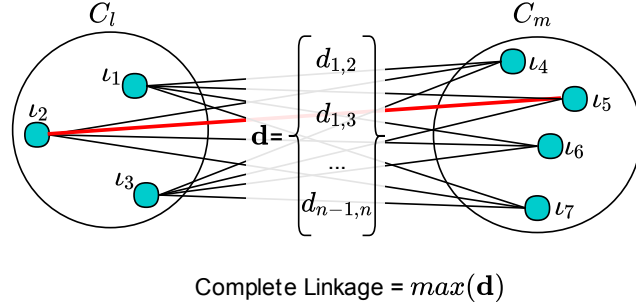


Figure 4.4: Complete linkage for agglomerative clustering proximity measure illustrated as the maximum distance between elements of two clusters.

Complete linkage clustering results in clusters which are less likely to merge because of maximally large distances representing their proximities. Single linkage methods often suffer from noisy observations [Schütze et al., 2008] and provide a conceptually optimistic mechanism to cluster different clusters. Differently to single linkage, complete linkage is known to be robust against noisy observations and requires high similarity before further clusters are formed [Schütze et al., 2008]. This is a desirable property since our data comes from highly variable human annotators which inherently introduces noise in interval edges and labels used. Further, unnecessary and excessive clustering of intervals is avoided through using complete linkage. Therefore, we employ complete linkage to measure proximity between newly formed clusters.

Cluster representation

Once the clustering is complete, we represent the results by combining the labels and corresponding intervals into a single entity called an activity node ω . Given a clustering assignment Θ , we extract each cluster and convert it to an activity node. An activity node is generated from a cluster by computing the averages of all start and end times of the elements of the cluster,

and aggregating labels from all elements within the cluster to produce a label set. Formally, a multi-label activity node generated from a cluster C is denoted by $\omega(\bar{s}, \bar{e}, L)$ with start time $\bar{s} = \sum_{i \in C} s_i / |C|$, end time $\bar{e} = \sum_{i \in C} e_i / |C|$, and a collective set of labels $L = \{l_i\}_{i \in C}$. This can be seen in the last diagram of the figure 4.3. Note that having multiple labels, which are mostly synonyms, represent each activity interval provides a richer knowledge base for further processing and construction of activity models than using classic single label activity annotations. In conclusion, given an aggregated set of annotations from a video \mathbb{A} , the clustering approach devised results in a set of N_A activity nodes denoted by $\Omega = \{\omega_1, \omega_2, \dots, \omega_{N_A}\}$.

4.4 Learning Hierarchical Structure of Activities

The output from clustering aggregated annotations from various annotators for a single activity video produces a set of clusters each of which refer to a distinct activity occurring at some granularity within the video. Recall that we refer to each cluster as an activity node ω represented as a rich interval unit comprising of a start time \bar{s} , end time \bar{e} (in frames) and a set of English labels L that synonymously describe the activity occurring within that time frame. Figure 4.3 shows a sample clustering result of a training video. In this section, we tackle the task of learning a hierarchical structure that describes the overall activity within the video at different granularities as a compositional structure of the activity nodes. We propose a method that allows for learning the *parent-child* relationship present between each pair of activity nodes within a video. Despite the simple premise of the problem, further analysis of this task reveals a number of challenges. In the remaining part of this section, we first describe the challenges that make this task hard, we then describe the simplifying assumptions before finally explaining the mechanism of learning the overall hierarchy of the activity.

4.4.1 Challenges

Different to typically used knowledge-driven approaches to learn and model inherent compositional structures of complex activities, we attempt to learn a data-driven hierarchical model of an activity as described by annotation data. This allows for capturing human knowledge and perception of the activity hierarchy. However, learning a unified and abstract hierarchical

description of an activity from human labelled data presents the following challenges:

1. **Parent-child relationship:** Consider a sample set of activity nodes in a single video as shown in the final diagram of figure 4.3. Note that through clustering across aggregated annotations, activities identified at different levels of granularities are combined into a single list of activity nodes. No information is present about the relationships between these activity nodes i.e. which activity node is a parent or child activity of which other activity node. In a perfect and noise-free labelling of activities, this task is trivial since a simple assumption which states that every activity whose node temporally subsumes another activity node is assumed to be a parent activity of the subsumed activity, given this assumption all activities are related to the top-level activity. However, in a freely labelled activity video obtained from various annotators, noisy and subjective temporal boundaries of activities are common, making the task non-trivial. Figure 4.5 illustrates a set of common activity nodes from a single video with noisy boundaries. Using some simplifying assumptions, we propose a method to automatically infer the *parent-child* relationships between activity nodes which exhibit noisy temporal boundaries.

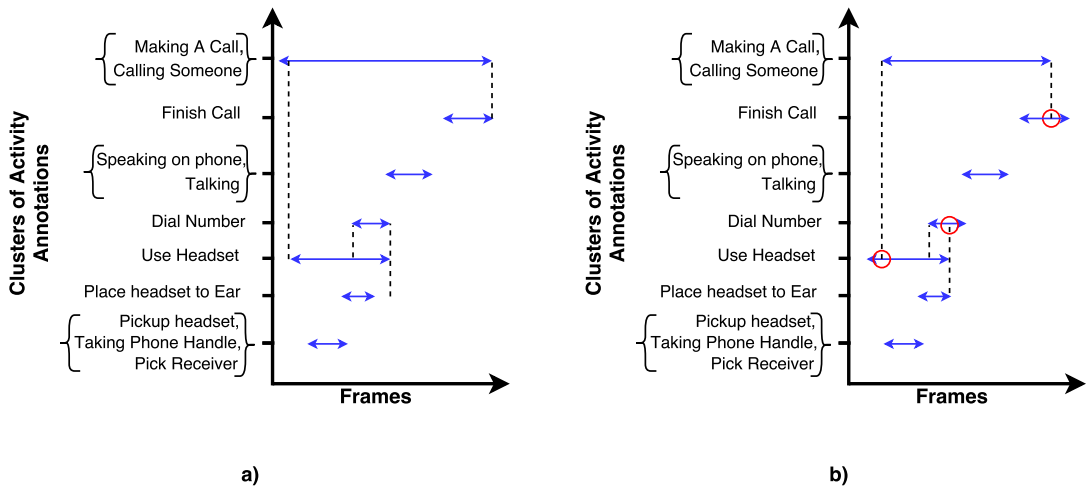


Figure 4.5: Clean and noise-free annotations shown on the left, notice the full subsumption of activity intervals within one another. The right figure shows a real-world example of annotations. Due to subjective temporal boundaries and noise, the red circles show *parent-child* activity nodes with partial subsumption. This makes the task of inferring the correct *parent-child* relationships non-trivial.

2. **Parallel activities:** Another challenge is encountered when different annotators identify different activities occurring concurrently i.e. within the same or similar start and end times. Annotations of these activities and their sub-activities exist together with other concurrently running activities and their respective sub-activities. Aggregating annotation labels across different annotators further exaggerates this mixing creating a challenge to separate compositional structures of different concurrent activities from one another. For example, refer to the figure 4.5, note that activity regarding ‘making a call’ has sub-activities such as ‘dial number’ or ‘use headset’. However, within the same time slots, there are other identified activities like ‘smoking a cigarette’ and its sub activities ‘take a puff’. This presents a challenge when inferring the inherent hierarchy of activity from labels in the form of: how to disambiguate labels of sub-activities of one activity from another within the same time frame without any knowledge of *parent-child* relations.
3. **Noisy labels:** A generic challenge encountered in many machine learning systems is noisy labelling. In our annotations, noise refers to either inaccurate or inconsistent temporal boundaries, erroneous activity labels and identification of spurious or irrelevant activities. Some of the *noise* has been addressed in the first challenge, however, further difficulty is faced when annotators identify spurious or irrelevant activities from the main activity of the video, such as ‘scratching head’, ‘staring into the camera’, ‘thinking’ etc. A design choice made here is to regard such activities as spurious activities since we argue that activities which are not objectively recognisable by the employed low-level activity recognition system constitute as noise. Despite these noisy identifications, our framework is designed to eventually filter out such activities through probabilistic likelihood computation. This is described in-depth in upcoming sections.

4.4.2 Assumptions

To solve the problem of learning a compositional hierarchical model of the activity from annotations, we need to overcome the above mentioned challenges. Similar to any machine learned model, assumptions are introduced to bound the capabilities on the model by proposing a solution on a slightly simplified task. We present the assumptions used in our method next.

1. **Spatial Constraint:** The first assumption states that observed activities are spatially constrained. Activity videos recorded in the datasets that have been used in this thesis are *daily living* activities such as ‘having breakfast’, ‘taking medicine’, ‘cleaning a microwave’ etc. The camera frame is therefore focused on one or two subjects performing these activities. Since each video comprise of only one primary top-level activity, as was intended by the dataset, we assume that all sub-activities identified by annotators for that video are children activities of the top-level activity or activities. For example, in a video capturing a human subject performing an activity of ‘having lunch in a restaurant’, we expect to observe identified activity labels such as ‘preparing tea’, ‘grabbing sugar’, ‘making payment’ etc. with higher probability than ‘scratch head’ or ‘refuel a car’ etc. This is confirmed from observing the datasets used as well as common knowledge.
2. **Temporal Coverage:** This assumption follows from the spatial constraint assumption stating that any set of activity nodes that are temporally subsumed by another activity completely or partially are all possible candidate children of the subsuming activity. Given that activities are spatially constrained, this is a reasonable assumption since it is highly likely for human annotators to only identify activities relating to the top-level activity presented within a video. However, it is important to note that in case of parallel activities, the *parent-child* relationships are harder to define since for any one sub-activity label, there are potentially many candidate parent activity labels.

4.4.3 Cost-based Optimum Activity Hierarchy Inference

We now present our method to learn the activity hierarchy from a video by inferring hierarchical structure of activity nodes. From agglomerative clustering described in section 4.3, for a single training video, we obtain a set of activity nodes $\Omega = \{\omega_1, \dots, \omega_{N_A}\}$. Given this set Ω , we propose a mechanism to learn an activity hierarchy \tilde{H} of activities from Ω . Each \tilde{H} is defined as a set of *parent-child* relations between qualifying pairs of activity nodes. A *parent-child* relationship between any two activity nodes ω_i and ω_j is denoted as $\omega_i \triangleleft \omega_j$ where ω_j is a parent activity of ω_i . This relationship is automatically inferred between any two activity nodes by devising an optimisation which takes into account 1) temporal overlap between the activity nodes and

the semantic similarity between their linguistic labels. In depth explanation of these processes is provided further on in this section.

Learning a *parent-child* relationships between all pairs of activity nodes in Ω allows us to infer a hierarchical arrangements of activity nodes $\omega \in \Omega$. Thus creating a hierarchical tree of the inherent top-level activity within the video. In this section, we first present some standard notation, then cover newly devised comparison methods used to compare activity nodes and finally present the optimisation mechanism to learn the hierarchical model of the activity from a single video's annotations.

Graph Notation and Bipartite Graph Matching

A graph $G = (V, E)$, with p vertices and q edges comprises of a vertex/node set $V(G) = \{v_1, \dots, v_p\}$ and an edge set $E(G) = \{\epsilon_1, \dots, \epsilon_q\}$ where each edge $\epsilon \in E$ can be either an unordered pair of vertices denoted by $\{v_i, v_j\}$ or an ordered pair denoted as (v_i, v_j) . Directed graphs are simple graphs where all edges are ordered pairs of vertices.

Bipartite graphs are a special class of graphs of the form $G = [V = (X, Y), E]$, comprising of a set of vertexes V and a set of undirected edges $E = \{\{v_i, v_j\} | v_i \in X, v_j \in Y \forall i, j\}$ that represents the links between vertexes. In simple terms, bipartite graphs are graphs where the set of vertices of the graph are split into two sub-sets X and Y . No edges exist between vertexes within the same set, however, full connectivity is established between vertexes of different sets. A special case of the bipartite graph is the weighted complete bipartite graph [Weisstein, 2002]. These graphs have the following properties:

- The set of vertexes V is partitioned into two independent subsets X and Y .
- For any two vertexes $v_1 \in X, v_2 \in Y$, there exists an edge $\epsilon \in E$ denoted by $\{v_i, v_j\}$.
- There are no edges formed between any two vertexes v_i and v_j from the same set i.e. where $v_i, v_j \in X$ or $v_i, v_j \in Y$.
- Each edge $\epsilon \in E$ is assigned a weight $w(\epsilon)$. This weight resembles the cost of the similarity measure between the two vertexes adjacent to this edge.

A bi-adjacency matrix $\mathbf{B}^{|X| \times |Y|}$ is often defined to encode the edge weights amongst the two sets of vertexes in the weighted bipartite graph. Unlike the adjacency matrix of standard graphs, this matrix is not strictly a square matrix, albeit to the possible disparity between the sizes of the two partitioned sets of vertices X and Y of the graph.

A common operation on bipartite graphs in graph theory is the matching of the two lists of nodes X and Y . This task is known as maximum weighted bipartite matching [West et al., 2001], also described as the assignment problem. An assignment problem in the context of a complete bipartite graph matching is defined as finding the optimal assignment of vertexes from set X to set Y such that the sum of the weights (cost) of the assigned edges is minimized. Formally, given sets X and Y of equal cardinality or sizes $|X| = |Y|$, and a cost bi-adjacency matrix \mathbf{B}^c where $\mathbf{B}(i, j) \in [0, 1]$ captures the cost of assignment of vertex $v_i \in X$ to $v_j \in Y$. The aim is to find a bijective function $f : X \mapsto Y$ such that the cost function $\sum_{x \in X} \mathbf{B}_{x, f(x)}^c$ is minimized. For example, consider a set of jobs, a set of workers and a cost matrix which stores the time each worker takes to perform each job. An assignment problem seeks to find the assignment of workers to jobs such that 1) no two workers are assigned to the same job and 2) the overall time to complete all jobs is minimised. The problem can be solved in polynomial time using the Hungarian algorithm [Munkres, 1957]. This algorithm finds an optimal assignment in polynomial time $\mathcal{O}(V^2E)$ as opposed to the more trivial brute-force search approach which can take exponential time.

Activity Nodes Matching

Extracting hierarchical structure of activity in a video requires inferring the *parent-child* relationship between activity nodes of that activity. As briefly mentioned before, one requirement to establish this relationship between any two activity nodes is semantic closeness between their label sets. Note that this is different to the semantic similarity used before during clustering in that the closeness between nodes requires computing a similarity measure between *uneven sets* of labels rather than two single labels. Using concepts from graph theory mentioned in the previous section, we explain how this comparison is achieved.

In order to evaluate whether two activity nodes are linguistically describing the exact same activity, we observe the semantic similarities between the corresponding sets of activity labels

of each node. Formally speaking, given two activity nodes $\omega_i = (\bar{s}^i, \bar{e}^i, L^i)$ and $\omega_j = (\bar{s}^j, \bar{e}^j, L^j)$, we only compare the label sets L^i and L^j to generate a single semantic measure in the range $[0, 1]$ between two label sets. This computation is independent of the activity's start and end times since semantic similarity is computed using the linguistic labels only.

The goal is reduced to designing a similarity function $\lambda^L(L^i, L^j) \in [0, 1]$ which takes the labels sets of two nodes as inputs and computes a similarity score between them. Recall that we have previously defined a function $\lambda^l(l_i, l_j) \in [0, 1]$ which take two individual activity labels l_m and l_n and returns their semantic similarity value. Using this function, we first compute the pairwise similarities for each label from L^i with each other label from L^j to produce a similarities matrix similar to a bi-adjacency matrix, denoted by \mathbf{B}^s . This matrix is square shaped if the two lists L^i and L^j are of equal sizes or rectangular otherwise. An example activity's sample label sets and the bi-adjacency matrix information is shown graphically in figure 4.6. This graph closely resembles the weighted bipartite graph from graph theory mentioned before.

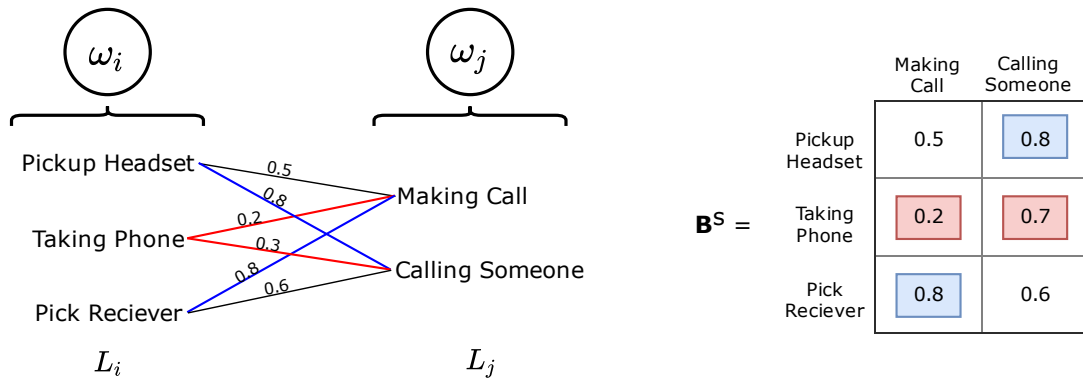


Figure 4.6: The example shows matching of two activity nodes using weighted bi-partite graph matching concepts. Inter-label similarity is computed using the off-the-shelf linguistic semantic similarity measure λ^l . The Hungarian algorithm is used to solve the assignment problem between the two node lists, where blue edges represent assigned edges. The red edges represent unassigned edges which are used to penalise the overall score.

From the bipartite matching formulation, we are able to find direct parallels with our specific label matching problem. Matching the two sets of labels is equivalent to matching the two sets of vertices in X and Y in graph G provided the bi-adjacency cost matrix B^c which encodes the cost between each pair of vertices from X and Y . Finding closely matching labels between the two label sets, independently of their position in the sets and the sizes of the label lists

thereby indicates high similarity between the label sets of two activity nodes. The two sets of labels in our problem definition are the two sets of vertexes in the bipartite graph i.e. $X = L^i$ and $Y = L^j$. The edges of the graph resemble the negative semantic similarity $w(\epsilon) = \lambda(l_i, l_j)$ of the labels that are represented by the two connecting vertexes of the edge ϵ .

We use the assignment problem formalism to match our label lists, but we also augment this mechanism further to account for our domain specific constraints. In our implementation, two amendments from the common formalism of the assignment problem are made. First, we capture the similarities rather than costs between the two sets of labels in a bi-adjacency matrix, we therefore simply take $B^c = 1 - B^s$ to transform the similarities to costs. Second, the traditional formalism of assignment problem assumes the two sets of vertexes to have equal cardinality i.e. $|X| = |Y|$, however, the two sets of labels from two activity nodes are highly unlikely to share the same size. This is simply because each activity is identified by any arbitrary and unbounded number of annotators. Moreover, the clustering process on the raw activity annotation labels further results in highly variable sizes of label sets between activity nodes.

Matching activity nodes with disparity between the number of labels of each node requires care when computing a similarity score. For example, matching a node with 15 labels with a node with 1 label may result in a single highly matched link between the lists. However, if the remaining unmatched links indicate a significantly different score than the matched link, then this disparity needs to reflect in the overall score since this difference is indicative of noisy labelling. Furthermore, an activity node having a high number of labels reflects a confidence level of that activity since this activity has been identified and labelled by many annotators. Conversely, an activity node which has few labels is likely to be spurious and therefore carries a low confidence value. We collectively encode these aspects within our similarity measure by considering not only highly matched labels within the two nodes but also consider the number of unmatched labels and influence the similarity measure with a weighting based on the relative sizes of the lists.

We perform this by using the output from the Hungarian algorithm to add in a measure of list sizes variability. As mentioned before, the Hungarian algorithm finds a bijective function $f : X \mapsto Y$ representing optimal assignments that minimise edge costs. We use this function to extract the subset of edges $\epsilon_a \subseteq E$ that connect vertexes $\{v_i, v_j\}$ which were assigned $f(v_i) = v_j$

by the Hungarian algorithm, seen as blue edges in figure 4.6. Formally the assigned edges are $\epsilon_a = \{\{v_i, v_j\} \in E | f(v_i) = v_j\}$. Moreover, we also extract the edges $\epsilon_u \subseteq E$ that connect the label vertexes $\{v_i, v_j\}$ which remained unassigned $f(v_i) = \phi$, seen as red edges in figure 4.6. Formally these unassigned edges are denoted as $\epsilon_u = \{\{v_i, v_j\} \in E | f(v_i) = \phi\}$. Given these two sets of edges, the set of active edges is defined as the set that contains both assigned and unassigned edges denoted as $E' = \epsilon_a \cup \epsilon_u$. Using these definitions of active edges that represent the assigned and unassigned vertexes along with the cost of assignments as edge weights, we propose the overall similarity between two activity nodes to be computed using equation 4.3.

$$\begin{aligned} \lambda^L(L_i, L_j) &= \frac{\frac{|\epsilon_a|}{|E'|} \sum_{e \in \epsilon_a} w(e) + \frac{|\epsilon_u|}{|E'|} \sum_{e \in \epsilon_u} w(e)}{|E'|} \\ &= \frac{|\epsilon_a| \sum_{e \in \epsilon_a} w(e) + |\epsilon_u| \sum_{e \in \epsilon_u} w(e)}{|E'|^2} \end{aligned} \quad (4.3)$$

Intuitively, this equation builds a semantic similarity measure between two activity node's label sets (L_i, L_j) based on highly matched labels from the two lists, represented in the first term in equation 4.3 ($\sum_{e \in \epsilon_a} w(e)$), and also influences the measure by adding a second term to reflect the unmatched labels ($\sum_{e \in \epsilon_u} w(e)$). The contributions of the two terms is judged by the ratios of the assigned and unassigned vertexes. Finally, the term is standardised by dividing by the total number of active edges in order to form a similarity score bound in the $[0,1]$ range.

Intersection of Activity Nodes

In this section we describe the computation of intersection of activity nodes used later in the chapter to infer *parent-child* relationships between nodes. Recall that an activity node ω is defined as a 3-tuple (\bar{s}, \bar{e}, L) where \bar{s} and \bar{e} are the start and end time of the activity and L is the label set of synonymous phrases describing that activity. The intersection between two activity nodes ω_i and ω_j is thought of as the positive temporal overlap between the intervals of the activity nodes i and j . This overlap is denoted by either $o(\omega_i, \omega_j)$ or o_{ij} and computed using equation 4.4. Note that if two intervals are disjoint, the temporal overlap is set to 0.

$$o(\omega_i, \omega_j) = \begin{cases} \min(e_i, e_j) - \max(s_i, s_j); & \min(e_i, e_j) > \max(s_i, s_j) \\ 0; & \text{otherwise} \end{cases} \quad (4.4)$$

Hierarchical Structures in Graph Theory

The set of activity nodes $\omega \in \Omega$ is simply a set of rich intervals embedded on a number line representing number of frames by their start and end time and label set, as shown in figure 4.5. In graph theory, such type of data is referred to as interval data. There exist graphical representations that are aimed at specifically abstracting interval data in literature. Abstraction of interval data refers to encoding the data in a graphical form which 1) abstractly represents the arrangement of these intervals, and 2) encodes the inherent hierarchical structure of the intervals. Relationships, such as overlap or starts, between intervals are often used within the encoding so that specific boundary details, which are start and end times, of the interval can be ignored. A popularly used representation to encode interval data is interval graphs [de Ridder et al.,]. An interval graph can be defined as a set of edges and vertexes as $G^I = (V^I, E^I)$. Each vertex corresponds to a single interval instance and an edge is set between every pair of intervals that *intersect* in time, see equation 4.4. The edges are defined as $E^I = \{\{v_i, v_j\} \mid v_i \cap v_j \neq \phi\}$. Figure 4.7 a) shows a sample set of intervals on a time line and their corresponding interval graph representation in figure 4.7 b).

Interval graphs have been previously used to represent intervals in the context of activity recognition. Authors in [Duckworth et al., 2016b] use interval graphs to represent activity interval along with their temporal order by embedding Allen's temporal relation along the edges of the graph. These graphs efficiently and effectively represent interval data in a concise graphical format, however, they lack any compositional information or hierarchical tree-like structure that defines *parent-child* relation between intervals. A richer graphical form is therefore needed to encode these relations. A subclass of interval graphs is known as trivially perfect graphs [Golumbic, 1978]. These graphs impose a stricter condition on creating edges between the vertexes. An edge is only set between two intervals if one is fully subsuming the other as opposed to only interaction (positive overlap), formally written as $E^I = \{\{v_i, v_j\} \mid o(v_i, v_j) = |v_i| \text{ or } |v_j|\}$. A trivially perfect graph can be used to capture naturally occurring hierarchical structures in

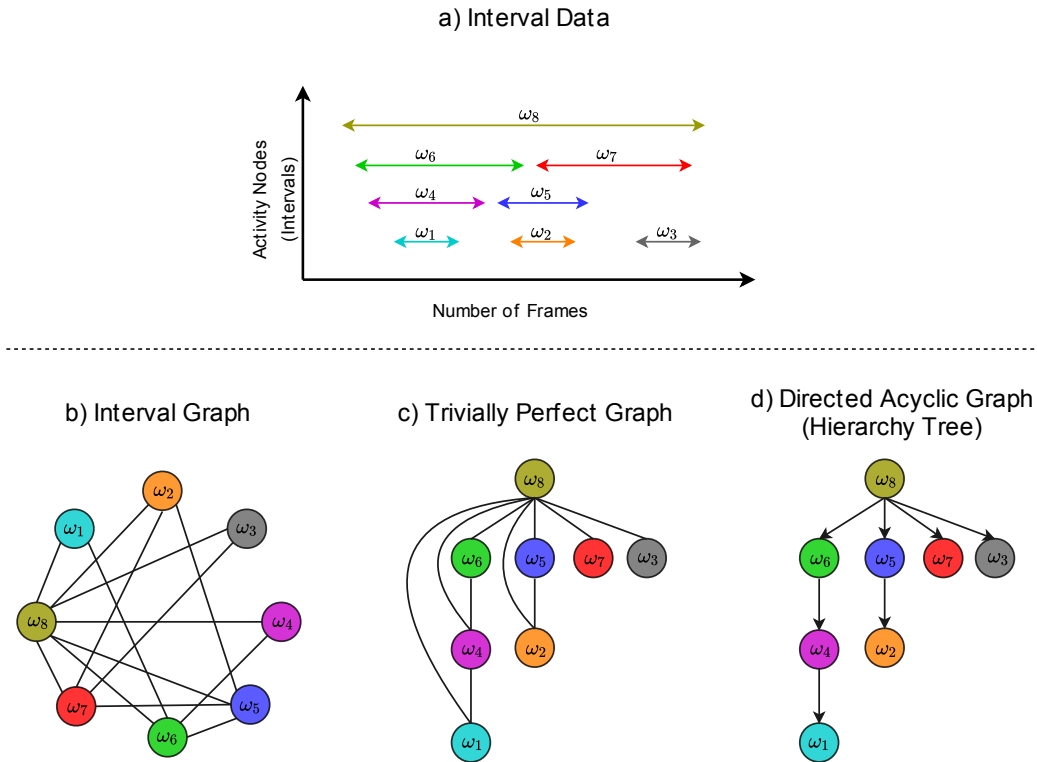


Figure 4.7: Sample interval data shown in a). The corresponding interval graph representation of the data shown in b). The trivially perfect graph in c) represents the interval data in a hierarchical format based on full subsumption criteria. d) The directed acyclic graph is a transitive reduction of the trivially perfect graph. This graph most closely represents the activity hierarchy structure we aim to learn.

interval data as shown in figure 4.7 c). Finally, this graph can be simplified by removing redundant edges using a transitive reduction algorithm [Aho et al., 1972] into a directed acyclic graph, figure 4.7 d). Note that nested intervals are connected to their immediate fully covering intervals by directed edges progressively resulting in a natural structure of a hierarchy.

Inspired by these models from graph theory, we find that such structures are analogous to human activities where an activity often comprises of a sub-activity and each of those further comprise of sub-sub-activities and so on. This arrangement seen in a nested activity interval data somewhat similar to our aggregated annotation activity nodes $\omega \in \Omega$. We therefore, arrange our interval activity nodes as a trivially perfect graph and perform a transitive reduction to produce a hierarchical tree; however, differently from the edge conditions imposed in graph

presented above, we relax the constraint of full subsumption and allow for both full and partial subsumption when generating the edges. More details on this choice is covered in the next section.

We propose that for an activity node ω_i to be a *candidate child* of an activity node ω_j , the following two factors need to be jointly considered:

1. Activity node ω_i temporally subsumes or significantly overlaps ω_j . We allow partial overlap along with full subsuming of intervals when deciding the relation since always observing fully subsumed intervals requires noise free annotations of temporal boundaries. This is rare when different annotators are annotating the same video. Note that, while a parent could have many children, only one activity node ω_j^* will be the actual parent of some ω_i .
2. The two activity nodes ω_i and ω_j must have semantically-similar labels. Recall that each activity node is represented by start and end time of the activity and a set of labels (\bar{s}, \bar{e}, L) . Comparison of the two sets of linguistic labels of two different nodes is explained in detail in section 4.4.3. This is an important condition that ensures that two activity nodes are semantically similar to confirm that parallel activities, that share the same temporal time frame, are correctly paired with their respective parent nodes. This logic derives from the fact that children activities of a certain parent activity would share higher semantic similarity between their labels than an activity which does not belong to this parent. For example, consider activities *reading newspaper* consisting of sub-activities such as *flipping page*, *closing sheet* etc. and *having breakfast* with its sub-activities such as *consuming coffee*, *eating sandwich* etc. Though both these activities are occurring in parallel with their intervals overlapping each other's, the resulting hierarchy produced would define each of the activity with only its relevant children.

Inferring the *Parent-Child* Relationship between Activity nodes

In this section, we use the tools mentioned in previous sections and describe our approach to infer the *parent-child* relationship between semantically similar activity nodes based on their temporal information i.e. start and end frames. Recall that for each activity node $\omega(\bar{s}, \bar{e}, L)$, the

parent-child relationship is estimated not only through semantic similarity of the label sets, as described in the previous section, but also through their start and end times. It is reasonable to assume that any child activity is most likely to occur during its parent activity's time frame and thus the child activity's time interval must be significantly subsumed by the parent activity's interval. Using such reasoning, we infer *parent-child* relationship between all pairs of activity nodes allowing us to generate an activity hierarchy.

Though this closely resembles the creation of a trivially perfect graph, there is one important issue that arises when using real-world acquired data. From our datasets, it is noted that the temporal boundaries of activities are distorted from 1) averaging across multiple instances of identified activity labels through clustering and 2) subjective nature of temporal edges of activities since different annotators identify different edges for the same activity, see figure 4.5. Determining *parent-child* relations between different activity nodes based on full subsumption of intervals, as is the case in trivially perfect graphs, becomes insufficient since many child activity nodes exhibit intervals that may only be partially subsumed by their respective parent node's intervals. We solve this problem through introducing an optimisation formalism which evaluates all partially and fully subsuming parent nodes for each child nodes and finds the optimal set of relationships resulting in an optimum activity hierarchy.

Formally, to find the optimal parent activity node, denoted by ω_p , for a child node, denoted by ω_c , temporal overlap between their respective intervals is considered for the *semantically-relevant* candidates. A cost function $\mathcal{C}(\omega_p \triangleleft \omega_c)$ as shown in equation 4.5, is used to gauge the suitability of assigning an activity node ω_p as a parent activity of an activity node ω_c .

$$\mathcal{C}(\omega_c \triangleleft \omega_p) = \begin{cases} \frac{||\omega_p||^2}{o_{pc} (||\omega_p|| - ||\omega_c||)} & \text{if } ||\omega_p|| > ||\omega_c|| \text{ and } o_{pc} > 0 \\ \infty & \text{otherwise} \end{cases} \quad (4.5)$$

$||I_*||$ denotes the length of any activity node interval, i.e $\bar{e} - \bar{s}$, and o_{pc} denotes the quantitative measure of overlap/intersection between any two activity node intervals computed using the previously defined equation 4.4. A set of preconditions filter activity nodes by assigning non-infinite costs to only the candidate activity nodes qualifying the conditions:

1. Each candidate parent node is within the temporal frame of the child node i.e. positive

overlap,

2. Each candidate parent node is larger in length than child nodes i.e. length of the competing parent activity node is greater than the length of child activity node $||\omega_p|| > ||\omega_c||$,
3. Each candidate parent node shares high linguistic semantic similarity computed using the previously defined activity node label lists comparison measure, see section 4.4.3, $\lambda^L(L_i, L_j) > \hat{\lambda}^L$ where $\hat{\lambda}^L$ is a parameter which decides the similarity allowance of two activity node label lists. The application of this semantic similarity filter enables excluding ‘noise’ nodes that bear no semantically valid connection to the main activity (e.g., ‘scratching head’, ‘fidgeting with pen’, etc.). At the same time, it allows separating parallel activities to individual hierarchies.

If these preliminary conditions are not met, an infinite cost is assigned which effectively disqualifies shorter and disconnected activity nodes as potential parent activities of a child activity node. Given that the preliminary conditions are met, we compute a non-infinite cost using the function presented in equation 4.5 to determine the suitability of activity node ω_p being assigned as the parent activity of ω_c . To better explain the components of this function, we present an expansion of the function in equation 4.6.

$$\begin{aligned} \frac{||\omega_p||^2}{o_{pc}(||\omega_p|| - ||\omega_c||)} &= \frac{||\omega_p||}{o_{pc}} \div \frac{||\omega_p|| - ||\omega_c||}{||\omega_p||} \\ &= \frac{||\omega_p||}{o_{pc}} \div \left(1 - \frac{||\omega_c||}{||\omega_p||}\right) \end{aligned} \quad (4.6)$$

For every child activity node, each *qualifying* parent node is quantitatively assigned a cost computed using equation 4.6 indicating their suitability as the parent activity. Figure 4.8 illustrates the intuition behind the various terms used in the cost function.

Figure, 4.8a diagrammatically illustrates the different terms used on an interval diagram. Figures 4.8b, 4.8c and 4.8d show example interval scenarios where an activity child node ω_c is assigned to one of the competing parent nodes ω_{p1} , ω_{p2} or ω_{p3} . The nodes highlighted in red show the most suitable parent since that assignment minimises the overall cost. These sub-figures further illustrate the contributing terms within the cost function presented above the

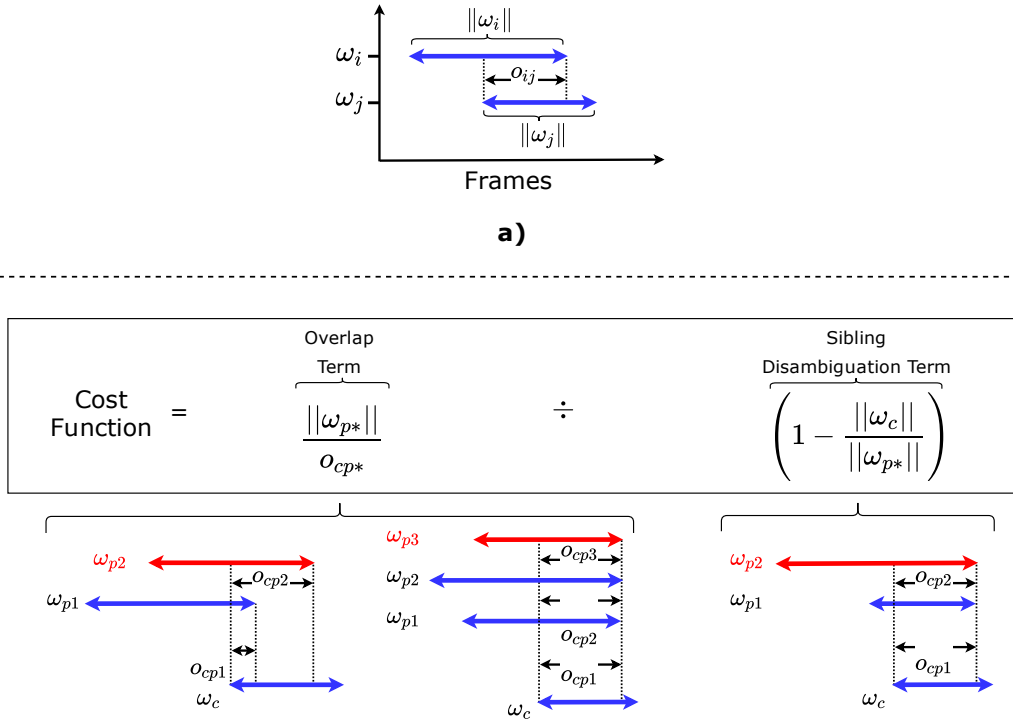


Figure 4.8: Justification of the terms of the cost function are shown in this figure. Notation used to represent overlaps and lengths between any two interval ω_i and ω_j shown in a). The cost function is presented along with sample examples justifying each term. ω_c denotes a child activity interval and ω_{p*} denote all candidate parent activity nodes for that child. The red parent intervals show the correctly chosen interval as the parent for each of the child nodes in the three example arrangements.

illustrations. The optimisation is deigned using **three** intuitive criterion to judge the *parent-child* relationship between intervals. We describe each of these three criterion next.

1. **Large overlap:** The first criterion for estimating a *parent-child* relationship is high overlap. This is a reasonable assumption as intervals with larger overlaps are assumed to be better suited parent intervals than those with low overlaps since low overlap terms may likely be due to noisy boundaries. Note that the overlap term of the cost function contributes to this as seen in figure 4.8b. The amount of overlap between the competing parent nodes ω_{p1} and ω_{p2} and child node ω_c is computed, and the function prefers parent node ω_{p2} since it exhibits a larger overlap, i.e. $o_{cp2} > o_{cp1}$, with the child node ω_c and thus produces a lower cost for the overlap term.

2. **Shortest parent node:** The second criterion is to select the shortest interval parent node from a set of parent nodes with increasingly larger intervals. This is a desirable property to ensure generation of a progressive hierarchy through the optimisation process. Whilst ensuring a good overlap with parent nodes, selecting the immediate next shortest parent node is also intended. In figure 4.8c, note that ω_c has equal overlap with all competing parent nodes. In such a case, using only the overlap amount o_{cp^*} within the overlap term in the cost function assigns equal costs to all parents ω_{p1} , ω_{p2} and ω_{p3} resulting in a random assignment of the parent node to the child node ω_c . A more desirable behaviour is to assign the immediate next shortest parent node to the child node. The length of each parent node $||\omega_{p^*}||$ is therefore also considered as the numerator in the overlap term, which when minimised yields the lowest cost. This is seen in figure 4.8c as ω_{p3} is selected as most suitable parent since this assignment results in the lowest cost amongst all competing parent nodes in this scenario.

3. **Disambiguate concurrent sibling activity nodes:** Selecting the immediate next shortest node as the parent *strictly*, as explained in the previous criteria, causes confusion between parallel activities and parent activities. Parallel activities, considered as sibling activities of a child activity node ω_c , belong to the same parent and are observed at the same time as ω_c . For example, ‘having breakfast’ may comprise of ‘preparing cereal’ and ‘reading newspaper’ occurring concurrently. However, they share a sibling relationship rather than a *parent-child* relationship. Typically, there tends to be a large overlap between parallel sibling activities and in cases where one sibling activity has a larger interval length than the other, that sibling qualifies as a potential parent candidate based on the predefined conditions from equation 4.5. Figure 4.8d shows this scenario whereby activity node ω_{p1} is likely to be a sibling activity of ω_c and ω_{p2} is more probable to be a parent activity of ω_c . We assert that an activity that has a similar length and high overlap to the child activity is likely a parallel sibling activity rather than a parent activity which tends to usually be significantly larger in length than its children activities. The sibling disambiguation term is therefore introduced, considering the relative lengths between the child activity node and candidate parent activity nodes $\frac{||\omega_c||}{||\omega_{p^*}||}$. The aim of this term

is to 1) yield a low value and approach zero as the potential parent and child activity nodes reach similar interval lengths and 2) remain in the range $[0,1]$. Note that this sibling disambiguation term is a denominator in the overall cost function and therefore low values of this term would produce high costs hence avoiding assignment of parallel sibling activities as parent activities. Figure 4.8d further shows that significantly longer ω_{p2} activity from the child node ω_c is correctly selected as the parent activity through the use of this term.

Hierarchy Generation

Equipped with a cost measure of assigning an activity node as a parent of another activity node, we are now able to generate the hierarchy tree of activity nodes denoted by H . Note that this tree reflects the hierarchical structure of activity nodes within one video. We achieve this through optimal discovery of *parent-child* relationships between activity nodes. The hierarchy $H = (\Omega, E^\Omega)$ is defined as a graph comprising of the set of activity nodes resembling vertexes Ω and a set of *parent-child* relationships between the nodes resembling the **directed** edges E^Ω i.e. (ω_i, ω_j) . As mentioned before, this hierarchy tree is similar to the directed acyclic graph shown in figure 4.7d).

By computing $c_{ij} = \mathcal{C}(\omega_i \triangleleft \omega_j)$ for all possible pairs of nodes whereby for each i^{th} node, there are a set of qualifying j nodes as potential candidates with positive overlap and larger interval length. An edge (ω_i, ω_j) is established where $\omega_i \triangleleft \omega_j \iff j = \arg \min_{1 \leq j \leq N} c_{ij}, \forall i$, breaking ties randomly. Note that the highest-level activity nodes would compute a cost of infinity with all other candidates through this optimisation. In these cases, the highest-level nodes $\Omega^r \subseteq \Omega$ are paired with an extra parent node denoted as ‘root’. This is done by setting edges $(root, \omega_i) \forall i \in \Omega^r$. A ‘root’ node is always included in any hierarchy denoting the highest level activity.

4.4.4 Encoding Temporal Ordering of Sub-activities

The final step in creating a complete activity hierarchy from an activity video is to encode some notion of the temporal order of the various sub-activities within the model. We have

mentioned before that the temporal order of the composing activities is a highly discriminative and representative feature of a complex activity. For example, a ‘consuming tea’ activity will primarily comprise of ‘making tea’ sub-activity observed *before* the ‘drinking tea’ sub-activity. In each of our hierarchy trees H , each parent activity node connects to its multiple children activity nodes. The immediate child activity nodes of a parent node can be thought of as a (partially) ordered set of *siblings* or the set of comprising activities of that parent. We encode the temporal ordering information amongst these sibling activities to represent the order in which child activities are observed for the parent activity. Performing this encoding for each parent activity node allows to transform each hierarchy H into a temporalised hierarchy \tilde{H} representing the complex activity fully equipped with temporal knowledge of all comprising activities.

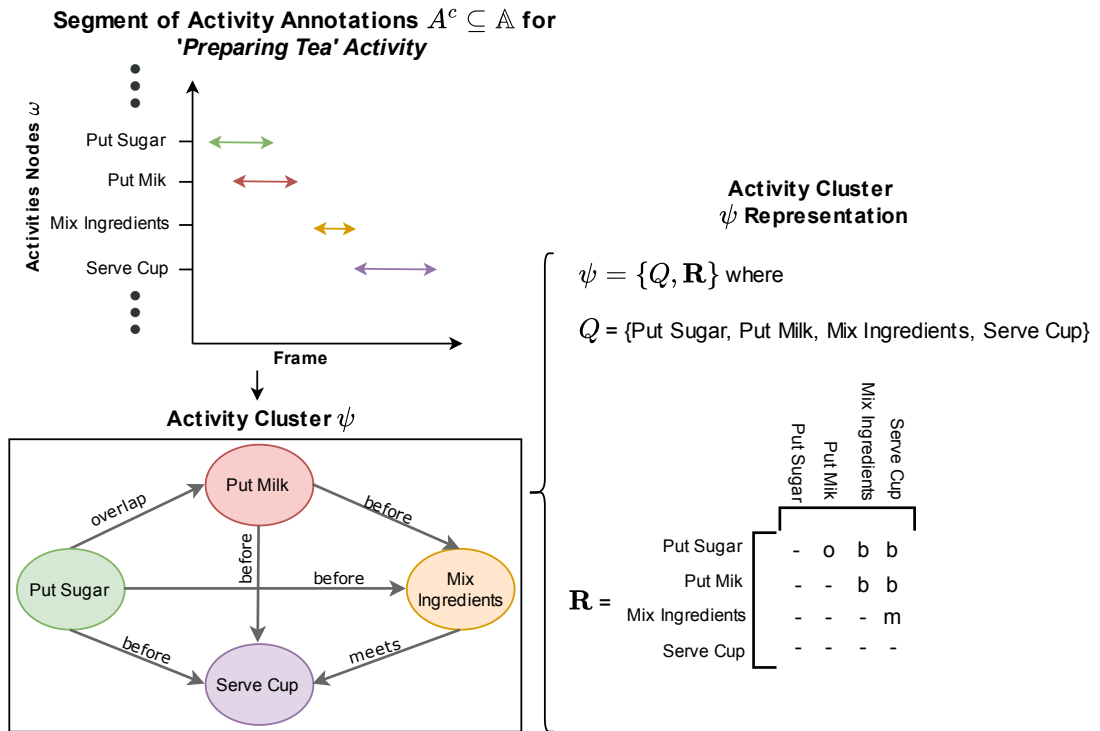


Figure 4.9: This figure represents a sample set of sibling activities for the ‘preparing tea’ activity. The corresponding activity cluster denotes the set of activity nodes with their temporal information. The is represented using an interval graph shown in graphical form (left) and in notation (right). Note that, for simplicity, only one label is shown within each activity node. In real-data, multiple labels comprise an activity node.

We employ the popular qualitative encoding of interval data known as Allen’s Temporal Algebra [Allen, 1983]. Allen’s temporal algebra has been introduced in chapter 2. This representation introduces categorical temporal relationships, between any two intervals, that abstracts away exact quantitative durations of intervals and encode the configurations of intervals in time using one of IA relationships. In this work, we take inspiration from activity graphs presented in [Duckworth et al., 2017, Duckworth et al., 2016a] to encode the temporal order of sub-activities in a similar fashion. Interval graphs [de Ridder et al.,] were introduced in section 4.4.3, where each activity node is set as a vertex in the graph and the presence of an edge indicates positive temporal overlap, or intersection, between the adjacent activity nodes. Authors in [Duckworth, 2017, Duckworth et al., 2016a] augment these interval graphs by transforming the edges into directed edges that carry a label that represents the IA relation which holds between the intervals of the adjacent activity nodes. Figure 4.9 represents an example sub-set of activity nodes from a given activity annotation $A^c \subseteq \mathbb{A}$ and the corresponding augmented interval graph. Note that ‘Put Sugar’ and ‘Put Milk’ intervals are related through a temporal *overlaps* relation, and that ‘Put Milk’ interval occurs before ‘Mix Ingredients’ thus uses the *before* relation. This is evident in the corresponding interval graph from the edge value connecting these two nodes. This technique is repeatedly applied to all pairs of activity nodes to create a complete interval graph. We denote this graph as an activity sibling cluster $\psi = \{Q, \mathbf{R}\}$ where Q is a set of the cluster’s activities, and $\mathbf{R} : Q \times Q \rightarrow \{\textit{meets, overlaps, before, ...}\}$ is a matrix which captures the pairwise Allen’s temporal relation between the members of Q .

Algorithm 2 Temporal Knowledge Augmentation

```

1: Input: Hierarchy Tree  $H = \{\Omega, E^\Omega\}$ .
2: Output: Temporalised Hierarchy Model  $\tilde{H} = \{\Pi, \Phi\}$ .
3: Initialise  $\Pi = \emptyset$ . ▷ Empty Set of Parent Nodes
4: for all  $\omega \in \Omega$  do
5:   if  $\omega$  is not a leaf node then
6:     Append  $\omega$  to  $\Pi$ . ▷ Add to Set of Parent Nodes
7:      $A^c = \{\omega_c \in \Omega \mid \exists \{\omega, \omega_c\} \in E^\Omega \forall c\}$ . ▷ Children Nodes of  $\omega$ 
8:      $\psi =$  Interval Graph  $\{A^c, E^c\}$  [Duckworth et al., 2017]. ▷ Activity cluster  $\psi$  of
       children nodes  $A^c$ 
9:      $\Phi(\omega) = \psi$ . ▷ Maps the activity cluster  $\psi$  to its parent  $\omega$ 
10:   end if
11: end for
12:  $H = \{\Pi, \Phi\}$ .

```

Algorithm 2 formally presents our approach to perform the temporal encoding of each hierarchy into a temporalised hierarchy. A hierarchy tree $H = \{\Omega, E^\Omega\}$ is taken as input and a temporalised hierarchy \tilde{H} is generated. For each parent node in the H , the set of its immediate children nodes (siblings), are encoded into an activity cluster ψ (augmented interval graph) using Allen’s temporal algebra as previously described. For a single parent node ω , this activity cluster conceptually represents the set of composing activities of ω . Their temporal order is encoded along the edges between each pair of activity nodes. Lastly, a function $\Phi : \Omega \mapsto \Psi$ maps the parent node to its new activity cluster. This encoding is performed for each parent node in the tree. The resulting model, called a temporalised hierarchy \tilde{H} , consists of all parent nodes Π along with their activity clusters mapped by Φ . Figure 4.10 shows a sample hierarchy H produced from the ‘making a call’ example, see figure 4.3 annotation clusters, and augmented with temporal knowledge using algorithm 2 to produce a temporalised hierarchy \tilde{H} of the activity for a single video of ‘making a call’.

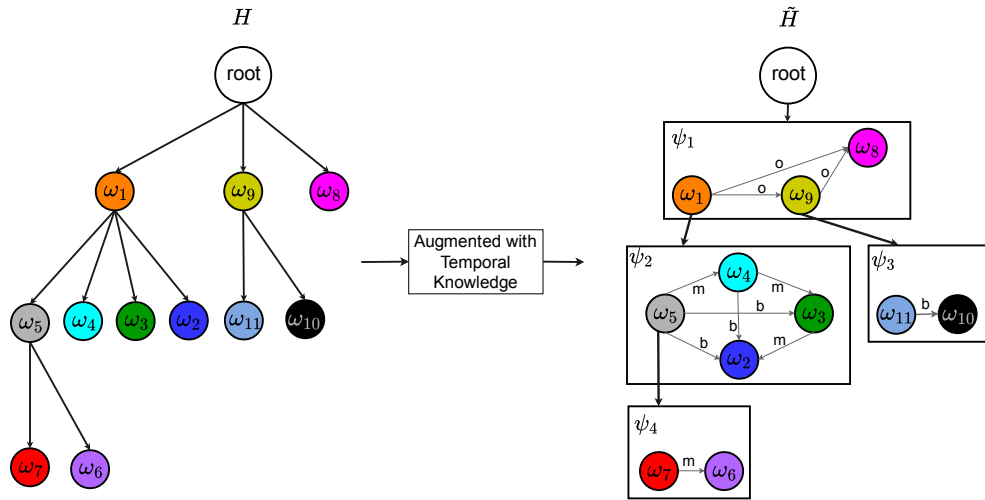


Figure 4.10: This figure presents an example hierarchy tree, and the temporalised hierarchy. Each set of siblings (nodes on the same level) have been converted into interval graphs encoded with Allen’s temporal relations. Note that these hierarchies represent the activity nodes learned from the ‘using the phone’ example in figure 4.3.

4.5 Abstracting a Generalised Activity Model

Given a set of $V^{Train} = \{v_1, v_2, \dots, v_N\}$ training videos of a top-level activity, we learn a temporalised hierarchy \tilde{H} for each training video $v \in V^{Train}$, to produce a corresponding set of temporalised hierarchies $\mathcal{H} = \{\tilde{H}_1, \tilde{H}_2, \dots, \tilde{H}_N\}$. The aim of any machine learning algorithm is to learn a generalised and abstracted model from training data provided. In our work, we treat each temporalised hierarchy \tilde{H}_i as a training instance of the top-level activity. This is true since each temporalised hierarchy $\tilde{H} \in \mathcal{H}$ abstractly represents the tree of activities describing the top-level activity in a single training video. Conceptually each temporalised hierarchy represents one variation of performing the top-level activity as observed from the training video. To construct an effective activity recognition system, naturally occurring variation in activities must be captured through the learning process. Therefore, in this section, we present a method inspired by the natural language processing (NLP) literature for abstracting a generalised activity model from different ‘training’ temporalised hierarchies learned in previous sections. This generalised activity model abstractly represents:

1. The hierarchical structure of the top-level activity.
2. The temporal order of all constituent sub-activities.
3. A probability distribution over all the variations for each parent activity, as observed from the training data.

4.5.1 Grammar Induction in Natural Language Processing

Natural Language Processing (NLP) is an area of research which focuses on enabling artificially intelligent systems to understand and analyse natural language in text or speech to perform useful tasks [Chowdhury, 2003]. NLP is used in many different fields such as machine translation, text processing, summarisation etc. One main aim of NLP is language understanding, allowing computers to understand the high level meaning of sentences. Language understanding requires an artificial system to learn many different aspects of language such as pronunciation, meaning of words, parts of speeches, semantic meaning etc. [Liddy et al., 1999]. One such aspect is the syntactical understanding of sentences. This aspect deals with learning the grammatical struc-

tural of sentences. The syntactic structure of sentences, much like activities, is compositional i.e. a sentence grammatical structure can be broken down into sub-components, each of which might be further broken down until each word is explained. A string of words is converted into a tree-like diagram which represents the grammar as a hierarchical structure [Mitchell, 1994]. Such a hierarchical syntactical structure is called a parse tree; a sample parse tree is shown in example 4.11. Note that at the lowest level each word is described by a part-of-speech for example, noun (N), verb (V) etc. Combinations of those descriptions compose a higher level description for example, a preposition (P) and a noun (N) collectively form a prepositional phrase (PP). This process repeats until the root sentence (S) is composed and the resulting parse tree fully describes the syntactical structure of the sentence as a hierarchy. The parse tree comprises of rules that describe the composition of each *symbol* (non-terminal or word) as a set of terminals or non-terminals in the tree. These rules, also known as production rules or re-write rules, can be read off the parse tree and written as a grammar set as seen in figure 4.11.

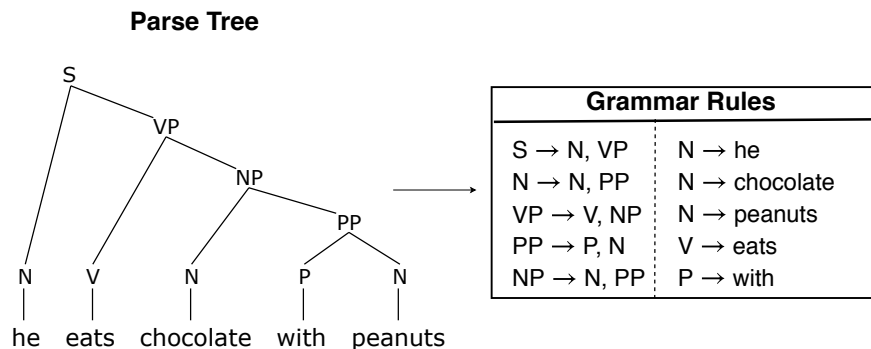


Figure 4.11: Example of input sentence parsed using the language grammar shown on the right.

Assuming that the parse tree is expertly annotated, we can assert that the grammar rules derived from this tree are generalisable to explaining other sentences. From this assertion, given many instances of different parse trees, learning a single generalised grammar which abstractly represents the syntactical knowledge of an entire language, is a problem in NLP known as grammar induction. Note that a parse tree closely resembles our activity hierarchy, whereby each activity node is described by an activity cluster which is conceptually an expansion of that activity. Therefore, the problem of automatic grammar induction given many parse

trees, translates well into the problem of learning a generalised hierarchical model given many temporalised activity hierarchies. We now explain a simple grammar induction technique which is then slightly augmented in order to learn our generalised hierarchical model for activities.

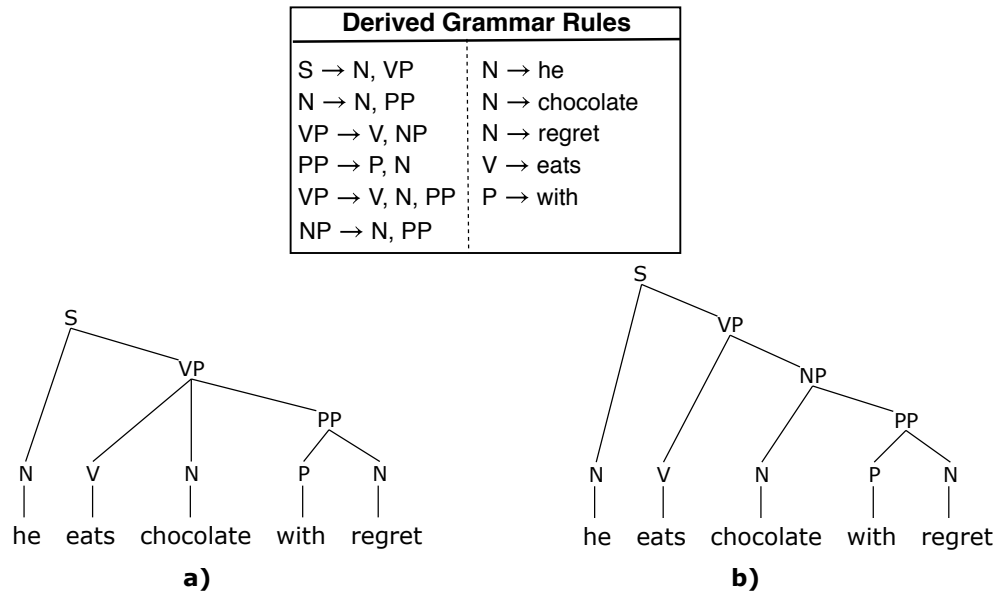


Figure 4.12: Example of ambiguity in parsing sentences using grammar rules. Parse tree a) represents the correct parsing of the words where ‘with regret’ is correctly identified as a prepositional phrase (PP) relating to a verb (V). Parse tree b) shows that ‘chocolate with regret’ is all a single noun phrase (NP), which is an incorrect classification.

One well known method of grammar induction is through utilising a tree-bank. A tree-bank is a set of hand constructed parse trees by expert linguists, such as the Penn tree-bank [Marcus et al., 1993]. The simplest way to learn a tree-bank grammar is by reading-off grammar rules from all the corpus parse trees and removing repeated rules [Charniak, 1996, Prescher et al., 2006]. This naive approach performs poorly when handling natural ambiguity that occurs within language. Ambiguity refers to uncertainty in meaning of a sentence, resulting in multiple parse trees that fit a single sentence however not all fitted parse trees are grammatically correct. An example of ambiguity is illustrated in figure 4.12. Note that, from the derived grammar rules at the top of the figure, two parses of the same sentence are presented. However, comparing to the sentence example parse tree presented in figure 4.11, the ‘chocolate with regret’ in this example is being treated incorrectly as a collective noun in the parse tree in figure 4.12b. The correct parse of this sentence is seen in figure 4.12a. In order to resolve such ambiguity,

the computational linguistics literature proposes the use of probabilistic grammar rules [Black et al., 1993, Salomaa, 1969, Charniak, 1996, Prescher et al., 2006]. For each rule in the derived grammar, a probability is learned according to how often that rule is observed in the training corpus. Thereby ambiguity is resolved by introducing a bias towards the ‘most often’ observed grammar rules as they carry higher probabilities. Learning these probability distributions across all expansions of grammar symbols is an ongoing research problem. Many methods have been proposed to automatically learn these probability parameters within a grammar, for example the inside-outside algorithm [Pereira and Schabes, 1992].

Here we adopt one of the simplest methods of learning the probability distributions using maximum likelihood estimation (MLE). Given a set of rules in a grammar \mathcal{G} , we can define each rule as $(J \rightarrow K)$ where J denotes any of the grammar symbols in the parse tree which are non-terminals or non-leaves and K is a set of symbols that J comprises of. These symbols can be any combination of terminals or non-terminals. The probability of a rule $\rho(J \rightarrow K)$ is simply the frequency of observed expansion of that rule divided by all expansions of the left hand side (*lhs*) of that rule. Equation 4.7 formally presents this computation.

$$\rho(J \rightarrow K) = \frac{\text{count}(J \rightarrow K)}{\text{count}(J \rightarrow *)} \quad (4.7)$$

From equation 4.7, it follows that $\sum \rho(J \rightarrow *) = 1$ which denotes that probabilities of all expansions of *lhs* symbol must equal 1 in the rule set. Furthermore, equation 4.8 shows the computation of the probability of a complete parse tree, denoted by $p(\tau)$, is the product of the probabilities of all rules used within the parse tree.

$$\rho(\tau) = \prod_{J \rightarrow K \in \tau} \rho(J \rightarrow K) \quad (4.8)$$

4.5.2 Building General Activity Models using Grammar Induction

Inspired by grammar induction methods from the NLP literature described in section 4.5.1, in this section we present our method to learn a generalised hierarchical model from a set of temporalised hierarchies $\tilde{H} \in \mathcal{H}$. Each hierarchy is closely related to a parse tree and learning a generalised hierarchical model of activities is equivalent to learning a grammar from parse

trees. However, the following extensions are required to be made to the learning process from context of natural language to activity hierarchy domain.

1. **Node Comparison:** Symbols in language grammars and parse tree are consistently represented as the same set of letters, for example ‘S’ or ‘NP’ etc. Comparison of different symbols in this case is trivial. In our temporalised hierarchies, the equivalent entity to symbols are activity nodes, which are represented as a set of multiple labels. Comparison of activity nodes requires a more complex method than language symbols, as proposed in section 4.4.3. We therefore use the previously described similarity metric to establish a measure of closeness between activity nodes rather than exact matching.

2. **Activity cluster Comparison:** The expansion of a rule read from a parse tree comprise of a set of sequential symbols; for example ‘S’ may expand to ‘V NP PP’. Expansion of a corresponding activity node in a temporalised hierarchy is represented as an activity cluster rather than a string of symbols, see figure 4.7. Recall that these activity clusters comprise of an augmented interval graph of activity nodes rather than simple sequence of symbols. Unlike exact comparison of expansions of symbols in grammar performed by simple string matching, we require a comparison method that compares interval graph structures within two activity clusters and establish their equality. We design a novel measure to perform this comparison which is similar to the activity node matching. In activity node matching, pairwise scores are computed amongst uneven lists of language labels using the method described in section 4.4.3. Activity clusters are correspondingly a set of uneven activity nodes. We use the activity node matching measure to generate a similar bi-adjacency matrix encoding similarities between the activity nodes of the two clusters. Given this matrix, we can use the same method in equation 4.3 to compute a score. However, one key difference when matching activity clusters, is the presence of temporal edges between nodes within each cluster. We are able to measure a distance amongst the temporal edges between nodes in the two clusters by comparing the relationships they carry. Recall that for temporal relations, we use the relations of Allen’s temporal logic, $rel = \{before, meets, overlap, starts, during, finishes, equal\}$ and their inverses. A distance between them is defined as the distance between any two relations,

to be the separation in the list, normalised by the number of relations. This measure along with the node similarity score define our measure for cluster matching.

3. **Leaf nodes:** Leaf nodes in a parse tree refer to words of a sentence. At the first level of a parse tree, words are first tagged by their parts of speech symbols. These symbols exist in the syntactical rules within the grammar and are then used to infer the remaining tree. In our temporalised hierarchies, since lowest level leaf nodes are implicitly simple and short activities, which are part of other activities, syntactical parsing can be performed directly without need for tagging.

Given the extensions mentioned above, we propose Algorithm 3 to learn a generalised hierarchical model of the overall activity from different training instances, as encoded in temporalised hierarchies, using a maximum likelihood estimation method similar to grammar induction described in section 4.5.1. The learned model is made probabilistic to allow for variation, uncertainty and robustness against noisy observations.

Algorithm 3 Abstraction of Hierarchies

```

1: Input: Temporalised Hierarchies  $\tilde{H}_a = (\Pi^a, \Phi^a)$  and  $\tilde{H}_b = (\Pi^b, \Phi^b)$ 
2: Output: Generalised Hierarchical Model  $\mathcal{M} = (\Pi, \Phi, P)$ 
3:  $\Pi := \Pi^a \cup \Pi^b$  ▷ Unique set of collective parent nodes
4: for all parent activity nodes  $\pi \in \Pi$  do
5:    $\Psi :=$  Concatenate  $\Phi^a(\pi)$  with  $\Phi^b(\pi)$  ▷ Group of collective activity child clusters
6:   Initialise  $n := |\Psi|$  ▷ Number of clusters in  $\Psi$ 
7:   Initialise  $P^\pi : \Psi \rightarrow [0, 1]$ 
8:   for all child activity cluster  $\psi \in \Psi$  do
9:     Append  $\psi$  to  $\hat{\Psi}$ 
10:    Initialise  $p := 1$ 
11:    for each cluster  $\psi' \in \Psi - \psi$  do
12:      if  $\lambda^\psi(\psi, \psi') > \hat{\lambda}^\psi$  then ▷ Activity cluster comparison upto some threshold
13:         $p = p + 1$ 
14:        remove  $\psi'$  from  $\Psi$ 
15:      end if
16:    end for
17:     $P^\pi(\psi) = p/n$ 
18:  end for
19:  Add  $P^\pi$  to  $P$ 
20:   $\Phi(\pi) = \hat{\Psi}$  ▷ Map the new set of merged clusters to the parent node  $\pi$ 
21: end for

```

To illustrate this in detail, we present a sample example in figure 4.13. The algorithm takes

as input two training temporalised hierarchies, say \tilde{H}_a and \tilde{H}_b , and builds a generalised model by merging these hierarchies. The resultant model is then input back with a new training hierarchy to produce a further generalised model. This process repeats until all training hierarchies are incrementally used to build the final generalised hierarchical model. Note that the abstraction process is incremental allowing for real-time learning and abstraction of a *generalised hierarchical model* of the activity.

Recall that the set of temporalised hierarchies is $\mathcal{H} = \{\tilde{H}^1, \tilde{H}^2, \dots, \tilde{H}^N\}$ previously learned for all N training videos. Each temporalised hierarchy $\tilde{H} \in \mathcal{H}$ is represented as a 2-tuple, $H = (\Pi, \Phi)$. In line 3 of the algorithm, Π denotes the unique list of all the parent (non-leaf) activity nodes from the two input temporalised hierarchies. A unique list of activity nodes is obtained by performing activity node matching as described in section 4.4.3. Φ maps each $\pi \in \Pi$ to the set of children activity clusters ψ belonging to π . Ψ refers to the collective set of activity clusters for a parent node across the two temporalised hierarchies whereas $\hat{\Psi}$ refers to the abstracted set of activity clusters following cluster matching. The algorithm output a generalised model $\mathcal{M} = \{\Pi, \Phi, P\}$ which comprises of the abstracted parent activity nodes Π , their child activity clusters mapped by Φ and a new component P which maps a parent node $\pi \in \Pi$ to a probability distribution over its child activity clusters $\Phi(\pi)$. This distribution defines the likelihood of a child activity cluster, capturing the variation in the way a parent activity can be performed. From this, similar probability reasoning as linguistic grammar induction follows that $\sum_{\psi \in \Phi(\pi)} P(\psi) = 1$. This states that the probabilities across all clusters of each single parent node adds to 1. Note that in the generalised hierarchical model, parent activity nodes would connect to multiple activity clusters as opposed to temporalised hierarchies where only a single activity cluster expanded a parent node, thus capturing variation by introduction of ‘OR’ branches.

An example of building the generalised hierarchical model is illustrated in Figure 4.13. Starting at the *root* node and progressing onto any other parent nodes highlighted in red, child activity clusters are compared using the cluster matching measure (Line 12) described in the next section. Unique clusters are extracted as the children of new parent nodes in the merged model (Line 9). The probabilities of child activities are computed from the frequency of observing each compared to the total number of possible ways (Line 17) the activity parent

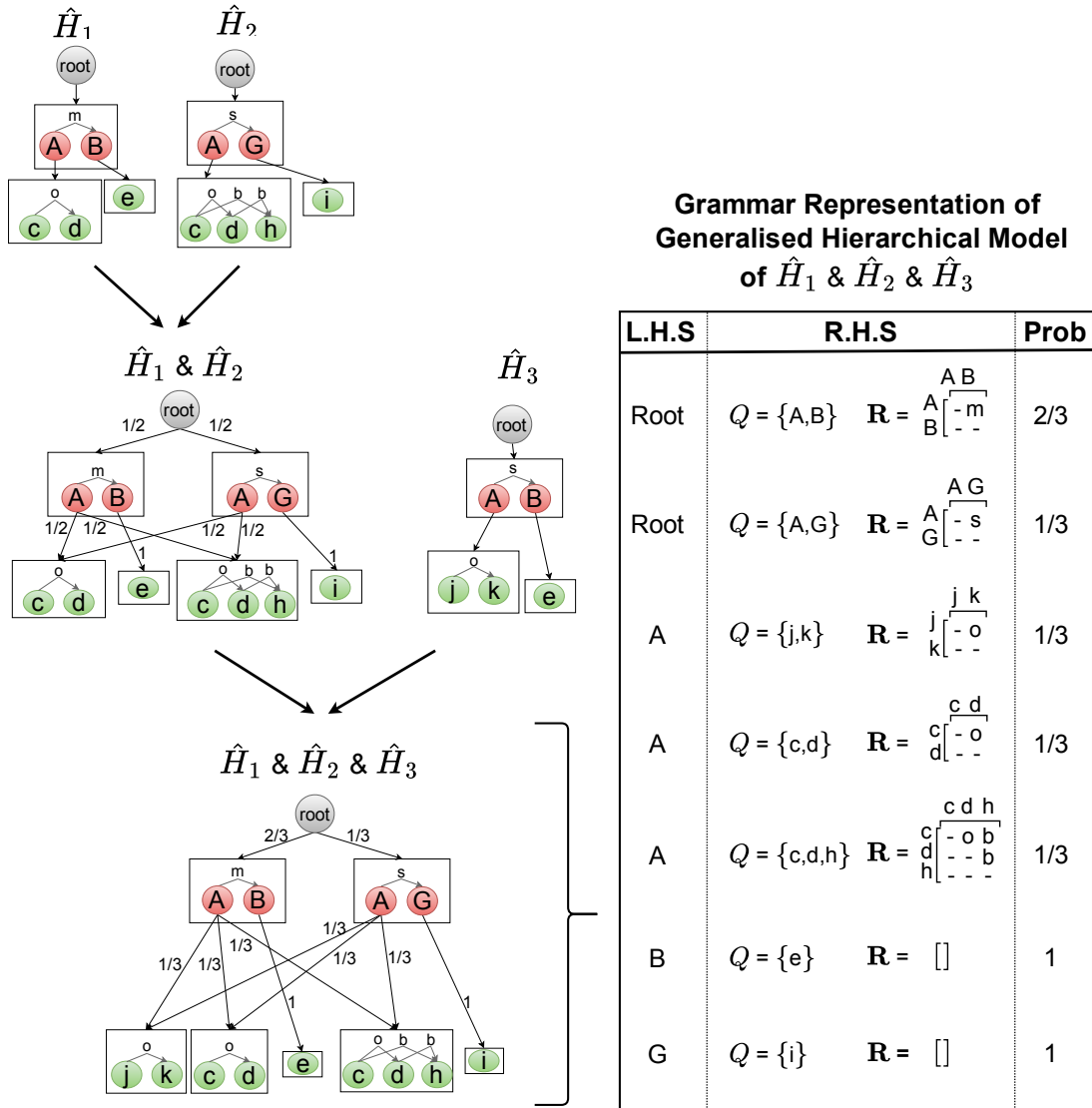


Figure 4.13: Example of the incremental abstraction of temporalised hierarchies shown. Starting at the root node, activity clusters are merged based on their similarity. Concurrently, probability values associated with each cluster are also learned from their frequency of observation. The final model is represented as an extended grammar where the right hand side of each rule is augmented with a relations matrix \mathbf{R} encoding the inter-symbol Allen’s interval relations between the elements of Q .

node can be described. The resulting model includes all unique children from both training hierarchies along with the probabilities of their occurrence based on their observed frequency during training. Note that multiple child clusters with the same parent denote the alternative

ways in which that parent activity could be observed with the associated probability, denoting an ‘OR’ relationship. The siblings within a cluster, though linked by temporal relations, denote an ‘AND’ relation. The resulting generalised mode is analogous to an ‘AND-OR’ graph and therefore can be rewritten as a language like grammar. The grammar transformation of the model is necessary to allow for recognition. As seen in the final part of figure 4.13, we rewrite the final abstracted model into a grammar like structure referred to as an *activity grammar* \mathbb{G} .

4.6 Recognition

The generalised hierarchical model learned captures the syntactical structure of complex activities along with their variation as seen from training data. To utilise the learned model, we must be able to recognise latent syntactical structure of the underlying activity at varying levels of granularity from a test video. In this section, we describe our approach to recognise the activity hierarchy in an unseen activity video.

Since the generalised hierarchical model is represented as an activity grammar, which closely resembles a language grammar, it is therefore logical to utilise well established and deeply researched parsing techniques that parse input sentences using a grammar model. Similarly, given the activity grammar, we can perform recognition of an activity hierarchy over an input action stream. Parsing in the context of linguistic grammars is a process of analysing a sentence represented as a sequential set of words, and recursively applying grammar rules describing the syntactical structure at different levels of granularity. Similarly, in our context of activity recognition, provided the generalised hierarchical model represented as an activity grammar, a set of low-level actions resembling words. These can be parsed to generate the inherent hierarchy of higher-level activities which best describe those actions. However, the key extensions described in section 4.5.2 also are needed to apply language parsing approaches that use traditional linguistic grammar, to activity hierarchy parsing using activity grammar. Firstly, an input stream of primitive actions is needed to build the remaining activity hierarchy over the video. This is described in the next section.

4.6.1 Low-level Action Recognition

In language parsing, the input to the parser is often a grammar model and a set of ordered words representing a sentence. If the grammar used lacks lexical rules, that encode vocabulary similar to those shown in the right column of the grammar in figure 4.11, then the sentence is first preprocessed to perform part-of-speech tagging where each word is tagged with its corresponding grammar symbol. This tagging process results in a sequence of primitive grammar symbols which are terminals in the grammar model and can now be parsed. Similarly, in our parsing mechanism, the input is a test video and an activity grammar model. We therefore first perform pre-processing on the video to recognise and extract the *lowest*-level actions i.e. the primitive actions which exist as *terminals* in our activity grammar model. The low-level recognition stage can be thought of as tagging video segments with the corresponding primitive action which occurs in that video segment. This process results in a temporally complex stream of terminals (primitive actions) which can now be parsed using our activity grammar model to generate an activity hierarchy tree over the input stream. Temporally complex refers to an input stream of actions which share complex temporal relations between them, such as ‘overlaps’, ‘during’ etc. This is different to language input streams which comprise of words that each pair of words share the same temporal relation ‘before’ i.e. each word follows the next, whereas actions can be observed in parallel.

Given the success of our QQSTR system ideal for simple activity recognition presented in section 3, we adopt this system here to perform primitive action recognition in our test videos segments. However, the QQSTR is a supervised learning system and therefore requires training data to learn low-level action classes. In our approach, we make use of our learned generalised activity model to extract low-level action classes and build training data for these classes. The generalised activity model is used to first identify terminals which are needed to be recognised, we assert that lowest-level activity nodes or leaf-nodes in the generalised activity model constitutes to low-level action classes. This is a valid assumption since our generalised hierarchy is constructed from human descriptions of activity hierarchy and therefore the lowest-level leaf nodes correspond to simplest and shortest activity segments as perceived and identified by human subjects. In the final activity grammar, these leaf nodes become

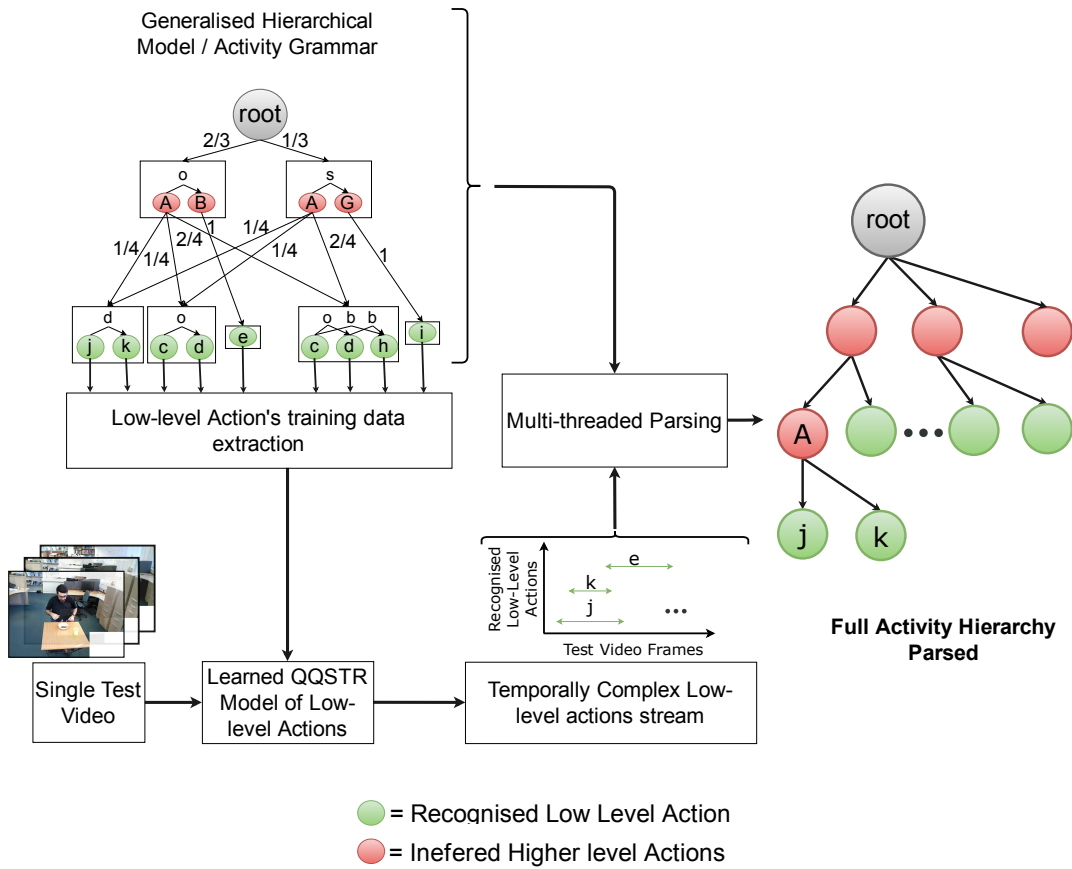


Figure 4.14: This flowchart represents the extraction of low-level action terminals from the generalised hierarchical model. The green nodes represent the terminals/leaf nodes. The classes of these nodes are learned from extracting data from raw videos. Given a single test video, these actions are classified to generate a temporally complex activity stream. Using the multi-threaded parsing method, the complete activity hierarchy describing the input terminals is recovered.

terminals which are required to be recognised and classified over a test video. Once leaf-nodes are extracted from the generalised hierarchical model, we then extract training video segments for each leaf activity node from the raw training video set. This training set of low-level activity classes is used to train a discriminative model (SVM) using our QQSTR approach, for low-level action recognition. Note that, though we utilise our QQSTR approach for low-level action recognition, our complex activity framework is capable of utilising any action recogniser to produce a temporally complex stream of low-level actions. The framework of this methodology is presented in figure 4.14.

Using this trained model, a set of terminal actions can be recognised and extracted from test video resulting in a temporally complex low-level action stream. We now describe the process to identify the latent high-level activity hierarchy structure over the low-level action stream within the test video. We refer to this inferred activity hierarchical structure over the test video as activity parse tree.

4.6.2 Extended Earley Parser for Activity Hierarchy Recognition

In this section, we explain how a set of low-level actions, recognised from a test video, can be used to infer the latent activity parse tree of all higher-level activities over the low-level action stream. As mentioned before, this task is similar to inferring a syntactical tree describing the grammatical structure over an input sentence. The process in the NLP literature is known as parsing and the resulting syntactical structure, usually a graphical tree, is known as a parse tree. The parsing problem has a rich history of research that has proposed many efficient and efficient parsing algorithms. The CYK algorithm [Hopcroft et al., 2001] and the Earley parser [Earley, 1970] are the two most prominent algorithms for parsing sentences using a context-free grammar. These algorithms take as input a sentence and a probabilistic grammar model and output the most likely parse tree which best describes the sentence. Whilst the CYK algorithm has time complexity of $\mathcal{O}(n^3)$, the Earley parser only takes $\mathcal{O}(n^3)$ in the worst-case, so on average perform better than the CYK algorithm. Earley parser is a type of chart parser. These parsers use dynamic programming by maintaining a chart of explored rules so that the explored sections do not need to be revisited. The Earley parser also performs parsing in a top-down fashion, where the grammar rules are observed and all probable parse trees, which conform to the input words, are explored. That is to say all possible parse structures over every contiguous subset of the input are explored and stored. We therefore adopt an extended version of the Earley parser from NLP literature for parsing activity trees over the activity grammar in our approach.

The classic Earley parsing algorithm is presented in [Earley, 1970]. Here, we illustrate the workings of the Earley parser on a simplified example of an activity grammar and an input stream of low-level actions. We then briefly describe the extensions proposed by [Zhang et al.,

2011] on the classic algorithm to allow for the parsing using the full activity grammar learned from section 4.5, using the temporally complex low-level action stream.

Earley Parser Example

Figure 4.15 presents a sample activity grammar, a sample input set of low-level actions and the resulting output from the Earley parser. Also a few parsing solutions over the input stream are provided. Let us suppose an activity comprising of the following activity nodes: having breakfast (BF), preparing tea (PT), preparing coffee (PC), consuming tea (CT), consuming coffee (CC), put sugar (ps), put tea-bag (pt), put water(pw), eat cereal (ec), drink tea (dt) and read newspaper (rn).

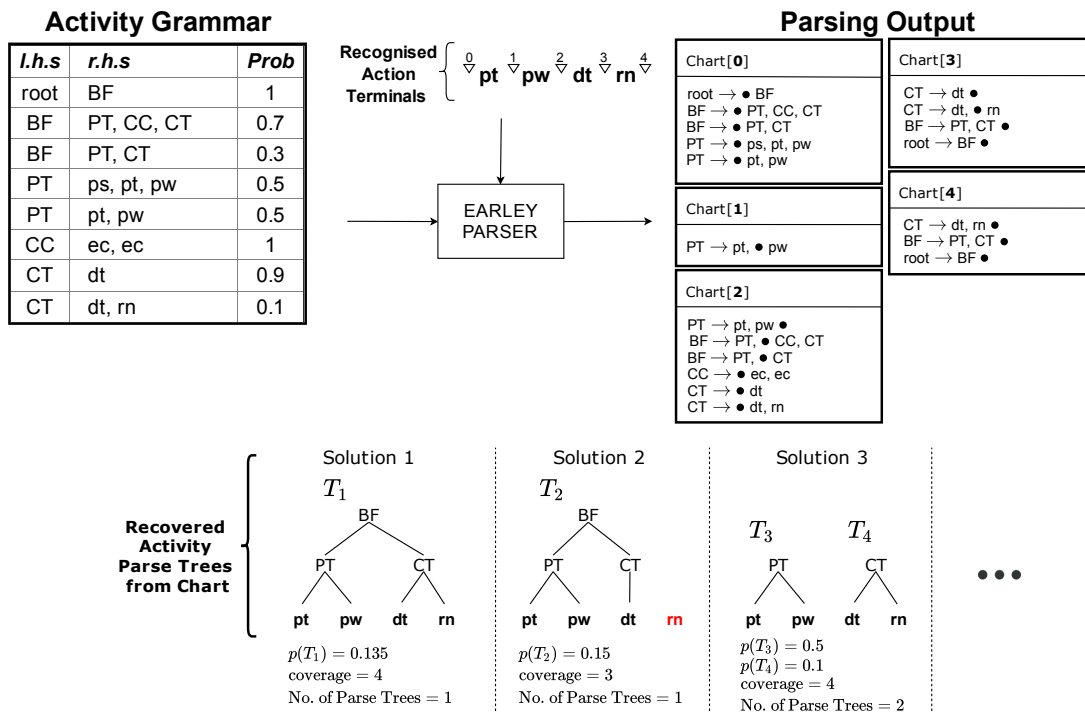


Figure 4.15: A sample learned activity grammar presented along with a sample recognised low-level action terminal stream. Note that this example omits temporally complex activity grammar and input action stream for simplicity. The parsing output shows the output from a standard Earley parser executed on the input stream. From such a parsing output, multiple solutions with partial parse trees describing any sub-set of the input stream can be retrieved. Finally, the probability of the parse trees, the coverage (number of terminals described by the trees), and number of parse trees are shown for each solution.

The set of activity grammar rules learned from abstraction of multiple temporalised hierar-

chies is shown in the activity grammar in the example. We have omitted the relational matrix and each activity node is represented by only one activity label which is consistent throughout the grammar. These simplifications are made so that the basic parsing mechanism using Earley parser can be illustrated. As mentioned before, the parser maintains a chart which records the complete parsing history in terms of the grammar rules explored so far at each position in the input terminals. The chart is shown on the right of the example. Note that the chart comprises of a set of rules partitioned by every position in the input stream (denoted by triangles). The detailed parsing algorithm can be found at [Earley, 1970]. At the end of parsing, note that the chart stored, at each position in the input stream, all grammar rules describing input symbols up to that point. The last chart, chart[4], represents the grammar rules which describe the entire sequence of low-level action. From this chart, we can find rules denoting the start symbol (root) and use back pointers to recover the expansions of the rule used at each of the previous charts to generate the parse tree shown in solution 1. This parse tree is the output from a standard Earley parser.

Since our activity grammar exhibits temporally complex rules, i.e. rules are described by a set of activity nodes in Q and temporal relations matrix which qualitatively defines the temporally complex relations amongst the members of Q in matrix \mathbf{R} . Furthermore, our input stream is a temporally complex set of low-level action terminals. Therefore, the traditional Earley parser requires modification to handle such additional complexity. [Zhang et al., 2011] provides an extended Earley algorithm which relaxes the constraint stating that the input stream must follow a *sequential* order. Briefly, this extended parsing is performed by allowing multiple threads of grammar expansions considering various sub-sets of the input stream comprising of symbols from various positions rather than a single range of input stream defining start and end positions. The resulting chart from the multi-threaded parsing would encompass trees describing many sub-sets of the input stream and not necessarily many ranges alone. Using this approach also allows for automatic handling of insertion errors in the input stream as well as methodical handling of deletion and substitution errors. Note that in figure 4.15, multiple solutions are recovered from the chart, assuming errors exist in the input stream. This is a valid assumption since we use automatic detection of low-level actions to form our input stream which is prone to errors. Solution 2 shows a tree which describes the first three symbols only,

assuming the fourth is an insertion error. Solution 3 shows two trees each of which describing some subset of the input stream. The probability of each tree is presented as the product of the rules used in constructing that tree. The coverage defines the number of input symbols described by the tree and the number of trees is the set of parse trees used within the solution. We propose that to find the best hierarchical description of the activity, we must recover the optimal set of minimum number of parse tree, which maximally describe the input stream with the highest probability. In the next section we describe how this is achieved through formalising an optimisation.

Selecting the Optimum Set of Parse Trees

In NLP, a sentence is successfully parsed if and only if a parse tree is found which agrees with these conditions: 1) the parse tree describes all words of the sentence and 2) the parse tree reaches the root node. These are strict condition in the context of activity grammar. Previous approaches follow these condition when parsing human activities using multi-threaded parsing [Zhang et al., 2011] or regular parsing [Younger, 1967]. We observe that a grammar model of activity learned from human descriptions generated from either expert annotators or crowdsourced annotators, exhibit high amount of inconsistency and noise amongst description of similar activities. Furthermore, describing complex activities in a hierarchical structure by their activity labels rather than a more abstract representation such as trajectories or motion patterns etc. creates highly specific activity rules. Given an unseen test video, an identical set of input stream similar to previously seen instances, needs to be detected to generate a full parse tree. This is unlikely in a human described activity grammar. Fuzzy or loose matching of temporal nodes during parsing alleviates this problem to some degree, however complete parse trees which describe the entire input stream are still unlikely. We therefore propose a mechanism to instead retrieve multiple partial parse trees each of which describe some part of the input stream. Figure 4.16 shows a temporally complex input stream of recognised actions from a test video, these are input to the activity grammar to parse multiple trees, each denoted by T_i , which describe some subset of the low-level action stream.

We propose an optimisation over the complete set of recovered parse trees $\{T_1, \dots, T_{N_\tau}\}$ recorded in the chart from Earley Parser, to select a sub-set of these parse given the following

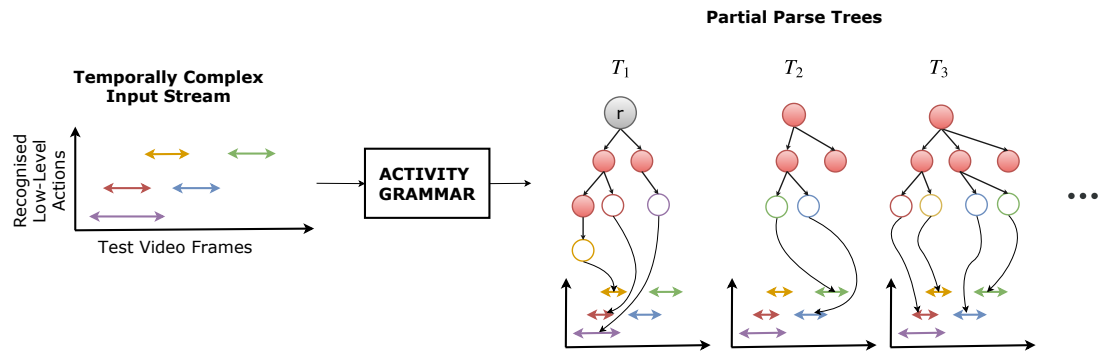


Figure 4.16: This figure shows a temporally complex input stream on the left. Using an activity grammar with the Earley multi-threaded parser, we are able to recover all partial parses over many subsets of the input terminal stream. T_1 describes the orange and purple input actions, T_2 describe the green and blue actions etc. Note that the lowest level nodes, (hollow circles outlined with the same colour as their activity node) correspond to detected actions as seen in the action terminal stream.

goals:-

1. **Maximum Coverage:** The optimisation should select a subset of trees such that maximum number of low-level action terminals are described. In other words, the set of parse trees should maximise coverage over the action input stream. Observe figure 4.16: notice that parse tree T_1 covers three action terminals from the input stream, parse tree T_2 covers two other action terminals, and parse tree T_3 covers four action terminals. An optimum solution here would be the selection of parse trees T_1 and T_2 since they would collectively cover the entire stream of action symbols.
2. **Minimum number of Trees:** Along with maximum coverage, it is also desirable to minimise the number of trees selected. In other words, we aim to find the minimum number of trees that are able to describe the maximum number of action terminals. Figure 4.17a illustrates this: notice the two solutions shown in figure 4.17a on the top and bottom the triangle illustrates coverage of action terminals by a parse trees. It is obvious that the bottom solution is the desirable one since it has a comparatively lower number of parse trees, hence each parse tree exhibits deeper hierarchy.
3. **Highest probability:** Each parse tree is associated with a probability. This is computed using equation 4.8 which is the product of individual probabilities of all rules used in

generating the parse tree. Along with maximum coverage and minimum trees, we aim to find a set of trees with the maximum likelihood. This is an aim to avoid selecting erroneous parse trees which are learned as result of noisy annotations yielding low-probability.

4. **Distinct Parse Trees Coverage:** Finally, the optimisation process should return a set of parse trees each of which distinctly describe some sub-set of the action terminals stream. No two selected parse trees should describe the same action terminal. This requirement is important to avoid confusion in the final description of the top-level activity hierarchy i.e. each low-level activity should have a unique interpretation in the induced activity hierarchy. Figure 4.17b illustrates this goal whereby the bottom solution is valid with distinct trees' coverage while the top solution is infeasible due to overlap between trees' coverage.

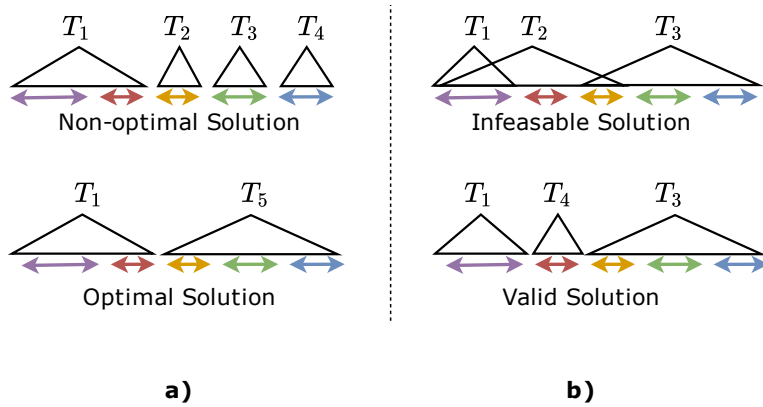


Figure 4.17: This example illustrates the correct selection of partial parse tree over an input action terminal stream. The input stream is shown to be sequential in this example for simplicity, however, the same requirements apply for a temporally complex input stream. The triangles refer to the set of terminals that are described by the parse tree denoted by T . Note that we aim to minimise the number of trees selected therefore the bottom solution is preferred over the top one in a). In b), our aim is to avoid selection of trees which describe the same terminal; therefore the bottom solution is preferred to the top solution here as well.

The goals defined above are all captured in an optimisation where a single objective function captures the first three goals while the set of associated constraints capture the fourth goal. Let us formally describe the optimisation process. Consider an input stream of low-level actions $\mathcal{L} = \{L_1, L_2, \dots, L_{N_L}\}$ and a set of parse trees $\mathcal{T} = \{T_1, T_2, \dots, T_{N_T}\}$, where each tree T_i describes a subset $\hat{L}^i \in \mathcal{L}$ of low-level action terminals. This information is organised into a matrix

$\mathbf{A} \in \{0, 1\}^{N_{\mathcal{T}} \times N_L}$ where each row corresponds to a single parse tree and each columns refer to a low-level action terminal. An element \mathbf{A}_{ij} of the \mathbf{A} matrix is a binary flag which denotes whether the j^{th} low-level action terminal is being described by the i^{th} parse tree. A vector $\mathbf{s} = \sum_j^{N_L} A_{ij}$ denotes the total number of low-level action terminals described by each parse tree. Finally, a vector \mathbf{p} defines the associated probability or likelihood of each parse tree.

The goal is then defined as: find the minimum set of optimum parse trees $\hat{\mathcal{T}}$ which distinctly describe the maximal number of input low-level actions. Using integer linear programming, we can formulate this problem as an objective function and a set of constraints. Equations 4.6.2 represents the objective function and the constraints of the optimisation.

$$\begin{aligned}
 & \underset{x}{\text{maximise}} && \sum_i^{N_{\mathcal{T}}} \frac{\mathbf{s}_i x_i}{N_{\mathcal{T}}} - x_i + \log \mathbf{p}_i x_i \\
 & \text{subject to} && \sum_i^{N_{\mathcal{T}}} \mathbf{A}_{ij} x_i \leq 1 \quad \forall j \in \{1 \dots N_L\}, \\
 & && 0 \leq x_i \leq 1, \quad x_i \in \mathbb{Z}, \quad i = 1, \dots, N_{\mathcal{T}}.
 \end{aligned} \tag{4.9}$$

Our objective function is made of three terms, each of which contribute to one of the goals of the optimisation. The first term $\frac{\mathbf{s}_i x_i}{N_{\mathcal{T}}}$, computes a score which represents the total number of low-level actions described by each parse tree. This value is normalised by the total number of trees to stay in the range of $[0, N_{\mathcal{T}}]$. This normalisation ensures the range is consistent with other terms in the optimisation. Having this consistency is important to avoid uneven ranges causing unwanted bias toward any term in the function. Such biases are usually enforced by introducing weight if needed [Savic, 2002]. The second term x_i is simply the selector term and which defines the number of selected trees. This term works towards selecting the minimum number of trees, hence the subtraction sign, in the optimisation. Note that a different way to formulate the objective function such that minimum number of trees are selected is by dividing the remaining terms by the number of selected trees i.e. divide by x_i . However, this makes division by zero a possibility and to avoid this case we must then further add this constraint: $\sum_{i=1}^m x_i > 0$ ensuring at least one tree is always selected as the solution. This is undesirable behaviour since this formulation adds an additional constraint creating higher complexity and

it forces selection of at-least one parse tree, this enables recognition over any input stream regardless of its correctness. This is why we therefore encode the ‘minimum number of trees selected’ requirement as a negative term added with the remaining terms rather than a division. Finally, the last term $\mathbf{p}_i x_i$, aims at selecting the trees with the highest probability. Probability values are present for each parse tree and are encoded in the vector \mathbf{p} . Collectively these terms simultaneously optimise the three different objectives.

The function alone can easily be maximised if all trees are selected, however, these trees will have redundancies in describing low-level actions as it is likely that multiple trees describe the same input low-level symbol. We therefore need to impose a constraint which ensures that each parse tree describes a unique set of input symbols and no two trees describe the same symbol. This is enforced by ensuring that the sum of the columns of the \mathbf{A} matrix i.e. number of parse trees describing each low-level action terminal, is at most 1. In other words, a maximum of one parse tree is allowed to parse an input symbol. Finally, we impose another constraint that ensures the decision variable x is a binary variable indicating 1 if the parse tree at the i^{th} index is selected as part of the solution and 0 otherwise. This variable is also required to be an integer which is enforced by the $X_i \in \mathbb{Z}$ constraint. This is an implicit condition in designing an integer linear program.

ILP is an NP-complete problem, however, there exist a plethora of efficient solvers to find an approximate solution in a reasonable time. We therefore use such an off-the-shelf ILP solver to solve this program. The result provides us with the **minimum** set of optimal partial parse trees that describe the **maximum** amount of low-level action terminals with the **highest** probability.

4.7 Conclusion

In this section, we presented an approach to model and recognise complex activities. We proposed a method of first clustering annotations received from multiple human annotators in order to remove redundant labels. The resulting set of clusters, called activity nodes, represent an activity at some time interval. We then proposed a cost-based optimisation to automatically infer the *parent-child* relationship between activity nodes in order to generate a temporalised hierarchy of the activity per video. We presented an algorithm to merge multiple hierarchies

and produce a generalised hierarchical model which abstractly captures the latent structure of activities and their sub-activities. The learned model is represented as a probabilistic activity grammar. After extracting the low-level actions from the learned model, we use our simple activity recognition method (QQSTR) to train a discriminative model of low-level activities. Given a temporally complex input stream of low-level actions, all possible activity parse trees are generated using the activity grammar. Finally, we propose a method to optimally select the *best* set of partial parse trees that describe the input low-level action detections. Hence inferring an activity hierarchy over the test video.

We have shown that, complex activities can be efficiently modelled using hierarchical approaches which are scalable with larger activities. We have further presented methods to perform inference over the learned hierarchical models. In the next section, we evaluate the presented approaches using a variety of datasets to demonstrate the effectiveness of our system.

Chapter 5

Experimental Procedure

5.1 Introduction

In this thesis, we have presented two main frameworks for activity recognition. The first framework focuses on activity recognition at a small scale, primarily aimed at modelling and recognising activities that span over few seconds, for example ‘pick up apple’, ‘move box’ etc. We presented a novel feature space that achieves abstract representation of simple activities by employing qualitative and quantitative representation. The second framework naturally progresses onto dealing with activities of higher complexity spanning longer time frame, for example ‘having breakfast’, ‘making a phone call’ etc. In this framework, syntactical structures of complex activities are learned as hierarchies by first acquiring and clustering freely labelled annotations of activities within activity videos from multiple annotators. Then, automatic inference of *parent-child* relationships between various annotated label sets provides an optimum temporalised activity hierarchy. Activity hierarchies from various videos are used to learn a generalised hierarchical model, represented as an extended activity grammar, of the top-level activity. Finally, a parsing technique is used to parse activity parse trees describing an unseen test video given automatically recognised low-level actions. In this chapter, we evaluate each of the presented techniques using various datasets and compare our approach to other techniques or well-defined baselines when applicable. The following experiments are carried out to evaluate

our approach.

1. Recognition of simple activities: In this experiment, we evaluate our qualitative and quantitative spatio-temporal features (QQSTR) designed for learning a discriminative model and recognition for simple activities.
2. Modelling mirrored activities: In this experiment, we evaluate the ability of our feature sets within the QQSTR framework to model and discriminate between mirrored activity such as ‘pushing’ and ‘pulling’.
3. Clustering annotations from multiple annotators: This experiment evaluates the clustering mechanism designed to cluster together semantically similar annotation labels, acquired from multiple annotators for a single video. The experiment further demonstrates the advantage achieved from utilising semantic similarity along with temporal boundary differencing within the overall distance measure designed for clustering.
4. *Parent-child* relationship inference: This experiment evaluates a highly subjective and key aspect of our approach, namely, the inference of *parent-child* relationships between different clustered label sets (activity nodes). We evaluate the use of the devised cost-function that gauges the presence of this relationship among activity nodes based on the temporal overlap along with semantic closeness.
5. Complex activity hierarchy recognition: This experiment evaluates multiple different components of our approach. We collectively evaluate 1) the learned generalised hierarchical activity model as a grammar, 2) the parsing of activity hierarchies over test videos, 3) the use of inexact activity cluster matching and 4) use of qualitative temporal knowledge representation within the activity grammar. The goal of this framework is complex activity recognition, therefore correctly inferred hierarchies over unseen test videos jointly validate these aspects of the overall approach. Furthermore, since we employ the QQSTR system for low-level action recognition, the effects of using this system on the overall task are also evaluated.

This chapter is organised as follows, first we present the evaluation measures employed to evaluate the performance of each component of our system. We then describe in detail the

rich and varied activity datasets used to demonstrate the strength of our approach. Finally we describe, in detail each of the above mentioned experiments along with results and discussions.

5.2 Evaluation Measures and Ground-truth

The following performance measures are used to evaluate different components of our system.

5.2.1 Accuracy

Accuracy is the simplest evaluation metric used in the literature to evaluate machine learning algorithms. Given some test data, accuracy is the ratio of correctly classified instances of the test data over all instances. In order to label an instance of data as correct, we require pre-annotated instances of the same data, also known as ground-truth. We are then able to use the ground-truth to compute an accuracy value by direct comparison of the output of our algorithm with the ground-truth data. Accuracy value is the most fundamental indication of the performance of our approach.

5.2.2 Recall

Though accuracy is a useful metric, on occasion the disparity between the number of total instances and number of relevant instances may be high in classification tasks. This results in misleading accuracy values that do not reflect the true performance. Therefore, we also use the recall measure [Ting, 2010] as described in equation 5.1 to capture the relative accuracy based on the relevant number of instances rather than all instances. TP refers to the number of true positive instances and FN is the number of false negative instances.

$$recall = \frac{TP}{TP + FN} \quad (5.1)$$

5.2.3 Precision

Precision values are related to recall values in that these values reflect the model's ability to retrieve relevant instances [Ting, 2010]. Having a high value for precision is generally thought

to have an inversely proportional effect on recall. Therefore, both measures are reported to gain a fuller understanding of the performance of our approach. Precision is computed using Equation 5.2. FP is the number of false positive instances.

$$precision = \frac{TP}{TP + FP} \quad (5.2)$$

5.2.4 Normalised Mutual Information

Normalised mutual information (NMI) presented by [Cover and Thomas, 2012], is a measure of mutual information between two label sets. Equation 5.3 is used to compute mutual information between two sets of label X and Y . The normalised mutual information shown in equation 5.4, forces an output to be bounded in the range $[0,1]$, where $H(*)$ represents the entropy of any set. An NMI value of 0 indicates no mutual information between sets X and Y therefore highly different label sets, while 1 indicates perfect correlation between sets, i.e. exactly same label sets. We use this metric to evaluate clustering results comparing ground truth cluster assignment labels with automatically generated cluster assignment labels.

$$MI(X, Y) = \sum_{y \in Y} \sum_{x \in X} P(x, y) \log\left(\frac{P(x, y)}{P(x)P(y)}\right) \quad (5.3)$$

$$NMI(X, Y) = \frac{MI(X, Y)}{\sqrt{H(X)H(Y)}} \quad (5.4)$$

5.2.5 V-Measure

Another commonly used metric to evaluate clustering results provided ground-truth cluster labels is the v-measure [Rosenberg and Hirschberg, 2007]. This metric is computed using *homogeneity* and *completeness* metrics. Homogeneity gauges whether the automatically generated clusters contain data points from the same ground truth class, whereas completeness observes data points of ground-truth cluster and check if all these elements belong to the same predicted cluster. The v-measure is then computed using equation 5.5 and its value ranges between 0 and 1 with higher values indicating better clustering performance.

$$\text{v-measure} = 2 \times \frac{\text{homogeneity} \times \text{completeness}}{\text{homogeneity} + \text{completeness}} \quad (5.5)$$

5.2.6 Correctly Parsed Terminals (CPT)

This measure is commonly used to quantitatively estimate the validity of a parse tree in language. We use the same measure to estimate the correctness of our activity parse trees. Given an input stream of terminals, in our case low-level actions, CPT reflects the ratio between correctly parsed terminals and all terminals. Equation 5.6 shows how this measure is computed. NC is the number of correctly parsed terminals, NI is the number of correctly identified insertion errors and finally NT is the number of total terminals in the input stream. We describe further details on correct identification of errors in experiment 5.5.3.

$$CPT = \frac{NC + NI}{NT} \quad (5.6)$$

5.2.7 Qualitative Analysis

Finally, we make use of qualitative analysis to visualise the results in a human-readable form. An example of this is visualising predicted labels of activities overlaid on video segments or observing activity parse trees on test videos and qualitatively estimating its correctness. This is an important mechanism to ensure the credibility of our approach especially given the subjective nature of *parent-child* relationship inference mechanism and *correct* activity hierarchies.

5.3 Datasets

We use three publicly available datasets to demonstrate the effectiveness of our approach. These datasets have been carefully selected in order to ensure a wide range of complexity and variety is embodied. Thereby allowing for a demonstration of our approach under multiple different conditions. The datasets include videos of daily activities being performed by human subjects. Each video comprises of one top-level activity and many sub-activities at various granularities. Activities range from short simple activities such as ‘pick up’ or ‘take a sip’ etc. to long complex

activities such as ‘lunch service in a restaurant’. Each of the datasets are described in detail next.

5.3.1 Cornell Activity Dataset-120 (CAD-120)

This popular dataset [CAD, 2013] was introduced by Cornell University, it consists of activities of daily living tasks. We use this dataset to benchmark our simple activity recognition approach as well as demonstrate our complex activity recognition framework. The dataset has been widely used in the past by various supervised and unsupervised activity recognition systems [Koppula and Saxena, 2013, Rybok et al., 2014], resulting in many benchmarks of activity recognition performance. This allows for easy comparison to existing approaches. The dataset consists of 124 RGBD videos of 10 daily living activities performed repeatedly by 4 human subjects. There is a total of 10 high-level activity classes, namely ‘arranging objects’, ‘cleaning objects’, ‘having a meal’, ‘making cereal’, ‘microwaving food’, ‘picking objects’, ‘stacking objects’, ‘taking food’, ‘taking medicine’ and ‘unstacking objects’. Figure 5.1 shows some sample images from this dataset.

Activity Complexity

This dataset comprises of activity videos which exhibit moderate complexity. The activities are performed by subjects facing a fixed camera with minimal background clutter. Activity classes are acted out in a gradual motion and each class is clearly defined. Activities are performed quite similarly across different instances with the exception of one subject who is left-handed.

However, some challenging aspects of the dataset are that activities presented are of natural daily tasks which lends some complexity compared to other traditional activity datasets where activities are often repetitive and of unnatural motion such as ‘jumping jacks’. Furthermore, human subjects are a combination of right-handed and left-handed subjects presenting a further challenge to employ a handedness-independent representation of activities for attaining high classification performance.



Figure 5.1: Sample images of the CAD dataset. Various activity classes presented including ‘Microwaving Food’, ‘Stacking Objects’ etc. Bottom row shows the automatic skeleton tracks and object detections overlaid over the images. Image taken from [Duckworth, 2017]

Hierarchical Structure

The dataset does exhibit some hierarchical structure since each of the high-level activities comprise of sub-activities. The set of sub-activities labelled in the dataset are ‘pouring’, ‘eating’, ‘opening’, ‘placing’, ‘reaching’, ‘moving’, ‘cleaning’, ‘drinking’ and ‘closing’. However, these activities are temporally sequential i.e. they sub-activities occur one after the other. This makes learning and eventual recognition of hierarchies easier. Also, all mentioned sub-activities reside at the same conceptual level of granularity resulting in two-level hierarchies. For the purpose of demonstrating our complex activity recognition learning and recognition framework, we therefore extend this dataset by introducing further annotations. Additional annotations create deeper hierarchies by introducing labels of intermediate levels of the existing two-level hierarchies. For complete unbiased labelling, independent expert annotators are employed

to label activities freely. This also ensures that the dataset has been annotated by multiple annotators, a requirement for our complex activity recognition framework.

Annotation Complexity

Annotation labels of activities are an important part of a dataset for demonstration of our complex activity recognition approach. These annotations are used to learn the hierarchical structures of high-level activities within videos. Activity labels provided for the CAD dataset are linguistically consistent across different instances of activities, however, they do exhibit some noise in their activity boundaries (start and end frame labelling). Deeper complexity arises when this dataset is extended by employing in-house independent annotators to further annotated the videos and identify more activities at intermediate levels of granularity. This additional labelling creates temporal boundary noise with the existing labelling. In order to isolate linguistic similarity computation, we preserve the label consistency by providing a pre-set list of labels to the annotators. If an annotator identifies an activity whose label isn't present in the pre-set list, the annotator is able to push the new label to the list. In this way, an annotator has the freedom on label selection whilst label consistency is ensured. This dataset is intended for demonstration of our simple activity recognition approach and our core contributions of the hierarchical learning approach, namely learning of the *parent-child* relationships.

Meta-data

Along with annotations of activities, the dataset comprises of 15 joint (x,y,z) automatic OpenNI skeleton tracks of the human subjects as well as automatic and partially ground-truth object tracks. Skeleton tracks are generated using the OpenNI tracker which suffers greatly from occlusion therefore causes noisy jitter. The partial ground-truth object tracks are generated by ground-truth labels of objects at every 50 frames and SIFT-feature [Lowe, 2004] guided interpolation for the intermediate frames. These tracks are reasonably accurate; however, the fully automatic object tracks suffer from mis-detections and high amount of jitter resulting in significant inaccuracies. Furthermore, object tracks also lack object labels i.e. objects are not uniquely identified across videos, however, we demonstrate that our classification system is able to perform well even in the absence of precise label information.

5.3.2 Leeds Activity Dataset (LAD)

This dataset was developed at Leeds University with the aim of presenting longer videos of activities [LAD, 2015]. Previous datasets suffered from activities spanning short time frames (less than one to one minute). Such short activities lacked deep activities hierarchies where the activity can be broken down into many levels of sub-activities. In order to explore hierarchical model’s descriptions of long and complex activities, the LAD dataset was created. The dataset comprises of 15 long videos of ‘having a meal’ activity performed by 5 different subjects repeatedly. This dataset comprises of greater complexity in hierarchical structure since activities are long and exhibit activities at many levels of granularity, therefore making it an ideal dataset for demonstrating our complex activity recognition approach. Examples of activity videos from this dataset are shown in figure 5.2.

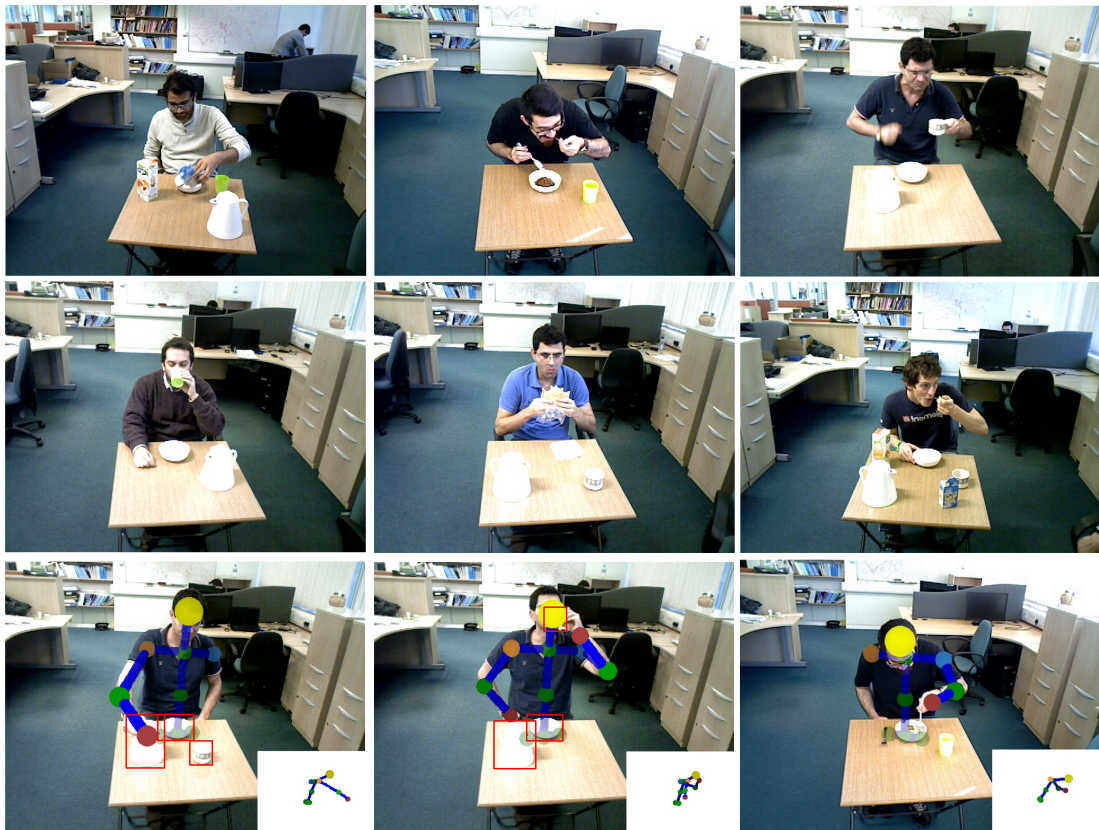


Figure 5.2: Sample images extracted from some videos of the LAD dataset. Bottom row shows automatic skeleton tracks and automatic object detections overlaid over the images.

Activity Complexity

This dataset comprises of activities that exhibit sufficient complexity. Activities are performed facing a fixed RGBD camera with moderate background clutter. Activity classes are acted out at normal speed and each class is clearly defined. Activities vary moderately across different classes as subjects switch hands and/or interact with different sets of objects or perform the same activity differently in terms of order of sub-activities. This introduces complexity when learning discriminative models from highly varying instances of an activity. Activities are also natural occurring daily living action such as such as ‘pickup mug’, ‘pour cereal’ etc. Differently to the CAD dataset, activities may occur in parallel or in a temporally complex arrangement. Also, activities comprise of many objects in the same scene and subjects are able to manipulate any subset of them freely.

Hierarchical Structure

The dataset was specifically constructed to have hierarchical depth within activities. Though there is one top-level activity of ‘having a meal’ within all videos, many sub-level activities exist further down in the hierarchy. These include ‘consuming cereal’, ‘consuming sandwich’, ‘preparing tea’ etc. Further smaller sub-activities comprise of ‘pickup mug’, ‘add milk’, ‘pour cereal’ etc. The dataset provides hierarchies of up to 4 levels of depth. As mentioned before, sub-activities often exhibit temporally complex ordering such as overlapping or parallel activities. This is a particularly desirable feature as it allows us to test our representation of activities using Allen’s temporal relations within the activity grammar \mathbb{G} .

Annotation Complexity

Activity annotations were performed by 4-5 different annotators. In the first round, annotators were free to annotate activities at various levels of granularity. The annotations, since identified by multiple annotators, exhibit temporal boundary noise and linguistic variability. However, to avoid linguistic complexity, annotations were standardised by manual merging of similar annotations into single labels for consistency across all instances of an identified activity. This dataset is therefore ideal to estimate performance of our core *parent-child* relationship learning

and inference mechanisms; in particular, given the larger depth of activities than the CAD dataset, it provides greater complexity for correct hierarchical learning.

Meta-data

The dataset comprises of annotations of activities at different levels of granularity. 15 joint (x,y) automatically generated skeleton tracks are provided. These skeleton tracks are better than OpenNI tracking used in CAD dataset however some noise and jitter is still present. Object tracking is performed by a table-top object tracker, however, it is only able to identify large objects. The lack of all object tracks presents a grave challenge in the simple activity recognition task since interactions are not fully captured without complete object tracking.

5.3.3 Complex and Long Activity Dataset (CLAD)

This dataset was constructed to further introduce longer activities with deeper hierarchies [Tayyub et al., 2017]. The LAD dataset, though exhibited decent sized hierarchical structures within the activities, had only one top-level activity class of ‘having a meal’ across the entire dataset. The CLAD dataset was collected at Leeds University with the aim of capturing long videos of multiple top-level activities comprising of numerous sub-activities and spanning over longer times. The motivation stems from the need for effective activity recognition methods for long-term autonomous operation of an artificially intelligent system such as a robot [Hawes et al., 2017] or a self-driving car. The dataset consists of 62 videos capturing activity classes at various granularities in the following three scenarios: ‘having breakfast’, ‘lunch in a restaurant’ and ‘working in an office’. These are the 3 top-level activity classes in the dataset with 20 videos each for ‘having breakfast’ and ‘working in an office’ scenario and 22 videos for ‘lunch in a restaurant’ scenario. Activities are performed by 5 different subjects. The higher complexity, longer length and more videos of activities in this dataset makes it highly desirable to stress test our complex activity recognition system. Figure 5.3 shows sample images of activities from the CLAD dataset.



Figure 5.3: Sample images from videos of the three main classes of activities: ‘having breakfast’, ‘lunch in a restaurant’ and ‘working in an office’ presented. Each column shows activities from each one of the top-level activities. Bottom row shows automatic skeleton tracks overlaid on the sample images.

Activity Complexity

Activities in this dataset were recorded using a fixed RGBD camera which replicated the view point of a mobile robot (e.g. PR2, Tiago, Scitovs A5). When collecting videos for this dataset, actors had complete freedom to perform the top-level activity as they perceived, utilising any sub-set of the available objects. This unscripted method resulted in highly complex activity videos exhibiting high amount of variation between instances of the same top-level activity. Activities were therefore performed in the most natural way possible at a normal speed. Moreover, some activities also exhibit multiple subjects which introduces inter-human interaction resulting in a more natural and complex scenario.

Hierarchical Structure

The dataset consists of long and complex activities which naturally exhibit deep hierarchical structure. Each top-level activity has activity description of up to 4 levels of hierarchy with sub-activities varying across granularity. Example sub-activities range from long activities such as ‘taking an order’, ‘making a payment’ etc., to intermediate level actions such as ‘stapling sheets’, ‘retrieving credit card’ etc. to lowest-level *terminal* actions such as ‘pick pencil’, ‘pour milk’ etc. These sub-activities do not strictly exhibit sequential order and are often performed in parallel causing temporally complex ordering especially in cases where two subjects are interacting for example ‘lunch in a restaurant’ scenario. We find that this dataset exhibits videos of activities with more complex hierarchical structure than the CAD and LAD datasets therefore making it highly suitable to test our methods fully.

Annotation Complexity

Traditionally annotation of activities within the video has been performed manually, usually by expert annotators who held some domain knowledge. The task becomes largely expensive, time consuming and may result in biased annotations [Nguyen-Dinh et al., 2013, Roggen et al., 2010]. In order to reduce cost and time to annotate activities, crowdsourcing platforms are increasingly gaining popularity. A crowdsourcing platform offers a large pool of world-wide workers that are able to perform a human intelligent task, such as annotation of a video, for a small financial incentive. For large datasets and long videos, as is the case for CLAD dataset, activities are annotated by employing multiple crowd-sourced workers for annotating each activity within each video.

Since multiple annotators are employed to annotate each single video, this ensures that a rich and varied perspective of the latent activity is reflected in the annotations. A single person usually tends to annotate videos sequentially one activity after the other in a flat temporal sequence. With multiple annotators, annotations at multiple level of temporal granularity are obtained. Combining these annotations from different workers provides a richer annotation of the video at multiple levels of granularity.

Though annotations cover a broad spectrum of latent activities at varying temporal granu-

larity, allowing annotators to freely identify and use language labels to label activities creates highly complex annotations. Annotators may not use linguistically similar labels to identify the same activity, and invariably slight variations occurs. Unlike, CAD and LAD datasets, exact matching of labels therefore becomes complex. Furthermore, annotators re-identify common and obvious activities which creates redundancies in the annotations. Finally, annotators comprise of a mixtures of native and non-native English speakers and come from varying backgrounds possessing little to no research knowledge. This results in labels that have grammar and spelling errors to refer to non-related activities, for example ‘scratching head’, ‘staring into the camera’ etc. or identify noise activity labels. All these issues make the annotations of this dataset highly complex and therefore an attractive and challenging dataset to evaluate our methodology on.

Meta-data

The dataset was provided with 15 joint (x,y,z) skeleton tracking of human subjects in the video from two different state-of-the-art trackers [Wei et al., 2016]. Skeleton and object tracks are critical for the success of our simple low-level activity recognition which in turn provides input for complex activity recognition. Poor performance of a low-level activity detector would have detrimental effect on higher level activity inference. Since the work presented in this thesis does not relate to object tracking, we have not explored to perform object tracking on this dataset and rely on skeleton tracks alone.

The dataset is also provided with point clouds, depth images and finally crowd sourced annotations. Each video is annotated by an average of four to five annotators and each annotation instance is of the form: activity label, start time and end time.

5.3.4 Summary

In summary, three datasets are utilised, Cornell Activity Dataset (CAD), Leeds Activity Dataset (LAD) and Complex and Long Activities Dataset (CLAD). Each dataset offers increasingly complex scenarios which help in demonstrating certain aspects of our framework. Our overall activity recognition approach tackles recognition of simple activities, these are usually lower

level activities spanning short time frames, and complex activities which are activities that span longer time and are usually more complex and comprise of sub-activities in nature.

Table 5.1 shows an overview of each dataset and information on the properties of each dataset. Note that CLAD dataset offers the highest complexity with longest videos, multiple human subjects, crowdsourced annotations, deep hierarchies and noisy labels in temporal boundary and linguistic variation.

Property	LAD	CAD	CLAD
Number of Videos	15	120	62
Length of Videos (Minutes)	0.5-1	0.3-0.5	3-5
Multiple Subjects in an Activity	×	×	✓
Crowdsourced Annotations	×	×	✓
Number of Activity Classes	10	26	≈ 50
Depth Of Hierarchy (Levels)	3	2-3	up to 5
Labels Noisy Temporal Boundries	✓	✓	✓
Linguistic Variation in Labels	✓	×	✓

Table 5.1: Overview of the properties of every dataset used during experimentation.

5.4 Evaluation of Qualitative and Quantitative Representation

In this section, we evaluate our approach for recognition of activities using qualitative and quantitative spatio-temporal representation (QQSTR) as presented in Chapter 3. This approach is aimed at modelling and recognition of simple activities which span short periods of time (0 - 2 minutes). Figure 5.4 presents the overall framework for the QQSTR activity recognition system. Note that all sections of the framework are tested on all the datasets.

In this system, we use the pre-processed skeleton tracks of human subjects and/or object tracks within the activity video to extract two types of features, qualitative features and quantitative features. Each of these features are aimed at capturing some aspects of interactions that occurs between objects and people during an activity that abstractly represents that activity.

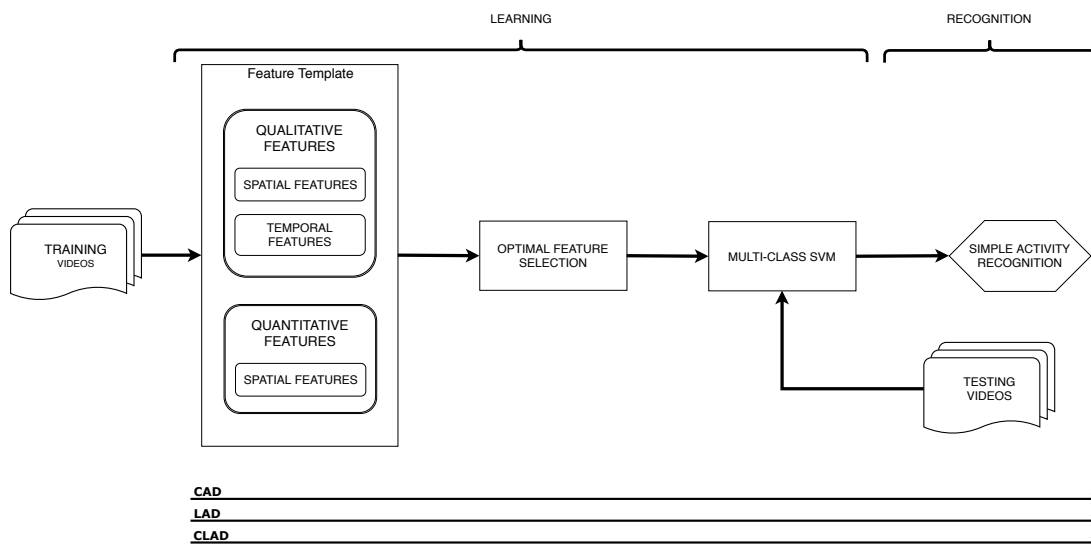


Figure 5.4: Framework of our QQSTR system for simple activity recognition. All three datasets are used to test the full system end-to-end.

These features are then placed through a feature selection process to filter out non-discriminative and redundant features. Then, a multi-class support vector machine (SVM) is trained to learn the various classes of activities within datasets. Finally given unseen test videos, the trained SVM is used to classify activities occurring within the test videos. In this experiment section, we evaluate the overall approach and report recognition results from different datasets. We also discuss the strength and contribution of different types of features engineered and draw interesting conclusions on each set. Finally, we validate the use of specific design choices, such as the use of compression techniques, made within the framework to discern mirrored activities.

5.4.1 Experiment 1: Evaluation of Simple Activity Recognition

In this experiment, we evaluate the overall framework of simple activity recognition as shown in figure 5.4. The evaluation of this system is performed by splitting the datasets into training and testing folds. The training set is used to 1) compute qualitative spatial relationships, 2) estimate best parameters and 3) extract features to learn a discriminative model of activity classes through training an SVM model. A testing set is then used to evaluate the learned model by classifying each instance and comparing the predicted activity classes to the ground truth classes. Accuracy, precision and recall values are reported as the quantitative analyses.

Confusion matrices are also provided for a further overview of the achieved empirical performance. We use the previously described datasets to evaluate the system. Results obtained on each of these datasets are described next.

Cornell Activity Dataset-120 (CAD-120)

The CAD-120 dataset comprises of 120 videos. Recall that the available ground-truth comprises of a single label of the overall activity occurring within each video, as well as labels of sub-activities (start frame, end frames and activity label) present within each overall activity are provided. We evaluate the two level of activities separately. There are 4 human subjects performing the activities and the dataset is equally divided amongst the 4 human subjects. Note that one of the subjects is left-handed. We compare our approach to multiple state-of-the-art results previously established on this dataset. Moreover, similar to previous state-of-the-art approaches, we perform a 4-fold cross validation, whereby training is performed on 3 subjects and testing of the learned model is done on the fourth subject. The values of accuracy, precision and recall are averages of results obtained across all folds during cross-validation. The variation in these values across different folds is also reported as an accompanying error value. Tables 5.2 and 5.3 present the results obtained on the higher-level activities and sub-activities through employing our simple activity recognition framework as compared to other state-of-the-art approaches. Recall that the dataset natively comprises of annotations of activities at two levels of granularity therefore we evaluate our system on these two levels separately.

In table 5.2, results are partitioned by 1) use of temporal segmentation of sub-activities and 2) use of ground-truth or automatic object tracks. The use of temporal segmentation of sub-activities refers to a system which relies on ground-truth labelling of comprising activities in order to aid in the recognition of the higher-level activity. For example, for the recognition of an activity such as ‘microwaving food’, competing systems require ground-truth knowledge of component activities such as ‘opening microwave door’, ‘placing food in’ etc. along with their temporal boundaries. Our system does not rely on ground-truth temporal segmentation of comprising activities and therefore does not appear in that partition. The other partitions in the table are according to the type of object-tracks used. These tracks are either partially ground-truth or automatically generated. Automatic object tracking often produces object tracks which

exhibit jitter and noise which in-turn propagates into features extraction resulting in lower overall performance. In our QQSTR system, object tracks are used to compute qualitative relations amongst entities, noise at this level produces spurious relations that do not reflect authentic interactions due to activities. We make use of a smoothing median filter across the generated qualitative relations as explained in section 3.2.6. The optimal window size is set as 7 frames, relations which change drastically within this window are smoothed out resulting in less noise. We report results of our system, and related approaches, using both types of object tracks to further demonstrate the effect of noisy object tracking on our classification system.

Method	Accuracy %	Precision %	Recall %
<i>assuming ground-truth temporal segmentation</i>			
[Koppula et al., 2013]	84.7 ± 2.4	85.3 ± 2.0	84.2 ± 2.5
[Koppula and Saxena, 2013]	93.5 ± 3.0	95.0 ± 2.3	93.3 ± 3.1
<i>assuming no ground-truth temporal segmentation</i>			
Baseline (Random)	10.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0
[Koppula et al., 2013]	80.6 ± 1.1	81.8 ± 2.2	80.0 ± 1.2
[Koppula and Saxena, 2013]	83.1 ± 3.0	87.0 ± 3.6	82.7 ± 3.1
QQSTR-gt-tracks	95.2 ± 2.0	95.2 ± 1.6	95.0 ± 1.8
<i>assuming no ground-truth temporal segmentation and no ground-truth object bounding boxes</i>			
Baseline (Random)	10.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0
[Koppula et al., 2013]	75.0 ± 4.5	75.8 ± 4.4	74.2 ± 4.6
[Rybok et al., 2014]	78.2	-	-
QQSTR-auto-tracks	75.8 ± 6.8	77.9 ± 11.0	75.4 ± 9.1

Table 5.2: Performance measurements with and without assumption of ground-truth temporal segmentation based on accuracy, precision and recall of the **higher-level activities** in the CAD dataset.

In table 5.2, we show that our approach outperforms previous benchmarking state-of-the-art approaches by a significant margin. Observe that we achieve a 95.2% performance with a precision of 95.2% and a recall of 95.0%. This is a significant improvement of 12.1%, 8.2% and 15.0% in terms of accuracy, precision and recall respectively over [Koppula and Saxena, 2013] in the case where ground-truth temporal segmentation of component activities is not used by

the competing system. We further demonstrate the superiority of our system, presenting an improvement of 1.7%, 1.2% and 1.7%, over the competing systems [Koppula and Saxena, 2013] even when ground-truth temporal segmentation of component activities is available to the other systems. These results show significant superiority to benchmarking methods.

Results achieved using ground-truth object tracks conceptually demonstrates the upper-bound of our approach since it utilises near-perfect object tracking. Object tracking is not a goal of the work presented in this thesis and therefore we utilise partially ground truth tracks to demonstrate the main contributions of classification capability of our QQSTR system. We also use automatically generated tracks to replicate a more real-world scenario and still achieve a decent performance of 75.8% accuracy, 77.9% precision and 75.4% recall. This is a marginal improvement of 0.8%, 2.1% and 1.2% in accuracy, precision and recall respectively over [Koppula et al., 2013]. However, when compared to [Rybok et al., 2014], we notice a slight decline of 2.4% in accuracy. [Rybok et al., 2014] do not use object tracks since their approach is completely based on saliency and optical features generated from raw pixel data, therefore their system is more immune to noise generated through object or skeleton tracking, and hence the higher accuracy.

Method	Accuracy %	Precision %	Recall %
<i>assuming ground-truth temporal segmentation</i>			
Baseline (Random)	11.1 ± 0.0	11.1 ± 0.0	11.1 ± 0.0
Baseline (Majority Class)	34.6 ± 0.0	3.8 ± 0.0	11.1 ± 0.0
[Hu et al., 2014]	87.0 ± 1.9	89.2 ± 4.6	83.1 ± 2.4
[Koppula and Saxena, 2013]	89.3 ± 0.9	87.9 ± 1.8	84.9 ± 1.5
QQSTR-gt-tracks	82.2 ± 2.0	73.4 ± 3.3	80.0 ± 0.3
QQSTR-auto-tracks	77.3 ± 2.8	70.9 ± 4.0	70.7 ± 3.1

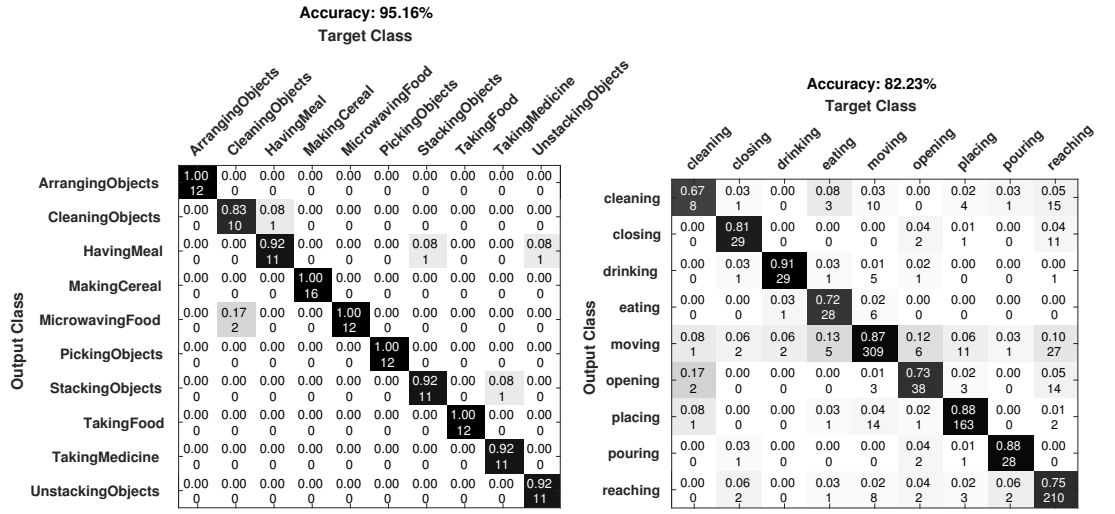
Table 5.3: Performance measurements with and without assumption of ground-truth temporal segmentation based on accuracy, precision and recall of the **sub-level activities** in the CAD dataset.

The results achieved from modelling and recognition of sub-level activities are shown in table 5.3. Sub-level classes present high amount of imbalance between different classes, there-

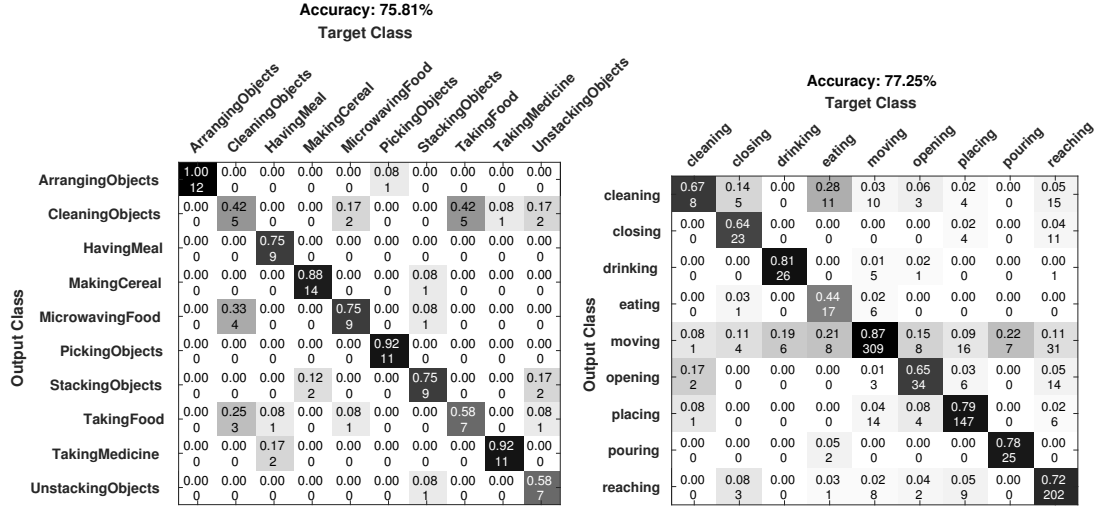
fore, we report two baselines in this table. The random baseline assumes equal probability of classification across all classes in the dataset. This baseline does not take into account the relative number of instances within each class. Therefore, the majority class baseline is reported alongside; this baseline represents performance achieved if all instances are classified into the majority class. Our system is shown to perform significantly better than the two reported baselines when using both the automatic object tracks and the ground truth object tracks. In case of classification of data with high inter-class imbalance, high precision and recall values are complimentary to high accuracy values to indicate good performance. This is evident since our system achieves 73.4% precision and 80.0% recall with a 82.2% accuracy in case of the ground truth tracks. Similarly, high values in case of the noisy automatic tracks are reported. Competing systems report higher performance compared to our approach in the sub-level activity recognition however we demonstrate an overall higher performance in high-level activity recognition.

To obtain a deeper understanding of the performance, we present confusion matrices for the higher-level and sub-level activities obtained from our system using automatic tracks and ground truth tracks in figure 5.5. Using ground-truth object tracking, we notice a strong diagonal in figure 5.5 a) indicating high performance and reputable performance using automatic tracking as seen in figure 5.5 c). This represents accurate recognition of the higher-level activities in the CAD dataset. In figures 5.5 b) and d), the confusion matrices show reputable performance achieved on sub-level activities as well using ground-truth and automatic tracking object tracking respectively. It can be seen that there is a lower decline in performance from using ground truth object tracks versus automatic object tracks in low-level activity recognition, figure 5.5 b) and d), than in high-level activity recognition, figure 5.5 a), c). This is expected behaviour since lower-level activities span shorter time and thereby exhibit lesser noise from jitter and object tracking as compared to higher-level activities. Also note that the highest amount of confusion occurs in the ‘moving’ activity as seen in figures 5.5 b) and d). The ‘moving’ activity comprises of the highest number of instances, and exhibits high visual similarity to other most activities such as ‘cleaning’, ‘pouring’ etc. Despite this high amount of visual similarity, 87% of the ‘moving’ activity instance were correctly classified when using ground truth object tracks.

These results demonstrate that our approach effectively abstracts the interactions between



(a) High-level activity classification using ground-truth object detections. (b) Sub-level activity classification using ground-truth object detections.



(c) High-level activity classification using automatically generated object detections. (d) Sub-level activity classification using automatically generated object detections.

Figure 5.5: Confusion Matrices showing classification results from high-level activities and sub-level activities using ground-truth object detections and automatic object detections, in the CAD dataset.

human subjects and objects within an activity. The system achieves state-of-the-art performance without using object labels or object affordance knowledge in the scene, which are required by competing approaches. We claim that our QQSTR system is designed to model simple activities. Using this dataset, we show a high performance achieved on both high-level

and sub-level activities. Note that the *overall* activities in each video in the dataset, though labelled as high-level, are relatively simple and short activities with limited number of human subjects and objects as compared to other datasets that we have used. Therefore, state-of-the-art performance presented on this dataset does in-fact indicate the strength of the QQSRT system on simple activities.

Leeds Activity Dataset (LAD)

The LAD dataset comprises of videos of longer and more complex activities than the CAD dataset. This dataset comprises of 15 videos where ground-truth labels are provided for activities at three levels of granularity. Activities are performed by 5 different subjects. Similar to the evaluation mechanism on the CAD dataset, we perform cross-validation by leaving one subject out for testing and training on the rest. However, since there exist no other approaches that have evaluated on this dataset, we make use of typical random and majority-class baselines as well as a non-trivial RCC-3 baseline to evaluate our system using this dataset. The RCC-3 baseline is the simplest form of our activity recognition system where only spatial topology is encoded using just the region connection calculus (RCC-3). Remaining processes such as feature extraction and subsequent classification is performed similar to our QQSTR system. This provides a simple yet non-trivial baseline to compare our results against. We use the accuracy, precision and recall values to compare our QQSTR approach with the baselines. Recall that both, skeleton tracking and object tracking are automatically performed on this dataset therefore no results are reported on using ground-truth object tracks which indicate upper-bounds of the system.

Each video in the dataset comprises of one top-level activity of ‘having a meal’. Since there is only one class at the highest-level in each video, we perform activity modelling and recognition on high-level activities using the activities present at the immediate next-level of granularity. Activities at this level comprise of many classes of child activities of the ‘having breakfast’ scenario. Another evaluation is then performed on the lowest-level of activities in the ‘having breakfast’ scenario. Note that, in this dataset, the low-level activity classes are labelled by object types for example ‘pick bowl’, ‘pick spoon’, ‘pour milk’, ‘pour water’ etc. Training a model of activities with this level of specificity will result in highly similar classes

each having only few training instances to learn from. Since our model does not rely on object identities, we merge classes of different activities based on the suffix activity label and ignoring the specific object used. This resulting in many more instances of sub-level activities in merged classes labelled as ‘pick’, ‘put’, ‘drink’ etc. Tables 5.4 and 5.5 present the results obtained from classification of activities using subject based cross-validation.

Method	Accuracy %	Precision %	Recall %
<i>assuming ground-truth temporal segmentation</i>			
Baseline (Random)	14.3 ± 0.0	14.3 ± 0.0	14.3 ± 0.0
Baseline (Majority Class)	31.0 ± 0.0	4.4 ± 0.0	14.3 ± 0.0
Baseline (RCC-3 Only)	41.5 ± 8.3	42.8 ± 7.3	41.0 ± 2.2
QQSTR	62.2 ± 2.0	54.4 ± 3.3	58.4 ± 0.3

Table 5.4: Performance measurements based on accuracy, precision and recall of the **high-level activities** in the LAD dataset.

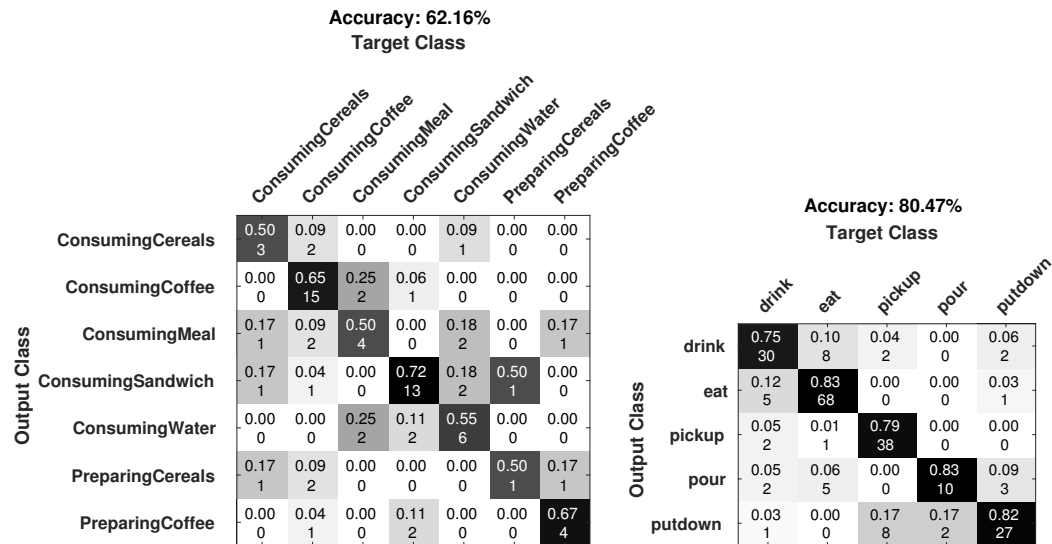
In table 5.4 we present performance results obtained from classification of high-level activities. Complementary to these results, the confusion matrix presented in figure 5.6 a), shows exact classification inaccuracies. Note that we achieve a modest performance of 62.2% with a precision of 54.4% and recall of 58.4%. These values show significant improvement over the baselines, however, as mentioned earlier, the activities modelled here are longer and more complex than the previous dataset. This fact, along with fully automated tracking, explains the modest performance in classifying high-level activities. We show in subsequent experiments that much higher performance is achievable on high-level activity classification using our complex activity modelling method presented in chapter 4.

In table 5.5, we present results obtained from classifying the sub-level activities or simple activities in the LAD dataset. These results validate the strength of our QQSTR system to effectively model simple activities. An accuracy of 80.5% along with a precision of 75.4% and a recall value of 80.5% is achieved, this is a large improvement over the baseline cases. The confusion matrix 5.6 b) presents a strong diagonal indicating a low amount of confusion amongst sub-level activity classes in the LAD dataset. Note that modelling sub-level activities achieves significantly higher performance than higher-level activities using our QQSTR approach as

Method	Accuracy %	Precision %	Recall %
<i>assuming ground-truth temporal segmentation</i>			
Baseline (Random)	38.1 ± 0.0	38.1 ± 0.0	38.1 ± 0.0
Baseline (Majority Class)	38.1 ± 0.0	7.6 ± 0.0	20.0 ± 0.0
Baseline (RCC-3 Only)	32.4 ± 9.6	34.1 ± 11.0	30.8 ± 8.2
QQSTR	80.5 ± 1.5	75.4 ± 6.4	80.5 ± 3.3

Table 5.5: Performance measurements based on accuracy, precision and recall of the **sub-level activities** in the LAD dataset.

shown from evaluation on the LAD dataset. This is expected behaviour since our QQSTR approach is designed to model low-level activities. However, in the CAD dataset, a converse behaviour was evident where higher-level activities performed better than sub-level activities using QQSTR. This is because even the higher-level activity classes within the CAD dataset are inherently of a simplistic nature and therefore our QQSTR approach performs well on modelling these activities as well as the sub-level activities. Further analysis of these results is provided in the upcoming discussions section 5.4.3.



(a) Confusion matrix presenting classification performance on the high-level activities. (b) Confusion matrix presenting classification performance on the sub-level activities.

Figure 5.6: Confusion matrices presenting classification performance of activities in the LAD dataset.

Complex and Long Activity Dataset (CLAD)

The CLAD dataset comprises of the longest and most complex videos compared to the CAD and LAD dataset. This dataset is used with the motivation to evaluate the hierarchical learning approach for complex activity recognition, as presented in further chapters. However, we use the QQSTR approach to model and recognise low-level activities present in the dataset. Recall from section 4.6.1 in chapter 4, that low-level activity classes are automatically emergent through the hierarchy learning process in the complex activity framework. To evaluate the QQSTR approach on this dataset at this stage, we manually extract the low-level activity classes from the CLAD dataset and present results of the simple activity recognition approach on these classes. Note that, activities in this dataset have been annotated freely by annotators through crowdsourcing. This results in inconsistent labels across different instances of the same activity. We resolve this problem to some extent through utilising our semantic clustering mechanism and activity node matching method, as explain in chapter 4. However, these mechanisms do not perform with complete accuracy and therefore could not be used to extract the low-level activity classes. Therefore, in order to train and learn activity models for simple activities, we use the ground-truth activity clusters available for each video and select relatively short length activity clusters as a sample of low-level activity classes. This ensure that activity classes selected are not influenced by automated clustering noise. We extract a total of 18 activity classes from 62 videos of the dataset through this process. Differently from previous datasets, we perform a leave-one-out cross-validation using this dataset. Accuracy, recall and precision values are presented, and results are compared against random and majority class baselines. Table 5.6 show the classification results obtained.

From table 5.6, we notice a good performance of 68.3% accuracy, 70% precision and 71.5% recall achieved. This is a high improvement over the baselines. However, the comparatively lower performance of this system against the other datasets is attributed to two causes. Firstly, this dataset relies on pure skeleton tracking and therefore interactions with objects are not captured to the fullest. One future work aspiration is to obtain object tracks for this dataset in order to test the full capability of our system. Secondly, since object classes are crowd-sourced, the disagreement between labels creates some inter-class dependencies as instances are

Method	Accuracy %	Precision %	Recall %
<i>assuming ground-truth temporal segmentation</i>			
Baseline (Random)	23.2 ± 0.0	23.2 ± 0.0	23.2 ± 0.0
Baseline (Majority Class)	15.9 ± 0.0	8.8 ± 0.0	5.5 ± 0.0
Baseline (RCC-3 Only)	34.3 ± 13.7	27.6 ± 7.1	31.2 ± 6.8
QQSTR	68.3 ± 3.4	70.0 ± 1.4	71.5 ± 2.4

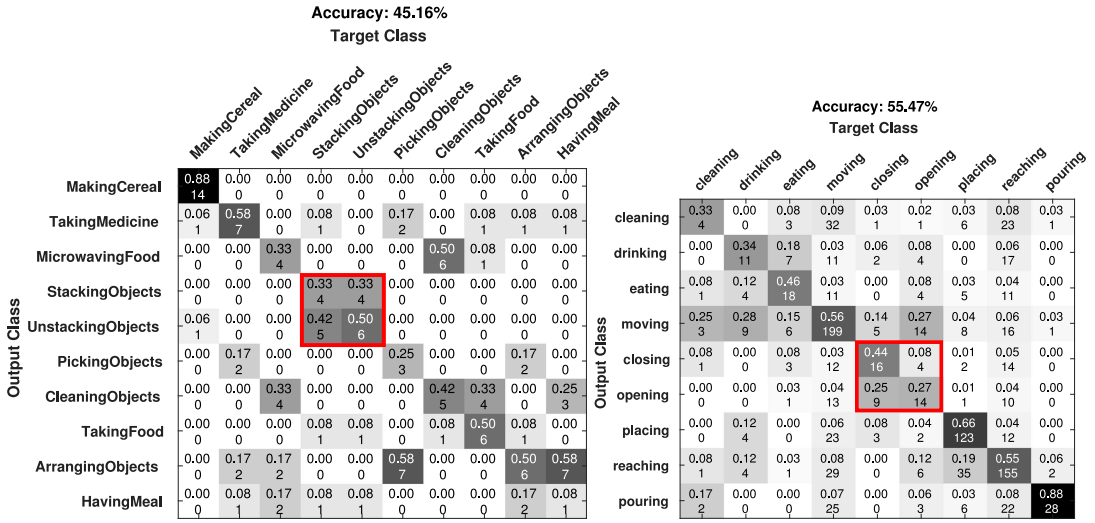
Table 5.6: Performance measurements based on accuracy, precision and recall of the sampled **sub-level activities** in the CLAD dataset.

shared across each class. In other words, there is unclear class separation and therefore unique representations of classes are not easily learned’ resulting in lower recognition performance. Each class of activity describes a primitive action. Automatic recognition of these primitive actions would allow us to infer higher-level actions. Note that the action classes used here are sampled from ground-truth labels of low-level action classes. In later experiments, we show that we are able to learn these classes through the complex activity modelling process.

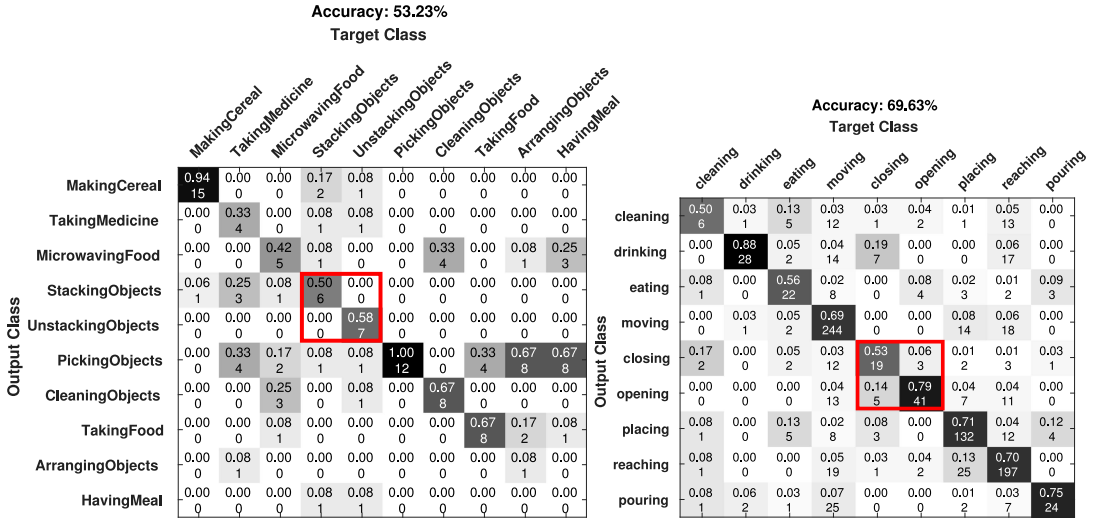
5.4.2 Experiment 2: Evaluation of Classification of Mirror Activities

In our QQSTR approach describe in chapter 3, we mention a challenge of representing mirror activities. Mirror activities are activities where one activity visually exhibits similar elements of another activity in reverse. For example, ‘pushing’ and ‘pulling’, ‘picking’ and ‘putting’ etc. are both mirror activities. We specifically designed representation aimed at discriminately representing such activities. Recall that we utilised a global compression mechanism over a local compression when encoding RCC-3 relational strings, see chapter 3, section 3.2.3. This is done to capture a holistic and complete picture of the activity. Secondly, we include simple qualitative trajectory calculus (SQTC) relations, see chapter 3, section 3.2.2, to represent an overall direction of motion within the activity. Utilising these two mechanisms, we evaluate their collective discriminative power to discern mirror activities in this experiment. We present our results in the form of confusion matrices resulting from classification of activities in the LAD and CAD datasets with and without the use of SQTC representation and global compression encoding. Figure 5.7 and 5.8 shows the set of confusion matrices which demonstrate the

effectiveness of the mentioned approaches on the two datasets.



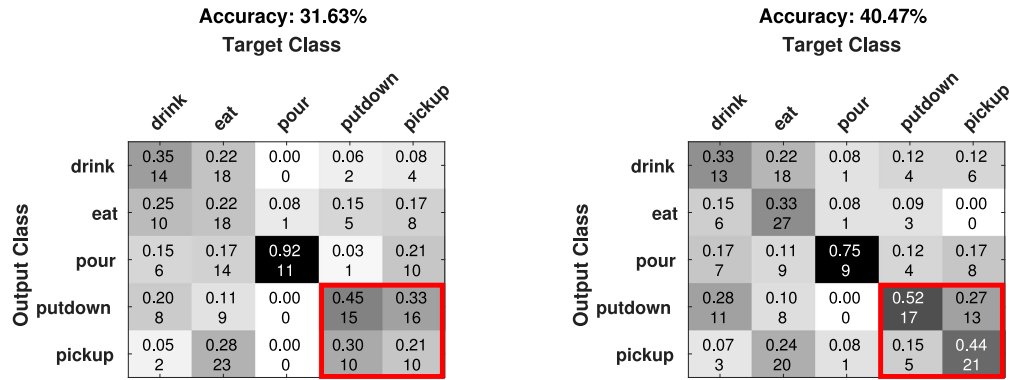
(a) Confusion matrix of high-level activity classification which shows high confusion between the ‘stacking objects’ and ‘unstacking objects’ activities, highlighted by the red rectangle. (b) Confusion matrix of sub-level activity classification which shows high confusion between the ‘closing’ and ‘opening’ activities, highlighted by the red rectangle.



(c) Confusion matrix of high-level activity classification which shows completely resolved confusion between the ‘stacking objects’ and ‘unstacking objects’ activities, highlighted by the red rectangle. (d) Confusion matrix of sub-level activity classification which shows some resolved confusion between the ‘closing’ and ‘opening’ activities, highlighted by the red rectangle.

Figure 5.7: Confusion matrices presenting classification performance of discriminating mirror activities in the CAD dataset. The red rectangle highlights the confusion and shows improvement across columns.

Since the QQSTR system comprises of multiple feature spaces each designed to model some



(a) Confusion matrix of sub-level activity classification which shows high amount of confusion between the 'putdown' and 'pickup' activities, highlighted by the red rectangle. (b) Confusion matrix of sub-level activity classification which shows mostly resolved confusion between the 'putdown' and 'pickup' activities, highlighted by the red rectangle.

Figure 5.8: Confusion matrices presenting classification performance of discriminating mirror activities in the LAD dataset. The red rectangle highlights the confusion and shows improvement across the rows.

aspect of activities, in this experiment, we only use the basic RCC-3 feature space. This is done to isolate the representation so that the effects of global compression encoding method and inclusion of the SQTC representation can be magnified in the results by evaluating the system with and without these features included. Figures 5.7 and 5.8 show confusion matrices resulting from classification of activities from the two datasets. The generally low accuracy value seen in these matrices is due to the use of a limited subset of features as mentioned before. Figure 5.7 presents confusion matrices of high and low-level activities from the CAD dataset. The red rectangle highlights the high confusion between mirror activity classes. In figure 5.7 a) this confusion is evident in higher-level activities between the 'stacking' and 'unstacking' activities, while in figure 5.7 b) some amount of confusion also appears in sub-level mirror activities namely 'closing' and 'opening'. Figure 5.7 c) and d) show confusions matrices obtained when the model was augmented with SQTC representation and global compression was used rather than local compression to encode the qualitative relationships. Figure 5.7 c) shows a complete elimination of confusion between the higher level mirror activities whilst figure 5.7 d) shows an improvement in discerning sub-level mirror activities as well. Similar behaviour is also noticed in the LAD dataset. Figure 5.8 a) and b) show sub-level activities in the LAD dataset with and without the SQTC and global compression encoding method. There is evidence of improvement

with the inclusion of the additional features as seen by the red rectangle in figure 5.8 b). ‘Put down’ and ‘Pick up’ activities are mirror activities which are hard to discern based on counts of spatial relations alone, however, with the use of our augmented representation, we show that this representation achieves a decent performance for discerning such activities with success. Higher-level activities for the LAD dataset are not explored here since those activity classes do not exhibit mirror activities.

5.4.3 Discussion

In these two experiments, we evaluated the performance of our simple activity recognition system. We used three datasets to demonstrate the effectiveness of our system. We demonstrated high performance when simple activities are modelled and recognised using the proposed representation. The CAD dataset yielded the highest performance overall and outperformed the state-of-the-art systems. CAD dataset comprises of short activities and relatively noise free object and skeleton tracking. Both levels of granularity in the CAD dataset exhibits activities which are simple, hence achieving high performance. The LAD dataset comprises of longer and more complex activities. We demonstrate a good performance on the lower-level simple activities as compared to the complex higher level activities in this dataset. This dataset only comprises of fully automated skeleton and object tracks and therefore a slightly decreased low-level performance is seen as compared to the sub-level activities in the CAD dataset. Finally, the CLAD dataset comprises of the longest and most complex activities compared to other datasets. Adding to the complexity is the crowd-sourced annotations of activities which results in multiple labels describing each activity. Therefore, activity classes are not clearly extractable. However, using partial ground-truth annotations, we extract some activity classes which resemble the lowest-level simple activities and evaluate our system on these. We show that good performance is achievable on simple activities in this dataset compared to baselines, despite the inconsistency in labels, imbalance amongst classes and lack of object tracking.

Recall that the QQSTR system proposed for modelling and recognition of low-level activities is a conglomeration of multiple feature spaces. These include qualitative relational features using RCC-3 and SQTC features (F_1, F_2) , temporal features (F_3) and quantitative features

(F_4). We assert that each of these features are designed to capture some latent abstract aspect of simple activities and that collectively they are able to achieve a high performance. We present figure 5.9 which shows performance values achieved on simple activity recognition using various combinations of these feature space. Results shown in this figure originate from best classification of simple-activities on each of the datasets.

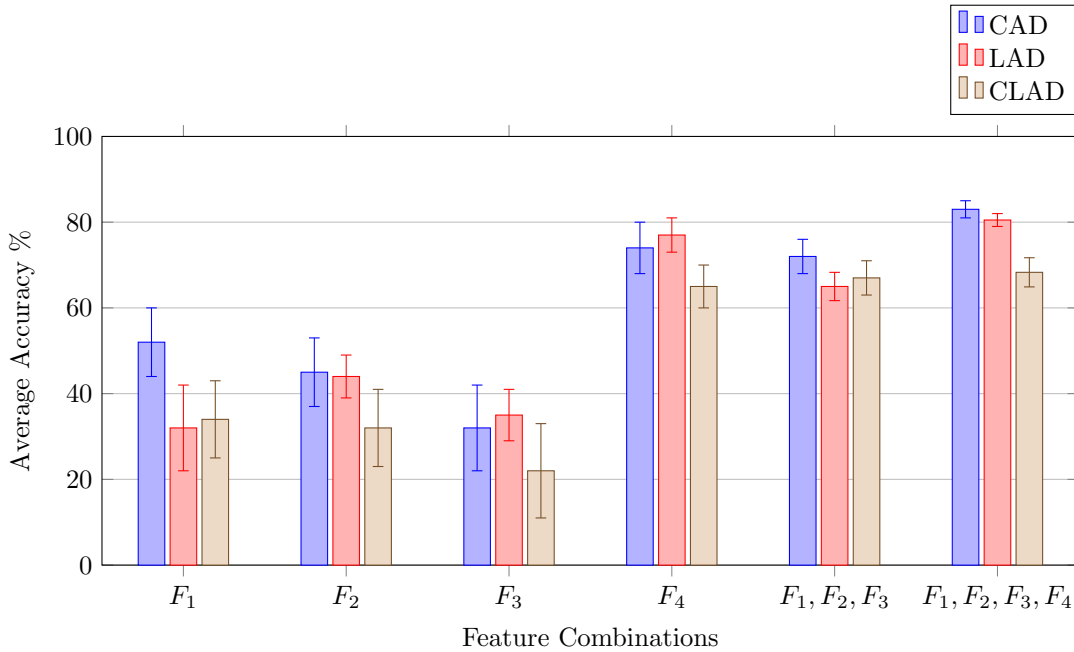


Figure 5.9: A bar chart showing accuracy values achieved on sub-level activity recognition across all datasets with different combinations of feature spaces. F_1 refers to the RCC-3 QSRs feature space, F_2 refers to the SQTTC QSR feature space, F_3 refers to the learned temporal QSRs feature space and F_4 refers to the quantitative feature space. The error bars represent the standard deviation for each accuracy value.

Interesting insight can be drawn from figure 5.9. It is apparent that each of the features individually capture some aspect of the activity resulting in modest performance values. However a general improvement in performance is evident as features are merged together. The F_4 feature refers to the quantitative features used. These features capture the distribution properties of distances between elements in an activity. It can be seen that this feature space alone achieves high performance. Features F_1, F_2, F_3 show the combination of our qualitative features. These features achieve approximately similar average performance across all datasets as quantitative features. However, ultimately combining the two types of feature spaces results

in statistically significant improvement in performance across two of the three datasets because of the combined discriminative strength. The third dataset (CLAD) does not demonstrate a large increase in performance; we believe this to be due to the high amount of noise in the class labels, skeleton tracks and lack of object tracks in this dataset. This results in insignificant contributions from the quantitative feature space (F_4) within this dataset. These results present a strong point towards our proposition of representing the activity using qualitative and quantitative representation to fully capture deep structures of an activity.

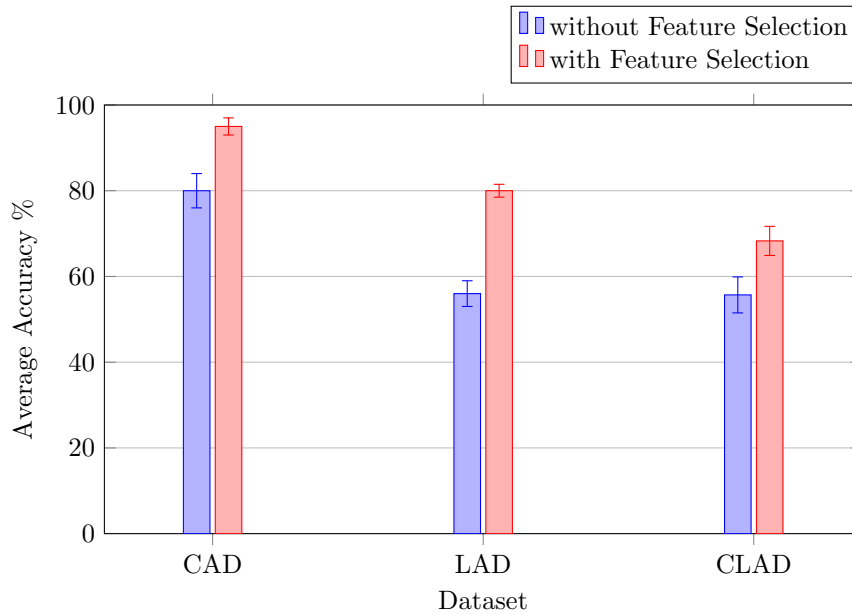


Figure 5.10: A bar chart presenting best accuracies achieved with and without use of automatic entropy-based feature selection. The error bars represent the standard deviation for each accuracy value.

The QQSTR system used a feature space F which comprises of different feature spaces F_{1-4} . As mentioned before each feature space is designed to capture latent discriminative structures within activities. However, there is bound to be overlap between representation of different features. That is to say that two feature spaces are likely to have some redundant representation amongst them. We therefore perform an entropy-based feature selection process, see chapter 3, section 3.5, to filter out non-discriminative and redundant features within the feature space. This process results in a highly discriminative feature space with lower dimensionality. Figure 5.10 represents a graph of best accuracies achieved in all datasets with and without using feature

selection. A significant improvement is seen across all datasets with the use of feature selection indicating the importance of this design decision.

5.5 Evaluation of Complex Activity Recognition

In these sets of experiments, we evaluate our approach of complex activity recognition. In this approach, we move away from discriminative modelling of activities from videos and leverage the natural hierarchical nature of complex activities to model these activities in a hierarchical manner. The premise is, given a set of low-level activity recognitions, higher level structures of activities can be recognised as compositions of lower-level actions. In a fully discriminative modelling approach, activities at each level of granularity must be modelled separately. This creates lots of redundant information being encoded between classes of low-level activities and classes of higher-level activities which comprise them. Therefore, we explore a generative method of modelling and recognising complex activities. We show in this section that this method is superior to the discriminative counter-part.

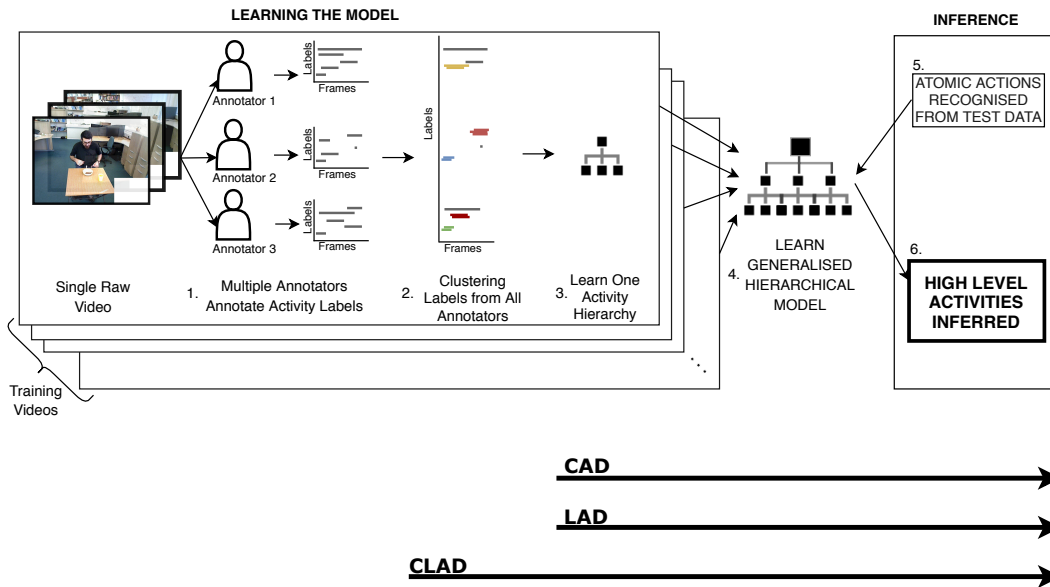


Figure 5.11: A flowchart of our complex activity recognition approach. The CLAD dataset is used to evaluate all parts of the system whereas the CAD and LAD datasets are used to evaluate all parts except the semantic clustering.

Figure 5.11 presents our complex activity recognition framework. Activity hierarchies are learned through human description of activity annotations elicited from multiple annotators. Natural variation through multiple person describing activities is utilised to learn the hierarchical structure of a complex activity. Annotations are first clustered into activity nodes to remove redundancy. A *parent-child* relationship is inferred between each pair of activity nodes to generate a temporalised hierarchy of activity nodes over a single training video. Hierarchies generate from multiple training videos are then merged and abstracted to form the generalised hierarchical model of the overall activity. This model is then used to identify low-level action classes. The QQSTR model for simple low-level action recognition is then trained to learn these action classes. Finally given a test activity video, an activity hierarchy is inferred over the automatic low-level action detections in the test video. We evaluate each of these parts using different datasets. In figure 5.11, it can be noted that the CLAD dataset is used to evaluate the entire system end-to-end whereas the CAD and LAD dataset are used to evaluate the system without the clustering mechanism since they exhibit little to no redundancy in annotated labels.

5.5.1 Experiment 3: Evaluation of Clustering Annotations

The first part of our complex activity recognition system is clustering redundant annotations resulting from eliciting annotations from multiple subjects for the same activity video. As described before, multiple subjects tend to annotate the same activity many times. For example, a video with a commonly identified activity such as ‘pick up cup’ may be annotated as ‘grab mug’, ‘lift cup’, ‘retrieve mug’ etc. When learning a hierarchy of events from videos, we require each annotation to refer to one activity only to avoid confusion when inferring *parent-child* relationships between annotations entries (activity nodes). We therefore propose semantic similarity based clustering which uses the semantic similarity between linguistic labels $\lambda^l(l_i, l_j)$ as described in chapter 4, section 4.3.3. In this experiment, we evaluate our semantic clustering against known methods, such as k-medoids, affinity propagation and spectral clustering. We use various commonly used metrics to validate the results against ground-truth cluster labels. Furthermore, we evaluate the effect of semantic similarity included within the clustering distance metric. Note that, as mentioned before and illustrated in figure 5.11, only the CLAD

dataset presents high amounts of inconsistency and redundancy between labels stemming from crowd-sourced annotators. The other datasets were annotated by expert annotators with research knowledge and therefore do not suffer from redundant or inconsistent labelling. We therefore focus our evaluation of the clustering mechanism on the CLAD dataset.

Using previously introduced notation, given a video with combined annotations across multiple annotators, we denote the annotation set as $\mathbb{A} = \{\iota_1, \iota_2, \dots, \iota_{N_D}\}$, where ι_i denotes any annotation instance comprising of start time, end time and a single English label or phrase describing the activity occurring within that timeslot. We utilised the agglomerative clustering algorithm with complete linkage which used our devised distance measure $\delta(\iota_1, \iota_2)$ that generates a distance value between any two annotation instances based on start, end time and linguistic semantic similarity to produce a pairwise distance matrix between all instances denoted by \mathbf{D} . The algorithm produces a cluster $\Theta = \{\theta_1, \theta_2, \dots, \theta_{N_D}\}$ assigning a cluster label to each instance. To evaluate this clustering, we generated a ground-truth clustering Θ^* of the annotation instance. Given Θ and Θ^* , we are able to compute similarity metrics namely normalised mutual information (NMI) [Vinh et al., 2010], homogeneity (HOM), completeness (COM) and v-measure (VM), between the two lists of assignment for assignments each video. We then average the results for each metric across all videos to generate the final metric of performance. Table 5.7 shows the results obtained through clustering.

Method	NMI	HOM	COM	VM
K-Medoids using temporal boundaries	0.65	0.66	0.63	0.64
Affinity Propagation	0.67	0.65	0.68	0.66
Spectral Propagation	0.77	0.75	0.75	0.75
Our Clustering Method without using semantic similarity λ^l	0.88	0.83	0.86	0.84
Our Clustering Method with using semantic similarity λ^l	0.95	0.92	0.91	0.90

Table 5.7: Performance of our clustering method, which uses the proposed distance measure between activity annotation intervals, compared to other commonly used clustering methods is presented. The metrics used are normalised mutual information (NMI), homogeneity (HOM), completeness (COM) and v-measure (VM).

Results and Discussion

To compare our results, we employ commonly used clustering methods, each of which use distance matrices as input. The distance matrix, denoted by \mathbf{D} in chapter 4, section 4.3.3, captures the pairwise distances between data points. The data points in our case are the activity annotation instances. We compare our method with a variation of the common k-means clustering called k-medoids [Park and Jun, 2009] which utilises a distance matrix instead of data points. Affinity propagation [Frey and Dueck, 2007] and spectral clustering [Ng et al., 2002] are other common clustering methods in the literature that use distance matrices as input. In order to make a fair comparison, we use a slight variation of k-means algorithm known as k-medoids. This algorithm uses a distance matrix between data points. We selected methods which use a pairwise distance matrix to perform clustering on because these closely relate to our approach. As can be seen from table 5.7, we show that our clustering mechanism is superior to common methods of clustering such as K-medoids, Affinity propagation and Spectral clustering. We also demonstrate that a high performance of 0.95 and 0.90 values for NMI and v-measure are achieved with the inclusion of semantic similarity computation of labels. Recall that semantic similarity refers to the linguistic similarity of labels. It is expected that activities which are labelled over the same temporal slot but do not exhibit any semantic similarity are not clustered as their distance values are penalised. This behaviour is evident through the high performance our clustering mechanism achieves as seen in table 5.7.

Qualitative Evaluation

To complement the results reported in table 5.7, we perform qualitative evaluation by random sampling of clustered labels across the dataset and visually inspecting them. In figure 5.12, we show 5 randomly selected clusters across the CLAD dataset, each of which has more than one data point. The clusters are extracted from each of the top-level activities of the CLAD dataset. Recall that the three top-level activities in the CLAD dataset are ‘having breakfast’, ‘lunch in a restaurant’ and ‘working in an office’.

As seen in figure 5.12, most clustered labels within each cluster exhibit high amount of linguistic similarity. Further investigation shows a high overlap between temporal boundaries

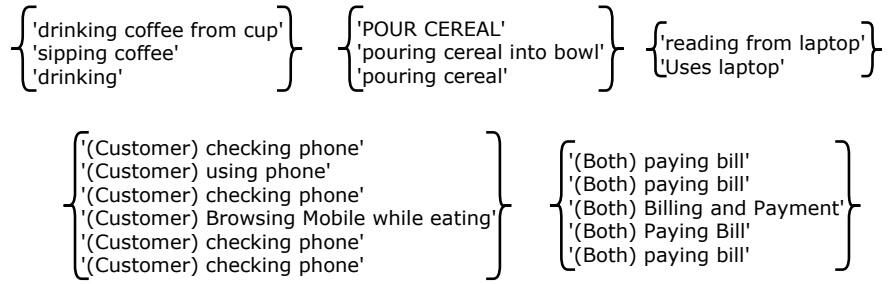


Figure 5.12: Sample output of clusters formed through our clustering mechanism. Note that linguistically close clusters are formed where semantically similar labels are merged into the same cluster.

of the clustered label validating the system’s ability to perform linguistic semantic similarity based clustering of redundant labels into exclusive clusters.

5.5.2 Experiment 4: Evaluation of *Parent-child* Relationship Inference

In this section, we evaluate our method for inferring a temporised hierarchy from human annotation data, based on optimum pairing of activity nodes (annotation clusters) as described in chapter 4, section 4.4. The task is described as inferring the *parent-child* relationships between any pair of activity nodes. Inferring this relationship between all pairs of activity nodes results in a hierarchy of activity nodes. We perform this inference based on the temporal overlap between intervals and linguistic similarity between the label sets of each activity node ω . We devise a cost function $C(\omega_c \triangleleft \omega_p)$ which assigns a cost for pairing activity node ω_c as a child activity of activity node ω_p . An optimisation is performed to find the optimum pairing between activity nodes which minimises the overall cost. Finally, the inferred hierarchies are augmented with Allen’s Interval temporal nodes amongst the children activity nodes of each parent node to form temporalised activity hierarchy denoted by \hat{H} . An optimum temporalised hierarchy is learned for each training video.

Individually-learned hierarchies per video are evaluated for correctness using ground-truth hierarchies. A hierarchy is represented as $H = (\Omega, E^\Omega)$, where Ω is a set of all nodes in the hierarchy and E^Ω is a set of learned directed edges (ω_i, ω_j) between the activity nodes $\omega \in \Omega$.

The directed edges correspond to the *parent-child* relationship between activity nodes. In order to evaluate the correctness of the hierarchy, we evaluate each inferred edge for correctness. Expertly annotated ground-truth edges are labelled for each hierarchy. The accuracy metric of a hierarchy is simply defined as the ratio of correctly inferred edges with all edges in a test hierarchy. We compute the accuracy of hierarchies inferred for each video in the datasets and report the average accuracies in table 5.8. To evaluate our system, we devise a well-defined baseline. As described in chapter 4, learning graph-like structures from annotations of activities has been previously explored in literature [Sridhar et al., 2010b, Sridhar et al., 2008]. We have previously shown that there exists a graph class from graph theory which arranges activity nodes in a hierarchical structure. These graphs are known as trivially perfect graphs (TPGs). TPGs can be created by assigning a *parent-child* relation edge between any two activity nodes where one is *fully* subsumed by the other. This is a stricter pairing paradigm that does not handle i) the variability in start and end times, ii) proper pairing of parallel activities and iii) semantic analysis of parent’s and the corresponding children’s labels. Trivially perfect graphs serve as a baseline point of comparison for our evaluation. The results are presented for all three datasets in Table 5.8.

METHOD	LAD	CAD	CLAD		
			<i>using semantic activity node matching λ^ω</i>	<i>using multi-label activity nodes</i>	<i>using single-label activity nodes</i>
Baseline (TPGs)	68%	24%	✓	61%	58%
			×	53%	53%
Learned Hierarchy	91%	89%	✓	79%	70%
			×	63%	63%

Table 5.8: Performance of our hierarchy learning approach which automatically infers the *parent-child* relationship between activity nodes is presented for all datasets. Our method is compared to a well-defined baseline which uses trivially perfect graph to infer the *parent-child* relationship. Also, the use of semantic activity node matching and the use of multi-label representation of activity nodes is evaluated and results reported.

Results and Discussion

Many different aspects of our hierarchical learning method are demonstrated in table 5.8. Firstly, it can be seen from the table that the LAD and CAD datasets achieve significantly high accuracies of 91% and 89% respectively compare to the baseline accuracies of 68% and 24%. This is expected since these datasets do not exhibit high amounts of temporal boundary noise and inconsistent labels. However, some temporal boundary noise is present as evident by lower baseline accuracies.

Next we evaluate the CLAD dataset, which exhibits multiple labels per activity nodes and utilises semantic similarity between linguistic labels to match activity nodes using the previously defined function λ^ω . We have presented accuracy results with using semantic similarity based activity node matching, indicated by \checkmark , and traditional string matching between activity nodes, indicated by \times . The ‘string matching’ acts as a baseline to evaluate the advantage achieved with using our proposed semantic similarity based activity node matching. There is a clear performance increase in all 4 cases where semantic activity node matching is used over simple string matching. An accuracy increase of 5%-16% is seen across all cases in baseline (TPG) and our learned hierarchy inference method. This indicates the advantage of using the semantic similarity of English linguistic labels over simple string matching to allow for variation in activity descriptions. We mentioned before that semantic similarity between two linguistic labels is an open research problem and therefore does not result in fully accurate similarity scores. Despite noisy semantic similarity values, it seems some notion of similarity is captured through the use of our activity node matching λ^ω . This is evident by the performance increase shown in table 5.8.

A design decision in representing activity nodes within a hierarchy was made to represent each activity node ω as a 3-tuple (\bar{s}, \bar{e}, L) where \bar{s} is the average start time, \bar{e} is the average end time and a label set L which is a conglomeration of all labels of clustered annotation instances in that activity node. Representing each activity node as a set of labels resulting from clustering instead of a single label has two advantages. First, having multiple synonymous labels captures deeper knowledge about the activity represented by the activity node, at least in terms of language. Second, the number of labels in the activity label set L of an activity

node reflects the confidence value of that activity node since it has been identified by that many annotators. This is appropriate since an activity which has been labelled by many annotators is more likely to be a correct labelling rather than one with few labels. This knowledge is implicitly used in our activity node matching λ^ω system where highly imbalanced activity nodes in terms of the number of labels are penalised accordingly. In order to validate our representation of activity nodes using multiple labels, we experiment on the CLAD dataset where each node is represented by multiple labels and compare it to a modified dataset where each node in the hierarchy is compressed to be represented by only one single label. The single label is simply selected as the first label in each node. The results in table 5.8 show that using multi-label activity nodes captures deeper knowledge compared to single label activity-nodes. Obviously, the results remain unchanged in the ‘not using semantic similarity λ^ω ’ since exact matching is used in these cases and the label list sizes are not considered to match activity nodes. But in the case where semantic similarity is utilised, we notice improvement in both the baseline and proposed hierarchy learning methods.

Finally, we can observe that overall accuracy of the CLAD dataset outperforms the baseline (TPGs) method since an accuracy of 79% is achieved over an accuracy of 61% when using our proposed hierarchy inference method. The effect of noisy and ambiguous annotations is apparent from the significant difference in a crowd-sourced labelled CLAD dataset versus clean present in-house labelled LAD and CAD datasets. Despite the added challenge of incorporating linguistic semantics to find the optimum pairing in the CLAD dataset, a decent performance is still achieved as seen when compared to the baseline.

Qualitative Evaluation

In this section, we present some well inferred activity hierarchies from each dataset. The aim here is to validate correctness of our system in a qualitative manner by visually observing the output and confirming correct logic. Figures 5.13, 5.14 and 5.15 present six learned temporalised hierarchies \hat{H} for the CAD, LAD and CLAD datasets. Two temporalised hierarchies are shown for each dataset. Sample image streams of some low-level actions are presented below each hierarchy; higher-level activities are a conglomeration of these low-level action streams. Note that, in keeping with previously used colour scheme, the red coloured nodes represent the parent

nodes whereas green coloured nodes represent low-level actions or terminals in the hierarchy.

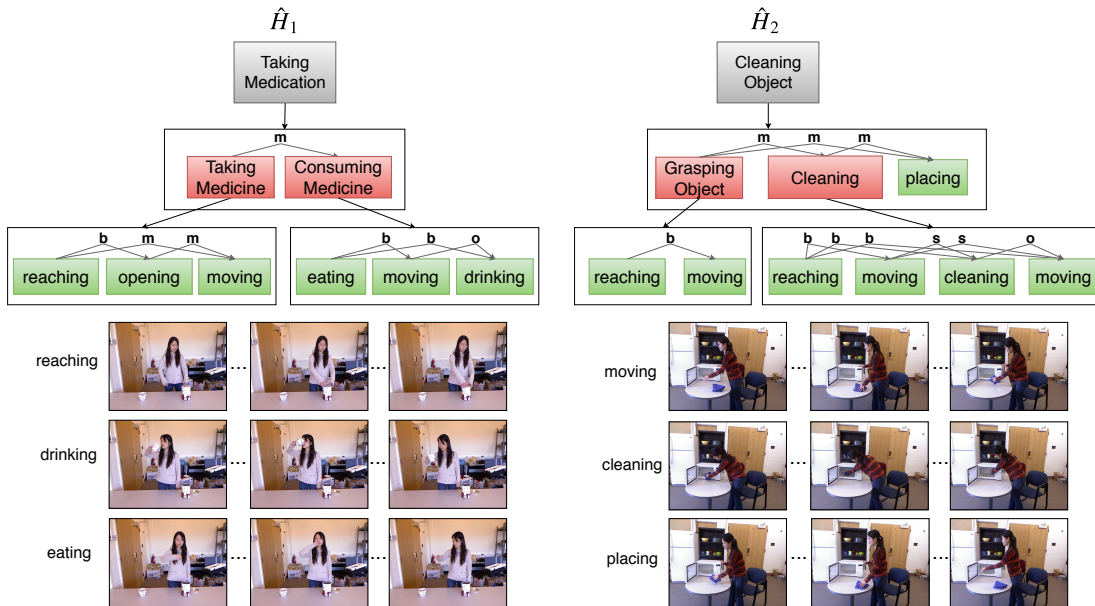


Figure 5.13: Example of temporalised hierarchies inferred from two training videos of the ‘Taking Medication’ and ‘Cleaning Object’ activities in the CAD dataset.

Figure 5.13 shows inferred hierarchies for two of the 10 top-level activities, namely ‘Taking Medication’ and ‘Cleaning Object’. Recall from figure 4.10 in section 4.4.4 that the final representation of learned activity hierarchies from single training videos are in the form of temporalised hierarchies as shown in the figures in this section. This representation shows inferred children activity nodes, of each parent activity node, as activity clusters encompassing the interval graph of sibling activities. We can notice that, in figure 5.13, most *parent-child* relationships between activity nodes have been correctly inferred. For example, ‘Taking Medication’ is a parent activity of ‘Taking Medicine’ *meeting* with ‘Consuming Medicine’. Each of those are further learned as parent activities of low-level actions such as ‘reaching’, ‘opening’ etc. correctly. Similar accuracy can be seen in figure 5.14 which shows two activity hierarchies learned from two training videos from the LAD dataset. A deeper hierarchy is seen in \hat{H}_1 of this figure, where ‘Prepare Coffee’ activity is learned to be a combination of ‘pickup’ and ‘pour’ sub-activities. Finally the CLAD dataset temporalised hierarchies shown in figure 5.15 which presents the hierarchies from ‘having breakfast’ and ‘lunch in a restaurant’ activities. We notice that deep hierarchies are learned in these examples as well as temporally complex ordering of

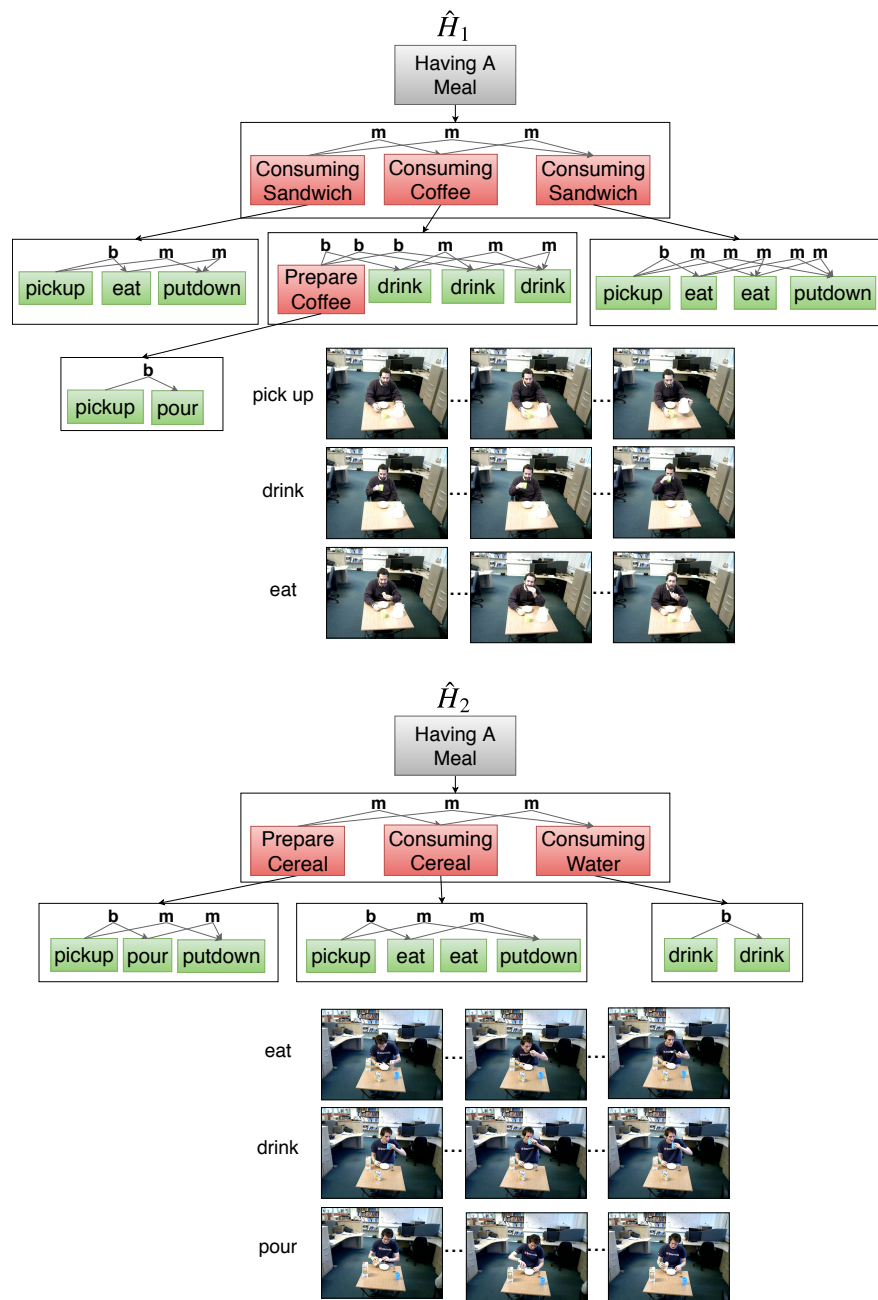


Figure 5.14: Example of temporalised hierarchies inferred from two training videos of the ‘Having a meal’ activity in the LAD dataset.

component activities are learned. This is evident as temporal relations such as overlap (o) are encoded along edges in interval graphs within activity clusters. We further notice that each activity node is represented by a set of labels. Moreover, in case of \hat{H}_2 in figure 5.15, this activity

is performed by multiple actors. The information as to which activity is being performed by which actor is encoded as a suffix within activity nodes as either customer (C), waiter (W) or Both (B). Generally, correct pairing of activity node’s *parent-child* relationships is also evident in most cases within this hierarchy.

We can also observe some inaccuracies in learning of *parent-child* relationships within these temporalised hierarchies. For example in figure 5.13 notice that the low-level action of ‘placing’ is incorrectly paired with the root activity making it as a sibling of ‘Grasping Object’ and ‘Cleaning’ activities. Also in figure 5.15, we observe that spurious activities such as ‘thinking’ or ‘stare at camera’ are misaligned and learned as child activities of ‘having breakfast’ rather than being identified as noisy instances. We rely on our probabilistic activity grammar model to filter out these instances as unlikely occurrences, though complete removal of such activities at this stage is one of our future aspirations.

Our last observation is on handling parallel activities. Specifically, observe the temporalised hierarchy \hat{H}_1 of the CLAD dataset in figure 5.15. Note that the ‘reading newspaper’ activity was observed in parallel to the ‘eating breakfast’ activity. Estimating the *parent-child* relationship between these activity through temporal overlap alone would have resulted in ‘reading newspaper’ incorrectly paired as the child activity of ‘eating breakfast’. This incorrect inference would have propagated further into sub-activity where ‘flip page’, ‘turn page’ etc. sub-activities would have been learned as children of other activities within the ‘having breakfast’ sub-hierarchy. However, using our cost-based optimisation, which not only relies on temporal overlap but also the node similarity based on linguistic closeness of labels, our system was able to correct *peel away* the ‘reading newspaper’ activity and all of its associated activity hierarchy from the ‘eating breakfast’ activity. This resulted in the correct pairing and estimation of both activity hierarchies as shown in \hat{H}_1 of figure 5.15.

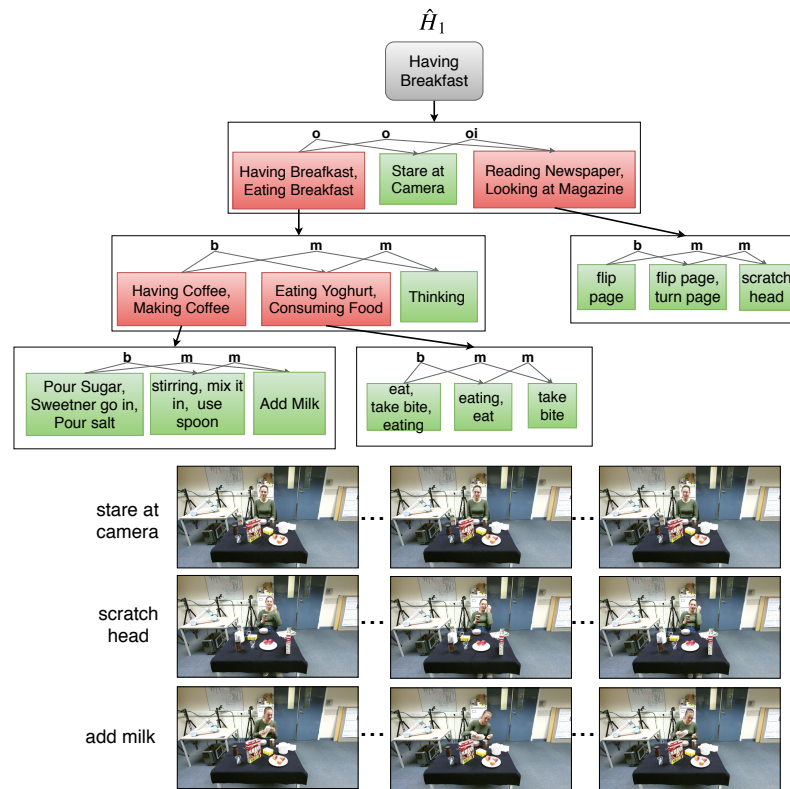


Figure 5.15: Example of temporalised hierarchies inferred from two training videos of the ‘Having Breakfast’ and ‘Lunch in a Restaurant’ activities in the CLAD dataset.

5.5.3 Experiment 5: Evaluation of Recognition of Complex Activity Hierarchies

The final stage of modelling complex activities is learning a probabilistic generalised activity model from temporalised hierarchies. The temporalised hierarchies generated from training videos have been evaluated in the previous experiment. These are used to learn the generalised activity model. In chapter 4, section 4.5, we presented a simple algorithm inspired from the NLP literature to treat each temporalised hierarchy as a language parse tree. Similar to inferring a tree-bank grammar from a set of parse trees, we present an algorithm to infer a generalised activity hierarchical model which is represented as an activity grammar \mathbb{G} . This model is used to extract low-level action classes which are trained using the QQSTR method. This training allows us to perform low-level action recognition in test video in order to generate an input stream of low-level actions. The input stream of low-level actions is then used to infer the activity hierarchy which best describes the low-level actions. In this experiment we evaluate the different aspects of the recognition system, specifically our contributions.

To evaluate our complex activity recognition system, we utilise a commonly used metric in literature known as the correctly parsed terminals (CPT) [Zhang et al., 2011]. Recall that terminals refer to the lowest-level actions in the generalised activity model. In NLP, these refer to the ‘words’ in a sentence. The recognition task, as explained before, is to infer an activity hierarchy, similar to a parse tree, over the input action stream. The CPT metric, mentioned in section 5.2.6, provides a ratio of the correctly parsed terminal to all terminals in the input action stream. We evaluate the recognised hierarchies by manually validating each inferred activity hierarchy parse tree by computing its CPT value. Recall from equation 5.6, the CPT accounts for correctly detected insertion errors, within the metric. Deletion and substitution errors are handled in the partial parse tree selection mechanism evaluated later in this chapter.

Having selected a reasonable metric for evaluation, we now set a baseline to compare our method against. Similar to the baseline selection in experiment 5.5.2, which was selected as a naive method present in the related graph theory literature, we set the baseline in this experiment as a relatively naive parsing mechanism present in the NLP literature. The baseline is devoid of all the proposed approaches aimed for abstracting a generalised hierarchical model and

recognition. Incremental addition of all proposed systems is then performed, and corresponding CPT values reported. Tables 5.9 and 5.10 present the results achieved from recognition of activity hierarchy parse trees using proposed contributions for automatically learned and recognised low-level input action stream as well as ground-truth low-level input action stream. We perform leave-one-out cross validation for each dataset and average the CPT values obtained across each fold.

Method	LAD (CPT)	CAD (CPT)	CLAD (CPT)
<i>assuming automatic low-level action stream recognition</i>			
1. Baseline (uses sequential grammar rules and input stream)	0.77	0.72	0.39
2. Our Method using temporally complex rules and single action tree recognition	0.75	0.79	0.70
3. Our Full Method using temporally complex rules and multiple partial activity tree recognition	0.86	0.89	0.72

Table 5.9: Performance of inferring activity parse tree on test videos in each dataset shown. Correctly parse terminals (CPT) metric is used to compute the performance values. Our method is incrementally equipped with proposed features and compared with a well-defined baseline which uses a relatively trivial mechanism. The table presents results with using automatic recognition of low-level action stream.

Results and Discussion

In tables 5.9 and 5.10 results are presented for three methods. Method 1 is our selected baseline method. Recall that a learned activity grammar consists of rules which comprise of a left hand side (l.h.s) and a right hand side (r.h.s) indicating the child activity nodes which expand the parent node in the l.h.s. Our extended grammar constructs the r.h.s using a set of child activity nodes Q and a temporal relations matrix \mathbf{R} which captures the complex temporal structure amongst the activity nodes in Q using Allen’s temporal relations. In our baseline system of complex activity recognition, we remove temporal relations matrix \mathbf{R} from the 1) the activity

Method	LAD (CPT)	CAD (CPT)	CLAD (CPT)
<i>assuming ground-truth low-level action stream recognition</i>			
1. Baseline (uses sequential grammar rules and input stream)	0.71	0.81	0.42
2. Our Method using temporally complex rules and single action tree recognition	0.83	0.83	0.69
3. Our Full Method using temporally complex rules and multiple partial activity tree recognition	0.94	0.97	0.85

Table 5.10: Performance of inferring activity parse tree on test videos in each dataset shown. The correctly parsed terminals (CPT) metric is used to compute the performance values. Our method is incrementally equipped with the proposed features and compared with a well-defined baseline which uses a relatively trivial mechanism. The table presents results with using ground-truth recognition of low-level action stream.

grammar and 2) the low-level action input stream. This results in an activity grammar that assumes sequential order of sub-activities in set Q and a sequential order of input action stream. Given these, we are able to utilise the simple off-the-shelf Earley parser to parse a single tree over the input action terminal stream. This method was demonstrated in figure 4.11 in section 4.5.1. Note that the basic Earley parser returns no results if a complete parse tree, reaching the ‘root’ node, cannot be found. In method 2, we introduce the use of temporally complex grammar rules, which exhibit \mathbf{R} relations matrix encoding the temporal relations between child sub-activities of a parent activity. Furthermore, we also utilise a temporally complex input stream of low-level actions. This allows for encoding parallel activities, overlapping activities and so on. Having a temporally complex input stream and grammar rules requires us to use our activity cluster matching λ^ψ as introduced in chapter 4. Therefore, the results reported for this system jointly demonstrates the utility gained from including temporal information within the extended activity grammar as well as the fuzzy matching mechanism between activity clusters, or right hand sides, during parsing. A high value of the CPT metric in this table also implicitly validates the correctness of the learned activity grammar rules. Also note that, as described in

chapter 4, we are required to use the multi-threaded parsing extension of the Earley parser, as proposed by [Zhang et al., 2011], to be able to parse these complex input streams. It can be seen that CAD and CLAD datasets gain a superior performance to the baselines of 0.79 over 0.72 and 0.70 over 0.39 respectively. Note that the proposed method on the CLAD dataset performs significantly better than the baseline as compared to the CAD dataset. The first reason for this is activities included in the CLAD dataset are longer and exhibit more complexity than CAD dataset. Also, the CLAD dataset exhibits multiple subjects within the same activity. Both these properties result in sub-activities that are observed in parallel, thereby exhibiting a temporally complex sequence. This sequence is therefore better captured with the inclusion of temporally complex rules and activity cluster matching mechanism resulting in a higher performance increase. The LAD dataset drops in performance slightly, which may indicate an over representation of the temporal sequence between the sub-activities of the LAD dataset, since most of the sub-activities of this dataset exhibit a sequential flow and therefore generate better performance using a naive system.

In method 3, we use our full approach which includes temporally complex rules and utilises activity cluster matching mechanism during multi-threaded parsing as introduced in method 2. This method further includes our proposed integer linear programming (ILP) approach of selecting an optimum set of partial parse trees, detailed in section 4.6.2. This is in contrast to the systems proposed so far which selects one activity parse tree with the highest probability as the description of the low-level action stream. Selecting multiple partial trees describes a larger proportion of the input low-level action stream, see figure 4.16; this is because low-level actions which are not describable by any one activity tree are attributed as deletion, insertion or substitution errors. Multiple activity trees circumvent this issue by describing sub-sets of the input stream using the least number of trees which maximises probability. The effects of including this feature is apparent from the high increase in performance over the previous methods in all datasets. The results of the full method on the LAD dataset, though exhibiting a drop in performance over the baseline in method 2, outperforms the baseline and method 2 with method 3 by generating a CPT value of 0.89. This is also true for the CAD and CLAD datasets, producing CPT values of 0.89 and 0.72 respectively. These results validate the use of an optimisation based system to return multiple activity parse trees rather than a single

parse tree.

Results discussed so far are all produced from activity parse trees generated over automatic detection of low-level actions within the test video. Note that, we assume temporal segmentation of activity before performing automatic detection of low-level actions within each video. The performance of low-level actions recognition using the QQSTR system has been reported in experiments in section 5.4. However, to fully demonstrate the strength of our complex activity recognition system, we use ground-truth low-level action streams. These are hand identified low-level action within each video. Using this ground-truth input stream eliminates the errors propagated through inaccurate results from automatic low-level action recogniser therefore allowing to report the upper-bound of our complex activity recognition system. These results also demonstrate our system's ability of using any low-level action recogniser to generate an input stream over which to infer complex activity hierarchies. Table 5.10 presents the results achieved using ground-truth action recognition. We notice a significant improvement across majority of the methods over all datasets. The CLAD dataset however presents lower performance compared to other datasets. This is because of the noise present in semantic similarity computation of linguistic labels since annotations in the CLAD dataset are inconsistent and require a clustering pre-processing step. Other datasets, achieve significantly high performance which indicates between inference of high-level activities using our complex action recognition system as compared to the recognition of high-level activity using the QQSTR system presented in experiments in section 5.4. We therefore assert that a higher performance is achievable in both LAD and CAD datasets using our complex activity recognition system, shown in table 5.10, as compared to the QQSTR system. This validates the preference of our complex activity recognition system which uses generative models to infer higher-level activities using low-level action recognition, rather than discriminatively modelling high-level activities using a feature-based approach. It is worth noting that all analysis of datasets, training and recognition of activity hierarchies were performed on midrange computer units. Our testing machine comprised of an Intel Core i7-4790 processor with 8 cores having a clock speed of 3.6 GHz and 16 GB of RAM. In the worst-case, our activity hierarchy recognition system processed the longest complex activity video, from the CLAD dataset, in under 4 minutes. On average, remaining testing videos were processed in significantly lesser time of around 2 minutes.

5.5.4 Conclusion

In this chapter, we evaluated our approaches of simple activity recognition as well as complex activity recognition. We used three datasets which exhibited varying levels of complexity, videos of natural human activities at varying levels of granularity to evaluate our approaches. We demonstrated that our simple activity recognition framework outperformed the state-of-the-art approaches in recognition of simple activities in one of the datasets, while in the remaining datasets, we achieved high performance as compared to the baselines. Our complex activity recognition approach was evaluated against all datasets using various well-defined baselines and commonly used metrics. We demonstrated that using our complete system, we are able to achieve significantly higher performance compared to baselines across all datasets. We further evaluated parts of our approach qualitatively by showing activity hierarchies learned from different datasets. Finally, we demonstrated the combination of the simple activity recognition system providing low-level input to the complex activity recognition system to achieve very high performance on all three datasets.

In the next chapter, we present final discussions, future work and general conclusions on the work presented.

Chapter 6

Discussion and Conclusion

In this thesis we presented a novel activity recognition system which is aimed at recognising activities of varying complexity for an intelligent autonomous system. Traditional activity recognition approaches have aimed at recognition of activities spanning short periods of time. However, as intelligent systems such as robots, autonomous cars, surveillance systems etc. gain greater computational power and perform autonomously for longer periods of time, there is an emerging requirement for activity recognition systems that models and recognises activities which span longer time periods. Motivated by this research requirement, in this thesis, we demonstrated that long and complex activities can be modelled and recognised as compositional structures whereby short primitive actions can be modelled and recognised using a highly descriptive feature space.

This chapter is organised as follows: first we summarise the methodology used in the presented frameworks, then we list our main contributions in this thesis. We then discuss potential future work that would address some of the shortcomings of the presented systems as well as propose methods to relax some of the simplifying assumptions made. Finally, we conclude the thesis with some final remarks about the work done in the field of activity recognition.

6.1 Summary

Simple Activity Recognition

We have presented two main frameworks for activity recognition. The simple activity recognition framework used a discriminative model (QQSTR) which jointly utilises qualitative and quantitative representation of interactions of entities within an activity. Qualitative representation comprised of a well-established algebra called the region connection calculus (RCC-3), which captures topological changes between entities within activities. We also introduced a qualitative representation to model the global direction of motion of entities within the activities called the simplified qualitative trajectory calculus (SQTC). The notion of temporal ordering was captured by learning a qualitative representation which captures the relative lengths of qualitative relations held within the activity. Given these representations, we further proposed a histogram-based feature extraction mechanism which captures the number of relational changes within pairs of entities, in the activity, in a fixed length.

We augmented the qualitative representation by representing lost information from over-discretisation, using the quantitative representation. This representation modelled the variation of distances between various entities within the activity by representing the distribution of distances using descriptive statistics. We modelled the distribution of distance between entity pairs using the first four moments of a distribution. We demonstrate the effectiveness of combining the two feature spaces to represent activities. Finally, we performed feature selection to remove redundant representation within the combined feature space and extract a highly descriptive feature set. A multi-class SVM was trained to learn the simple activity classes.

Complex Activity Recognition

Complex activity recognition framework leveraged the natural hierarchical nature of activities and built upon approaches from related fields which also exhibit hierarchical structures such as sentences in language. These approaches aided in representing complex activities as human-describable hierarchies of activities.

We utilised datasets that comprise of complex and long videos of activities and have been annotated by multiple annotators. For a given training video, annotations from all annotators

are first clustered to remove redundant labels and produce annotation clusters called activity nodes. Clustering is performed based on the temporal boundary of annotations as well as the semantic similarity between the linguistic labels of the annotations. This helped in avoiding clustering parallel activities which may exhibit similar temporal boundaries but vary in linguistic description.

We then presented a cost-based optimisation mechanism to automatically learn the *parent-child* relationship between activity nodes. The cost function was designed to infer this relationship based on the temporal overlap between intervals, and the semantic similarity between activity node's label sets. Similar to the motivation of using semantic similarity between labels during clustering, the use of semantic similarity between activity nodes helped to discern parallel activity and establish correct *parent-child* relationships. The inferred hierarchies were then augmented with temporal information using Allen's temporal algebra to create temporised hierarchies, where each set of sibling nodes were represented as an interval graph with temporal relations between nodes encoded along the edges of the graphs.

Temporalised hierarchies for all training video were then merged to generate the generalised activity model. The process included learning a probability distribution of activity clusters for each parent node and build a model incrementally. Similarity of two activity clusters was gauged using our inexact cluster matching mechanism. The learned model was then represented as an activity grammar.

Given the activity grammar, we first extracted low-level activity classes from this model. These classes represented the simplest actions within the hierarchy, in other words, the terminals of the activity grammar. We then trained a simple activity model (QQSTR) to learn the low-level activity classes. Given an unseen test video of a complex activity, low-level activities were first recognised in a temporally complex stream. We then used a multi-threaded parsing mechanism to utilise the activity grammar in order to generate all possible activity parse tree over the input stream of low-level actions. An integer linear programming based optimisation was then proposed to select the minimum number of partial parse tree which describe the maximum amount of input low-level actions with the highest probability. This allowed us to generate a hierarchical description of the complex activity over the input video.

6.2 Contributions

We list our main contributions in this work below:

1. The representation of a simple activity by capturing the interactions between various entities using both qualitative and quantitative representation. Qualitative representations are popular for achieving highly abstracted representation of various aspects within an activity. We showed that augmenting qualitative representation with quantitative representation jointly achieves a superior representation by capturing deeper aspect of simple activities.
2. We proposed a simple qualitative trajectory calculus which captures the global motion trend of entities within an activity. Furthermore, we proposed a variation on the run-length encoding method to obtain a global representation of interaction of activities within the scene.
3. We also proposed a method of feature extraction whereby a histogram of various qualitative relational changes between pairwise entities in the scene capture spatial and temporal aspects of an activity in a fixed-length representation.
4. For complex activity recognition, we presented an end-to-end framework which utilises annotations from multiple annotators to learn the inherent hierarchical structures of activities within activity videos. We observed that different human subjects described activities at different levels of granularity. We leveraged this to learn hierarchical structure of complex activities.
5. We proposed a mechanism to cluster annotation labels to remove redundancies across annotations from multiple annotators. The system utilises the temporal boundary of the annotation labels along with the linguistic similarity between labels.
6. Inference of *parent-child* relationship between two activity nodes, or annotation label clusters, is a key contribution of our work. We propose a cost-based optimisation that is able to learn the optimum temporalised hierarchy of activities from annotations within

a video, without explicit knowledge of *parent-child* relationships between activity annotation intervals. This process also included a novel mechanism for matching two uneven sets of linguistic labels.

7. Abstraction of a generalised activity model from temporalised activity hierarchies is another contribution of our work. We present an inexact activity cluster matching mechanism to abstract a model which captures the hierarchical structure of activities along with probabilistic distribution over the different variations of the complex activity.
8. Finally, we propose an optimised way of selecting a highly suitable set of activity parse trees over an input stream of temporally complex low-level actions generated using the learned activity grammar.

6.3 Future Work

Though we have demonstrated a system capable of performing the activity recognition at different levels of granularity and complexity, there are areas which can be improved and built upon. In this section we present some possible directions where our work can be improved; we also highlight the assumptions made and how these may potentially be relaxed.

Object Representation

In future work, we aspire to achieve a more cohesive representation of objects within our framework. In the current frameworks of simple activities, we represent classes as the activity label without including object types for example ‘pick up mug’ and ‘pick up milk’ are both classed as ‘pick up’ while discarding the specific object identifier. This method works well as the inherent activity of ‘pick up’ is learned rather than an object specific class. However, in our complex activity recognition system, our model recognises higher-level activities upon detections of lower-level actions. For example, an activity ‘Taking Medicine’ may comprise of ‘pick up medicine’ followed by ‘eat medicine’ followed by ‘put down medicine’. A different activity of ‘Eating Apple’ may comprise of ‘pick up apple’ followed by ‘eat apple’ followed by ‘put down apple’. In both activities, the set of basic comprising actions is the same ‘pick up’

followed by ‘eat’ followed by ‘put down’. We therefore train our low-level recognition models **with** object specific classes of activities, to discern between inferring respective higher-level activities.

A future area of research is to represent objects differently within the low-level activity classes, such that detection of raw action classes alone can be associated to the correct parent activity based on a parametrised object argument. One way is to represent object affordances rather than exact objects, as explored in [Koppula et al., 2013]. These affordances can be propagated higher up in the activity hierarchy as parameters associated with every inferred parent activity. The affordances or abstracted object labels can be learned from observation or from databases as explored in [Sridhar et al., 2008, Young et al., 2017].

Repetition of Sub-activities

In our complex activity framework, we learn hierarchical structure of activities from annotated training videos of human subjects performing activities. Our final model is an activity grammar which conceptually encodes the components of each parent activity. For example, assuming sequential ordering, the ‘consume water’ activity may be learned to have an expansion into ‘take sip’ followed by ‘take sip’. The corresponding rule would take the form:

$$\text{‘consume water’} \rightarrow \text{‘take sip’}, \text{‘take sip’}$$

Observing the same activity in a different training video or a different part of the existing video may generate the following rule:

$$\text{‘consume water’} \rightarrow \text{‘take sip’}, \text{‘take sip’}, \text{‘take sip’}$$

The definition of this rule is different to the first since the observation comprised of taking three sips rather than two sips. In our current system, these two instances will be learned as two separate variations of the ‘consume water’ activity and will be encoded in the final model as such, with a probability value associated to each variation learned from the data. We propose a direction of future work where, similar to grammar rules in language, in order to detect activities which exhibit repetitions by representing them using a parametrised recursive rule format rather than the current representation. This would achieve a higher level of abstraction in our final activity grammar. In language grammars, these recursive rules take the form:

$$\text{NP} \rightarrow \text{ADJ, NP}$$

Such rules are able to expand NP into any number of adjectives yet still complying by the rule. In our example of ‘consuming water’ above, we can rewrite the two rules as:

$$\text{‘consuming water’} \rightarrow \text{‘take sip’}, \text{‘consuming water’}$$

This way there can be any number of ‘take sip’ operations which will complete the ‘consuming water’ activity. Parametrisation of this recursive rule could involve learning limits on the number of times these repetitions can be observed from training data. Inclusion of such rules within the grammar would require a slightly modified parsing mechanism where a probability threshold would need to be set to stop infinite grammar expansions during top-down parsing methods. Another challenge associated with this approach, is to identify such repetitive activity rules within an activity grammar. Not all activities exhibit multiple repetitions and therefore determining which activity need to be allowed recursive rules is a challenge. Furthermore, an augmented representation of the associated temporal relations matrix would be needed to capture the internal ordering of the right-hand-side elements of an activity rule. We believe that this is a lucrative extension of our representation and an interesting direction to further research to explore.

Temporal segmentation of sub-activities

A complete end-to-end system of complex activity recognition would ideally take as input a video of a long and complex activity and output the human-describable activity hierarchy comprising of detected activities at various levels of granularity along with the *parent-child* relationships between activities encoded as directed edges in the hierarchy. We have demonstrated that our framework achieves this task. One of our simplifying assumptions is that we assume temporal segmentation of low-level actions. That is to say, the input video has been previously hand segmented whereby temporal boundaries of all low-level activities have been defined. Note that there is no information of what activity class lies within the temporal boundaries, this is automatically inferred by our QQSTR based low-level activity recognition system.

A future direction of research here is to introduce a system which is able to not only detect low-level actions within pre-set temporal windows in a test video, but also detect the location of

such actions. This would allow us to relax the requirement of temporal segmentation of actions and build a fully automated system of complex activity recognition.

Improved Datasets

Future work also proposes recording larger datasets which comprise of longer and more complex videos. Specifically, we aim to record datasets which move away from a laboratory environment where actors are replicating activities, into a more natural setting where natural human behaviour is observable. This would provide a greatly challenging dataset to test our methods on.

Longer videos and more instances of complex activities need to be recorded. A larger number of instances would allow our framework to learn activity grammars which capture the most common and likely behaviour of humans and learn accurate probability distributions over the variations of parent activities. Following from this, activity grammars can then be pruned to remove low-probability expansions of parent activity attributed as noise. Longer videos are also needed to learn deeper hierarchies of activities which span over many hours or potentially days. Collecting such a dataset is therefore one of our aspirations for future work.

Improved semantic similarity

In our complex activity recognition framework, we learn from human annotations of activities. We used a dataset which has been annotated by multiple annotators providing us with latent knowledge of the hierarchical structure of activities within the annotation. In order to process different annotations, we were required to match linguistic labels to gauge their conceptual similarity. We utilised a method, commonly used in the NLP literature, of finding a quantitative score which represents the conceptual similarity between two English phrases. For example, an activity labelled as ‘paying the bill’ and ‘using card machine’ may refer to related activities. Since this area of research is not the focus of this thesis, we used an off-the-shelf method to perform these comparisons.

We noticed that most errors and inaccuracies originating in our final activity grammar were due to inaccuracies in the off-the-shelf semantic similarity computation. Therefore, a future direction of research stemming from the sub-par performance of the semantic similarity

computation is to improve this measure by comparing visual representation of corresponding videos of activities. In other words, comparing two labels yield a similarity score based on language similarity by the borrowed system. In our domain we aim to improve this score by comparing the qualitative representation of activity video segments of the labels along with the semantic similarity. Similar activity labels must not only exhibit high linguistic similarity but also exhibit high qualitative representation within the activities.

Another area of future research is to improve the current semantic similarity measure by transforming it into a metric. Currently, the similarity values produced by the borrowed method is not a metric i.e. it does not follow the triangular inequality property of a metric. This forces us to indirectly include this measure within other distance metrics used during clustering or activity node matching. In the literature, this problem is known as the matrix nearness problem [Brickell et al., 2008, Sra et al., 2005]. It proposes an approximation of non-metric distance/similarity matrix into a matrix which follows all metric properties. Given this approximation, a future direction could be to use such metric similarity values that can be incorporated into other metric computation in a mathematically natural way, resulting in better representation of similarity.

6.4 Final Remarks

In summary, we have introduced novel methods for representation and recognition of human activities. As human activities vary in length, different approaches are introduced to tackle short activities with few entities and interactions and long activities which are composition of shorter activities. We provided a novel representation of simple activities by utilising qualitative spatio-temporal relationships and quantitative measures to encode various latent aspects of simple activities. As activities become lengthier, they can be naturally described as a hierarchical structure of composing activities. We proposed a new method to infer the hierarchical structure of activities from human annotations. We further present a method to abstract a generalised model of activities which captures the variations in activities structure from training data, the likelihood of each variation and the overall hierarchical structure. This model is represented as an extended activity grammar. We then extracted the classes of primitive actions from the learned model which resemble simple building blocks of more complex activities. Having

already presented a method for simple activity recognition, we utilised this method to learn a model of the extracted primitive actions. Finally, we proposed a method to recognise the activity hierarchy over an unseen test video of a long activity using the learned model and the detected primitive action stream.

In light of the results obtained, we have demonstrated methodology that is able to utilise relatively lower amounts of data to learn generalised models of highly complex and lengthy activities with multiple subjects and multiple object interactions. However, we note that a bottleneck in our proposed methodology is the reliance on linguistic semantic similarity of activity annotations presented as short language phrases. Since this is an open research problem in the NLP literature, we note that our method is highly sensitive to the accuracy of the language semantic similarity system used. Therefore, in retrospect, we would avoid the use of language semantic similarity unless a high performance is guaranteed, and simply use pre-set but amendable lists of activity labels to obtain annotations.

It is our hope that the research presented in this thesis will advance research in activity recognition towards tackling methods for representing long and complex activities and move away from traditional activity recognition aimed for simple activities. Furthermore, we demonstrated that our hierarchical representation of activities is explainable and human interpretable. We believe that, as deep learned discriminative model gain popularity, the requirement for explainable models is becoming necessary. Therefore, solutions presented in this research are of importance and would contribute and benefit the community.

Bibliography

- [CAD, 2013] (2013). Cornell activity dataset: Cad120, cad60. <http://pr.cs.cornell.edu/humanactivities/>. Accessed: 2014-06-30.
- [LAD, 2015] (2015). Leeds activity dataset. <http://dartportal.leeds.ac.uk/dataset/lad>. Accessed: 2015-06-30.
- [Aggarwal and Cai, 1997] Aggarwal, J. K. and Cai, Q. (1997). Human motion analysis: A review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102. IEEE.
- [Aggarwal and Cai, 1999] Aggarwal, J. K. and Cai, Q. (1999). Human motion analysis: A review. *Computer vision and image understanding*, 73(3):428–440.
- [Aggarwal and Ryoo, 2011] Aggarwal, J. K. and Ryoo, M. S. (2011). Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16.
- [Aho et al., 1972] Aho, A. V., Garey, M. R., and Ullman, J. D. (1972). The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137.
- [Ahuja and Todorovic, 2008] Ahuja, N. and Todorovic, S. (2008). Connected segmentation tree: a joint representation of region layout and hierarchy. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- [Aksoy et al., 2017] Aksoy, E. E., Orhan, A., and Wörgötter, F. (2017). Semantic decomposition and recognition of long and complex manipulation action sequences. *International Journal of Computer Vision*, 122(1):84–115.

- [Al-Omari et al., 2017] Al-Omari, M., Duckworth, P., Hogg, D. C., and Cohn, A. G. (2017). Natural language acquisition and grounding for embodied robotic systems. In *AAAI*, pages 4349–4356.
- [Albanese et al., 2008] Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V. S., Turaga, P., and Udrea, O. (2008). A constrained probabilistic petri net framework for human activity detection in video. *IEEE Transactions on Multimedia*, 10(6):982–996.
- [Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- [Amorapanth et al., 2010] Amorapanth, P. X., Widick, P., and Chatterjee, A. (2010). The neural basis for spatial relations. *Journal of Cognitive Neuroscience*, 22(8):1739–1753.
- [Arkhangelski and Fedorchuk, 1990] Arkhangelski, A. and Fedorchuk, V. V. (1990). The basic concepts and constructions of general topology. In *General Topology I*, pages 1–90. Springer.
- [Behera et al., 2012] Behera, A., Hogg, D. C., and Cohn, A. G. (2012). Egocentric activity monitoring and recovery. In *Asian Conference on Computer Vision*, pages 519–532. Springer.
- [Black et al., 1993] Black, E., Jelinek, F., Lafferty, J., Magerman, D. M., Mercer, R., and Roukos, S. (1993). Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 31–37. Association for Computational Linguistics.
- [Bleser et al., 2015] Bleser, G., Damen, D., Behera, A., Hendeby, G., Mura, K., Miezal, M., Gee, A., Petersen, N., Maçães, G., Domingues, H., et al. (2015). Cognitive learning, monitoring and assistance of industrial workflows using egocentric sensor networks. *PloS one*, 10(6):e0127769.
- [Bobick, 1997] Bobick, A. F. (1997). Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 352(1358):1257–1265.

- [Bobick and Davis, 2001] Bobick, A. F. and Davis, J. W. (2001). The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267.
- [Bogaert, 2008] Bogaert, P. (2008). *A qualitative calculus for moving point objects constrained by networks*. PhD thesis, Ghent University.
- [Brickell et al., 2008] Brickell, J., Dhillon, I. S., Sra, S., and Tropp, J. A. (2008). The metric nearness problem. *SIAM Journal on Matrix Analysis and Applications*, 30(1):375–396.
- [Broad, 1938] Broad, C. D. (1938). Examination of McTaggart’s philosophy, vol. ii.
- [Cha, 2007] Cha, S.-H. (2007). Comprehensive survey on distance/similarity measures between probability density functions. *City*, 1(2):1.
- [Chandrashekar and Sahin, 2014] Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Comput. Electr. Eng.*, 40(1):16–28.
- [Charniak, 1996] Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1031–1036.
- [Chen et al., 2015] Chen, J., Cohn, A. G., Liu, D., Wang, S., Ouyang, J., and Yu, Q. (2015). A survey of qualitative spatial representations. *The Knowledge Engineering Review*, 30(01):106–136.
- [Chen et al., 2012] Chen, L., Hoey, J., Nugent, C. D., Cook, D. J., and Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):790–808.
- [Choudhury et al., 2008] Choudhury, T., Consolvo, S., Harrison, B., Hightower, J., LaMarca, A., LeGrand, L., Rahimi, A., Rea, A., Bordello, G., Hemingway, B., et al. (2008). The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2).
- [Chowdhury, 2003] Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

- [Clementini et al., 1997] Clementini, E., Di Felice, P., and Hernández, D. (1997). Qualitative representation of positional information. *Artificial intelligence*, 95(2):317–356.
- [Cohn et al., 1997] Cohn, A. G., Bennett, B., Gooday, J., and Gotts, N. M. (1997). Representing and reasoning with qualitative spatial relations about regions. In *Spatial and temporal reasoning*, pages 97–134. Springer.
- [Cohn and Hazarika, 2001] Cohn, A. G. and Hazarika, S. M. (2001). Qualitative spatial representation and reasoning: An overview. *Fundamenta informaticae*, 46(1-2):1–29.
- [Cohn et al., 2002] Cohn, A. G., Magee, D. R., Galata, A., Hogg, D. C., and Hazarika, S. M. (2002). Towards an architecture for cognitive vision using qualitative spatio-temporal representations and abduction. In *International Conference on Spatial Cognition*, pages 232–248. Springer.
- [Cohn et al., 2012] Cohn, A. G., Renz, J., Sridhar, M., et al. (2012). Thinking inside the box: a comprehensive spatial representation for video analysis. In *KR*.
- [Cover and Thomas, 2012] Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.
- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.
- [de Ridder et al.,] de Ridder, H. N. et al. Information system on graph classes and their inclusions (isgci). <http://www.graphclasses.org>.
- [de Souza and de A.T. de Carvalho, 2004] de Souza, R. M. and de A.T. de Carvalho, F. (2004). Clustering of interval data based on cityblock distances. *Pattern Recognition Letters*, 25(3):353 – 365.
- [DeCarlo, 1997] DeCarlo, L. T. (1997). On the meaning and use of kurtosis. *Psychological methods*, 2(3):292.

- [Delafontaine et al., 2011] Delafontaine, M., Cohn, A. G., and Van de Weghe, N. (2011). Implementing a qualitative calculus to analyse moving point objects. *Expert Systems with Applications*, 38(5):5187–5196.
- [Dick and Ceriel, 1990] Dick, G. and Ceriel, H. (1990). Parsing techniques, a practical guide. Technical report, Technical Report, Tech. Rep.
- [Dollár et al., 2005] Dollár, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72. IEEE.
- [Downs and Stea, 1974] Downs, R. M. and Stea, D. (1974). *Image and environment: Cognitive mapping and spatial behavior*. Transaction Publishers.
- [Dubba et al., 2011] Dubba, K., Bhatt, M., Dylla, F., Hogg, D. C., and Cohn, A. G. (2011). Interleaved inductive-abductive reasoning for learning complex event models. In *International Conference on Inductive Logic Programming*, pages 113–129. Springer.
- [Dubba et al., 2015] Dubba, K. S., Cohn, A. G., Hogg, D. C., Bhatt, M., and Dylla, F. (2015). Learning relational event models from video. *Journal of Artificial Intelligence Research*, 53:41–90.
- [Dubba et al., 2010] Dubba, K. S. R., Cohn, A. G., and Hogg, D. C. (2010). Event model learning from complex videos using ilp. In *ECAI*, volume 215, pages 93–98.
- [Duckworth, 2017] Duckworth, P. (2017). *Unsupervised Human Activity Analysis for Intelligent Mobile Robots*. PhD thesis, University of Leeds.
- [Duckworth et al., 2017] Duckworth, P., Al-Omari, M., Charles, J., Hogg, D. C., and Cohn, A. G. (2017). Latent dirichlet allocation for unsupervised activity analysis on an autonomous mobile robot. In *AAAI*, pages 3819–3826.
- [Duckworth et al., 2016a] Duckworth, P., Gatsoulis, Y., Jovan, F., Hawes, N., Hogg, D. C., and Cohn, A. G. (2016a). Unsupervised learning of qualitative motion behaviours by a mobile

- robot. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1043–1051, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- [Duckworth et al., 2016b] Duckworth, P., Gatsoulis, Y., Jovan, F., Hawes, N., Hogg, D. C., and Cohn, A. G. (2016b). Unsupervised learning of qualitative motion behaviours by a mobile robot. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1043–1051. International Foundation for Autonomous Agents and Multiagent Systems.
- [Earley, 1970] Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.
- [Egenhofer and Franzosa, 1991] Egenhofer, M. J. and Franzosa, R. D. (1991). Point-set topological spatial relations. *International Journal of Geographical Information System*, 5(2):161–174.
- [Fernyhough et al., 1998] Fernyhough, J., Cohn, A. G., and Hogg, D. C. (1998). Building qualitative event models automatically from visual input. In *Computer Vision, 1998. Sixth International Conference on*, pages 350–355. IEEE.
- [Fidler and Leonardis, 2007] Fidler, S. and Leonardis, A. (2007). Towards scalable representations of object categories: Learning a hierarchy of parts. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- [Fleuret and Geman, 2001] Fleuret, F. and Geman, D. (2001). Coarse-to-fine face detection. *International Journal of computer vision*, 41(1-2):85–107.
- [Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *science*, 315(5814):972–976.
- [Fukushima, 1988] Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.
- [Galata et al., 2002] Galata, A., Cohn, A., Magee, D., and Hogg, D. (2002). Modeling interaction using learnt qualitative spatio-temporal relations and variable length markov models. In

- Proceedings of the 15th European Conference on Artificial Intelligence*, pages 741–745. IOS Press.
- [Gerevini and Renz, 2002] Gerevini, A. and Renz, J. (2002). Combining topological and size information for spatial reasoning. *Artificial Intelligence*, 137(1-2):1–42.
- [Golumbic, 1978] Golumbic, M. C. (1978). Trivially perfect graphs. *Discrete Mathematics*, 24(1):105–107.
- [Gowda and Ravi, 1995] Gowda, K. and Ravi, T. (1995). Agglomerative clustering of symbolic objects using the concepts of both similarity and dissimilarity. *Pattern Recognition Letters*, 16(6):647 – 652.
- [Gowda and Krishna, 1978] Gowda, K. C. and Krishna, G. (1978). Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition*, 10(2):105–112.
- [Gunning, 2017] Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA)*, nd Web.
- [Han et al., 2013] Han, L., Kashyap, A., Finin, T., Mayfield, J., and Weese, J. (2013). Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52.
- [Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer.
- [Hawes et al., 2017] Hawes, N., Burbridge, C., Jovan, F., Kunze, L., Lacerda, B., Mudrová, L., Young, J., Wyatt, J., Hebesberger, D., Kortner, T., et al. (2017). The STRAND project: Long-term autonomy in everyday environments. *IEEE Robotics & Automation Magazine*, 24(3):146–156.
- [Heer and Bostock, 2010] Heer, J. and Bostock, M. (2010). Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM.

- [Herath et al., 2017] Herath, S., Harandi, M., and Porikli, F. (2017). Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21.
- [Hongeng and Nevatia, 2001] Hongeng, S. and Nevatia, R. (2001). Multi-agent event recognition. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 84–91. IEEE.
- [Hopcroft et al., 2001] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):228–302.
- [Hu et al., 2014] Hu, N., Englebienne, G., Lou, Z., and Kröse, B. (2014). Learning latent structure for activity recognition. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1048–1053. IEEE.
- [Ibrahim et al., 2016] Ibrahim, M. S., Muralidharan, S., Deng, Z., Vahdat, A., and Mori, G. (2016). A hierarchical deep temporal model for group activity recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 1971–1980. IEEE.
- [Iserles, 1989] Iserles, A. (1989). Numerical recipes in c: The art of scientific computing.
- [Ivanov and Bobick, 2000] Ivanov, Y. A. and Bobick, A. F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872.
- [Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142.
- [Joo and Chellappa, 2006] Joo, S.-W. and Chellappa, R. (2006). Attribute grammar-based event recognition and anomaly detection. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 107–107. IEEE.
- [Kaisser and Lowe, 2008] Kaisser, M. and Lowe, J. (2008). Creating a research collection of question answer sentence pairs with Amazon’s Mechanical Turk. In *LREC*.
- [Ke et al., 2007] Ke, Y., Sukthankar, R., and Hebert, M. (2007). Spatio-temporal shape and flow correlation for action recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

- [Kim et al., 2010] Kim, E., Helal, S., and Cook, D. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1).
- [Kitani et al., 2005] Kitani, K. M., Sato, Y., and Sugimoto, A. (2005). Deleted interpolation using a hierarchical bayesian grammar network for recognizing human activity. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 239–246. IEEE.
- [Koppula and Saxena, 2013] Koppula, H. and Saxena, A. (2013). Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *International Conference on Machine Learning*, pages 792–800.
- [Koppula et al., 2013] Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, 32(8):951–970.
- [Laptev, 2005] Laptev, I. (2005). On space-time interest points. *International journal of computer vision*, 64(2-3):107–123.
- [Laptev et al., 2008] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- [Lara and Labrador, 2013] Lara, O. D. and Labrador, M. A. (2013). A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209.
- [Laube et al., 2005] Laube, P., Imfeld, S., and Weibel, R. (2005). Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668.
- [Lavee et al., 2009] Lavee, G., Rivlin, E., and Rudzsky, M. (2009). Understanding video events: a survey of methods for automatic interpretation of semantic occurrences in video. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(5):489–504.

- [Leon, 1980] Leon, S. J. (1980). *Linear algebra with applications*. Macmillan New York.
- [Liddy et al., 1999] Liddy, E. D., Paik, W., McKenna, M. E., and Li, M. (1999). Natural language information retrieval system and method. US Patent 5,963,940.
- [Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.
- [Marcus et al., 1993] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The Penn treebank. *Computational linguistics*, 19(2):313–330.
- [McCallum et al., 1998] McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI.
- [Mitchell, 1994] Mitchell, D. C. (1994). Sentence parsing. *Handbook of psycholinguistics*, pages 375–409.
- [Mohnhaupt and Neumann, 1991] Mohnhaupt, M. and Neumann, B. (1991). Understanding object motion: Recognition, learning and spatiotemporal reasoning. *Robotics and Autonomous Systems*, 8(1-2):65–91.
- [Molina et al., 2002] Molina, L. C., Belanche, L., and Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 306–313.
- [Moore and Essa, 2002] Moore, D. and Essa, I. (2002). Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI*, pages 770–776.
- [Munkres, 1957] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.
- [Natarajan and Nevatia, 2007] Natarajan, P. and Nevatia, R. (2007). Coupled hidden semi-Markov models for activity recognition. In *Motion and Video Computing, 2007. WMVC'07. IEEE Workshop on*, pages 10–10. IEEE.

- [Ng et al., 2002] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.
- [Nguyen et al., 2005] Nguyen, N. T., Phung, D. Q., Venkatesh, S., and Bui, H. (2005). Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 955–960. IEEE.
- [Nguyen-Dinh et al., 2013] Nguyen-Dinh, L.-V., Waldburger, C., Roggen, D., and Tröster, G. (2013). Tagging human activities in video by crowdsourcing. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, pages 263–270. ACM.
- [Oliver et al., 2002] Oliver, N., Horvitz, E., and Garg, A. (2002). Layered representations for human activity recognition. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, page 3. IEEE Computer Society.
- [Oliver et al., 2000] Oliver, N. M., Rosario, B., and Pentland, A. P. (2000). A Bayesian computer vision system for modeling human interactions. *IEEE transactions on pattern analysis and machine intelligence*, 22(8):831–843.
- [Ordóñez and Roggen, 2016] Ordóñez, F. J. and Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115.
- [Park and Jun, 2009] Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341.
- [Park and Aggarwal, 2004] Park, S. and Aggarwal, J. K. (2004). A hierarchical Bayesian network for event recognition of human actions and interactions. *Multimedia systems*, 10(2):164–179.
- [Pei et al., 2013] Pei, M., Si, Z., Yao, B. Z., and Zhu, S.-C. (2013). Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding*, 117(10):1369–1383.

- [Peng et al., 2005] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238.
- [Peng et al., 2016] Peng, X., Wang, L., Wang, X., and Qiao, Y. (2016). Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125.
- [Pereira and Schabes, 1992] Pereira, F. and Schabes, Y. (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics.
- [Piciarelli and Foresti, 2006] Piciarelli, C. and Foresti, G. L. (2006). On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27(15):1835–1842.
- [Piciarelli and Foresti, 2007] Piciarelli, C. and Foresti, G. L. (2007). Anomalous trajectory detection using support vector machines. In *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 153–158. IEEE.
- [Polana and Nelson, 1994] Polana, R. and Nelson, R. (1994). Low level recognition of human motion (or how to get your man without finding his body parts). In *Motion of Non-Rigid and Articulated Objects, 1994., Proceedings of the 1994 IEEE Workshop on*, pages 77–82. IEEE.
- [Poppe, 2010] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990.
- [Porikli, 2004] Porikli, F. (2004). Learning object trajectory patterns by spectral clustering. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 2, pages 1171–1174. IEEE.
- [Prescher et al., 2006] Prescher, D., Scha, R., Simaan, K., and Zollmann, A. (2006). What are treebank grammars? In *Proceedings of the Belgian-Netherlands Artificial Intelligence Conference (BNAIC). Namur*.

- [Pujari et al., 1999] Pujari, A. K., Kumari, G. V., and Sattar, A. (1999). Indu: An interval & duration network. In *Australasian Joint Conference on Artificial Intelligence*, pages 291–303. Springer.
- [Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. *KR*, 92:165–176.
- [Rashtchian et al., 2010] Rashtchian, C., Young, P., Hodosh, M., and Hockenmaier, J. (2010). Collecting image annotations using amazon’s mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*, pages 139–147. Association for Computational Linguistics.
- [Ravi et al., 2016] Ravi, D., Wong, C., Lo, B., and Yang, G.-Z. (2016). Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *Wearable and Implantable Body Sensor Networks (BSN), 2016 IEEE 13th International Conference on*, pages 71–76. IEEE.
- [Rodriguez et al., 2008] Rodriguez, M. D., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- [Roggen et al., 2010] Roggen, D., Calatroni, A., Rossi, M., Holleczeck, T., Förster, K., Tröster, G., Lukowicz, P., Bannach, D., Pirkel, G., Ferscha, A., et al. (2010). Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 233–240. IEEE.
- [Rosenberg and Hirschberg, 2007] Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*.
- [Rybok et al., 2014] Rybok, L., Schauerte, B., Al-Halah, Z., and Stiefelhagen, R. (2014). important stuff, everywhere! activity recognition with salient proto-objects as context. In *Ap-*

- plications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 646–651. IEEE.
- [Ryoo and Aggarwal, 2006] Ryoo, M. S. and Aggarwal, J. K. (2006). Recognition of composite human activities through context-free grammar based representation. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1709–1718. IEEE.
- [Salomaa, 1969] Salomaa, A. (1969). Probabilistic and weighted grammars. *Information and Control*, 15(6):529–544.
- [San et al., 2017] San, P. P., Kakar, P., Li, X.-L., Krishnaswamy, S., Yang, J.-B., and Nguyen, M. N. (2017). Deep learning for human activity recognition. In *Big Data Analytics for Sensor-Network Collected Intelligence*, pages 186–204. Elsevier.
- [Savic, 2002] Savic, D. (2002). Single-objective vs. multiobjective optimisation for integrated decision support, in: Integrated assessment and decision. In *Proceedings of the First Biennial Meeting of the International Environmental Modelling and Software Society*, pages 7–12.
- [Schuldt et al., 2004] Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: a local SVM approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE.
- [Schütze et al., 2008] Schütze, H., Manning, C. D., and Raghavan, P. (2008). *Introduction to information retrieval*, volume 39. Cambridge University Press.
- [Seber and Lee, 2012] Seber, G. A. and Lee, A. J. (2012). *Linear regression analysis*, volume 329. John Wiley & Sons.
- [Selvaraju et al., 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., et al. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, pages 618–626.
- [Serre et al., 2005] Serre, T., Wolf, L., and Poggio, T. (2005). Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994–1000. IEEE.

- [Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *ICCV*, page 1470. IEEE.
- [Sorokin and Forsyth, 2008] Sorokin, A. and Forsyth, D. (2008). Utility data annotation with amazon mechanical turk. *Urbana*, 51(61):820.
- [Sra et al., 2005] Sra, S., Tropp, J., and Dhillon, I. S. (2005). Triangle fixing algorithms for the metric nearness problem. In *Advances in Neural Information Processing Systems*, pages 361–368.
- [Sridhar et al., 2010a] Sridhar, M., Cohn, A., and Hogg, D. (2010a). Relational graph mining for learning events from video.
- [Sridhar et al., 2008] Sridhar, M., Cohn, A. G., and Hogg, D. C. (2008). Learning functional object categories from a relational spatio-temporal representation. In *ECAI 2008: 18th European Conference on Artificial Intelligence (Frontiers in Artificial Intelligence and Applications)*, pages 606–610. IOS Press.
- [Sridhar et al., 2010b] Sridhar, M., Cohn, A. G., and Hogg, D. C. (2010b). Discovering an event taxonomy from video using qualitative spatio-temporal graphs. In *ECAI 2010-19th European Conference on Artificial Intelligence, Proceedings*, volume 215, pages 1103–1104. IOS Press.
- [Sridhar et al., 2011] Sridhar, M., Cohn, A. G., and Hogg, D. C. (2011). From video to RCC8: exploiting a distance based semantics to stabilise the interpretation of mereotopological relations. In *International Conference on Spatial Information Theory*, pages 110–125. Springer.
- [Starner and Pentland, 1997] Starner, T. and Pentland, A. (1997). Real-time American sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer.
- [Subramanya et al., 2006] Subramanya, A., Raj, A., Bilmes, J., and Fox, D. (2006). Hierarchical models for activity recognition. In *2006 IEEE Workshop on Multimedia Signal Processing*, pages 233–237. IEEE.

- [Tan et al., 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [Tayyub et al., 2017] Tayyub, J., Hawasly, M., Hogg, D. C., and Cohn, A. G. (2017). CLAD: A complex and long activities dataset with rich crowdsourced annotations. *CoRR*, abs/1709.03456.
- [Ting, 2010] Ting, K. M. (2010). *Precision and Recall*, pages 781–781. Springer US, Boston, MA.
- [Turaga et al., 2008] Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udreă, O. (2008). Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video technology*, 18(11):1473–1488.
- [Van de Weghe et al., 2006] Van de Weghe, N., Cohn, A., De Tre, G., and De Maeyer, P. (2006). A qualitative trajectory calculus as a basis for representing moving objects in geographical information systems. *Control and Cybernetics*, 35(1):97–119.
- [Van de Weghe et al., 2005] Van de Weghe, N., Kuijpers, B., Bogaert, P., and De Maeyer, P. (2005). A qualitative trajectory calculus and the composition of its relations. In *International Conference on GeoSpatial Semantics*, pages 60–76. Springer.
- [Vilain et al., 1990] Vilain, M., Kautz, H., and Van Beek, P. (1990). Constraint propagation algorithms for temporal reasoning: A revised report. In *Readings in qualitative reasoning about physical systems*, pages 373–381. Elsevier.
- [Vinh et al., 2010] Vinh, N. X., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854.
- [Vishwakarma and Agrawal, 2013] Vishwakarma, S. and Agrawal, A. (2013). A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer*, 29(10):983–1009.
- [Voosen, 2017] Voosen, P. (2017). How AI detectives are cracking open the black box of deep learning. *Science Magazine*.

- [Wang et al., 1998] Wang, H., Bell, D., and Murtagh, F. (1998). Relevance approach to feature subset selection. *Kluwer International Series in Engineering and Computer Science*, pages 85–100.
- [Wang et al., 2018] Wang, J., Chen, Y., Hao, S., Peng, X., and Hu, L. (2018). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*.
- [Weghe et al., 2004] Weghe, N. V. d., Cohn, A. G., and Maeyer, P. D. (2004). A qualitative representation of trajectory pairs. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 1101–1102. IOS Press.
- [Wei et al., 2016] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732.
- [Weisstein, 2002] Weisstein, E. W. (2002). Complete bipartite graph.
- [West et al., 2001] West, D. B. et al. (2001). *Introduction to graph theory*, volume 2. Prentice Hall Upper Saddle River.
- [Westfall, 2014] Westfall, P. H. (2014). Kurtosis as peakedness. *The American Statistician*, 68(3):191–195.
- [Wöflf and Westphal, 2009] Wöflf, S. and Westphal, M. (2009). On combinations of binary qualitative constraint calculi. In *IJCAI*, pages 967–973.
- [Yamato et al., 1992] Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 379–385. IEEE.
- [Yang et al., 2015] Yang, J., Nguyen, M. N., San, P. P., Li, X., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI*, pages 3995–4001.

- [Yang et al., 2012] Yang, X., Zhang, C., and Tian, Y. (2012). Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1057–1060. ACM.
- [Young and Hawes, 2015] Young, J. and Hawes, N. (2015). Learning by observation using qualitative spatial relations. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15*, pages 745–751, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- [Young et al., 2017] Young, J., Kunze, L., Basile, V., Cabrio, E., Hawes, N., and Caputo, B. (2017). Semantic web-mining and deep vision for lifelong object discovery. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2774–2779. IEEE.
- [Younger, 1967] Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3 . *Information and control*, 10(2):189–208.
- [Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.
- [Zelnik-Manor and Irani, 2006] Zelnik-Manor, L. and Irani, M. (2006). Statistical analysis of dynamic actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1530–1535.
- [Zhang et al., 2017] Zhang, Q., Cao, R., Shi, F., Wu, Y. N., and Zhu, S.-C. (2017). Interpreting cnn knowledge via an explanatory graph. *arXiv preprint arXiv:1708.01785*.
- [Zhang and Zhu, 2018] Zhang, Q.-s. and Zhu, S.-C. (2018). Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39.
- [Zhang et al., 2010] Zhang, Y., Jin, R., and Zhou, Z.-H. (2010). Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52.

- [Zhang et al., 2011] Zhang, Z., Tan, T., and Huang, K. (2011). An extended grammar system for learning and recognizing complex visual events. *IEEE transactions on pattern analysis and machine intelligence*, 33(2):240–255.
- [Zimmermann and Freksa, 1996] Zimmermann, K. and Freksa, C. (1996). Qualitative spatial reasoning using orientation, distance, and path knowledge. *Applied intelligence*, 6(1):49–58.