

Thesis
Representing and Recognizing Temporal Sequences

A Thesis
Presented to
The Academic Faculty

by

Yifan Shi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
December 2006

Thesis

Representing and Recognizing Temporal Sequences

Approved by:

Professor Aaron Bobick
College of Computing
Georgia Institute of Technology

Professor Mubarak Shah
Computer Science Department
University of Central Florida

Professor Irfan Essa
College of Computing
Georgia Institute of Technology

Professor Monique Thonnat
INRIA

Professor James Rehg
College of Computing
Georgia Institute of Technology

Date Approved: July 2006

TABLE OF CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
SUMMARY	ix
I INTRODUCTION	1
1.1 Motivation	1
1.2 Activity Decomposition and Recognition	2
1.3 Contribution	4
II BACKGROUND	6
2.1 Activity Characteristics	6
2.2 Existing Method Overview	7
2.3 Representing Motion Field	8
2.4 Finite State Machine	9
2.5 Hidden Markov Model	11
2.5.1 Hierarchical HMM	12
2.5.2 Coupled HMM	12
2.6 Bayesian Network	13
2.7 Dynamic Bayesian Network	15
2.8 Stochastic Context Free Grammar	17
2.9 Stochastic Petri Net	19
2.10 Symbolic Network Approach	20
2.10.1 PNF-Network	21
2.10.2 Frame-based Method	22
III REPRESENTING THE TEMPORAL SEQUENCES: PROPAGATION NET 23	
3.1 Philosophy	25
3.2 Conceptual Model	25
3.2.1 Representing Temporal Relations	26

3.2.2	Representing Logic Relation	28
3.2.3	Prescribing conceptual model	30
3.3	Computational Model	31
3.4	Complexity	36
IV	P-NET RECOGNITION BY LOCAL MAXIMAL SEARCH ALGORITHM	38
4.1	Forward-Backward Calculation	39
4.2	Iteration	40
4.3	Complexity	43
4.4	Experiment	44
V	P-NET RECOGNITION BY DISCRETE CONDENSATION	48
5.1	Particle Filter	48
5.2	Discrete State Space	49
5.3	Round-Robin Selection for D-Condensation	51
5.4	Classification of Noisy Input	53
5.5	Complexity	54
5.6	Experiment: Glucose Meter Calibration Task	55
5.7	Comparison with SCFG	58
VI	LEARNING	61
6.1	Network Parameter Learning	61
6.2	Detector Learning	63
6.2.1	Framework	63
6.2.2	Semi-supervised Learning	65
6.2.3	Contrast Boosting	67
6.3	Experiments	71
6.3.1	Classification	73
6.3.2	Anomaly Detection	74
6.3.3	Individual Event Detection	75
6.3.4	Unknown Topology	77

VII EXTENSION AND OPEN QUESTIONS	80
7.1 Simulation Machine	80
7.2 Hierarchical Definition	81
7.3 Open Questions	84
7.3.1 Structure Learning	85
7.3.2 Anomaly vs. Outlier	88
7.3.3 Interleaved Activities	90
VIII CONCLUSION	92
REFERENCES	95

LIST OF TABLES

Table 1	the deficiency of existing models	25
Table 2	Overall Evaluation	56
Table 3	Labelling individual node	57
Table 4	Overall Computation Performance	58
Table 5	Abnormity detection results on the glucose calibration dataset	76
Table 6	Result of detecting individual nodes in indoor activity data set and glucose activity data set	78
Table 7	Classification ability on the exercise data set	79

LIST OF FIGURES

Figure 1	Example FSM to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D. A dot marks the progress of an activity, where the symbol to the left of the dot represents what is happening or has happened, and the symbol to the right marks what is expected; a hat on a symbol denotes that the motion represented by the symbol has finished.	10
Figure 2	Example BNT to represent an activity of four elements:A/B/C/D, where A happens before B/C, and B/C happens before D	13
Figure 3	Example DBN to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D	16
Figure 4	Example SCFG to represent an activity of four elements:A/B/C/D, where A happens before B/C, and B/C happens before D	18
Figure 5	Example Petri Net to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D. The events are represented by transition. The current status is represented by the token.	19
Figure 6	13 temporal relations between two intervals	26
Figure 7	An example of temporal relations with dummy nodes	28
Figure 8	An example of logic relation	29
Figure 9	Conceptual diagram for a glucose meter calibration process	30
Figure 10	Comparison of a conceptual model and its corresponding computational model	35
Figure 11	The independence assumption in forward computation may lead to a fake maximal point in true joint distribution.	41
Figure 12	Local Maximal Search Algorithm (LMSA) for P-Net inference	42
Figure 13	Propagation Nets for “calling” and “reading”	44
Figure 14	Architecture of a P-Net based activity recognition system	45
Figure 15	Results on “reading” and “calling” P-Nets	46
Figure 16	Comparison of straight Bayes net detector vs. P-Net labelling for Left-PutbackBegin	47
Figure 17	Potential consequent states when node B/C are active in last time step in the P-Net of Figure.10	51
Figure 18	Discrete Condensation (D-Condensation) for P-Net inference	52

Figure 19	Conceptual diagram for a glucose meter calibration process	55
Figure 20	Comparison of PNet label with ground truth and lower level vision output.	58
Figure 21	The online labelling of test sequences by P-Net. [] shows the frame number in the sequence. The color of the a nodes in P-Net shows the marginal distribution that represents the P-Net belief on whether the corresponding motion is occurring. (Refer to Figure.9 for the motions represented by the nodes.)	60
Figure 22	Learning the duration and observation model from the positive training set	63
Figure 23	Example of Propagation Net with the boosted event detectors.	64
Figure 24	Semi-Supervised Learning Algorithm to train P-Net and its underlying detectors in unified EM iterations.	66
Figure 25	Example of P-Net with the boosted ensemble as event detectors	69
Figure 26	A generalized version of ADABOOST and corresponding α from [19]	70
Figure 27	Contrast Boosting with weight on each dimension	70
Figure 28	Vision tracked indoor activities: “reading” and “calling”.	72
Figure 29	Vision tracked glucose monitor calibration process.	73
Figure 30	Classification results on Indoor activity dataset, Classification Rate = (No.of.Correct-No.of.Wrong)/Total	75
Figure 31	As there is no obvious structure in the exercise data set, we select a topology that has two sub-streams to start the training.	77
Figure 32	P-Net based simulation machine	81
Figure 33	Hierarchical definition of P-Net	84

SUMMARY

Activity recognition falls in the general area of pattern recognition, but it resides mainly in the temporal domain which leads to very distinctive characteristics. We first point out those important points, then we provide an extensive survey over the existing tools including FSM, HMM, BNT, DBN, SCFG and Symbolic Network Approach (e.g. PNF-network). These tools are inefficient to meet many of the requirements of activity recognition, leading to this work to develop a new graphical model: Propagation Net (P-Net).

Many activities can be represented by a partially ordered set of temporal intervals, each of which corresponds to a primitive motion. Each interval has both temporal and logical constraints that control the duration of the interval and its relationship with other intervals. Recognizing such activity requires the processing of multiple, parallel streams of intervals. P-Nets are introduced to take advantage of such fundamental constraints that it provides an graphical conceptual model to describe the human knowledge and an efficient computational model to facilitate recognition and learning.

A P-Net associates a node with each interval whose stochastic triggering function depends upon the state of its parent nodes. Each node is also associated with an observation function that describes perceptual evidence. This evidence, generated by a lower level perceptual module, is a unique indicator of the elemental motion that constitutes the activity.

P-Nets define an exponentially large joint distribution that standard bayesian inference cannot handle. To execute an inference task on a P-Net, we devise two approximation algorithms to interpret a multi-dimensional observation sequence of evidence as a multi-stream propagation process through P-Net. First, inspired by the Viterbi algorithm, Local Maximal Search Algorithm (LMSA) is constructed with polynomial complexity. LMSA, however, is

limited in that it must see the whole process before it can provide an explanation. Second, to facilitate real-time analysis, we introduce a particle filter based framework to explore the conditional state space. By modifying the original Condensation algorithm to sample the discrete state space more efficiently, we obtain Discrete Condensation (D-Condensation) algorithm.

To construct a P-Net based activity recognition system, we need two parts: the P-Net and the corresponding detector set. Given the topology information and the detector library, P-Net parameters can be extracted easily from a relatively small number of positive examples. To construct the detector library, one either has to build each detector by hand or to hand label each frame to train the individual detector. Either way, the process is expert-intensive work. To avoid this tedious process, we introduce a semi-supervised learning framework to build a P-Net and the corresponding detectors together. Once the topology of P-Net is manually specified, the P-Net and its evidence detectors are initialized on a small number of fully annotated examples. They are then refined together in a unified EM iteration by using additional non-annotated positive examples. Within this framework, we use boosted stumps as node detectors. Due to the nature of P-Net, detectors for different nodes can fire simultaneously, where normal multi-class boosting algorithm cannot work. Instead, we introduce the *Contrast Boosting* algorithm that forces the detectors to be as different as possible but not necessary to be non-overlapping.

The classification and learning ability of P-Nets are verified on three data sets, which are obtained from vision as well as an alternative sensor platform: 1)vision tracked indoor activity data set; 2)vision tracked glucose monitor calibration data set; 3)sensor data set on simple weight-lifting exercise. Comparison with standard SCFG and HMM prove a P-Net based system is easier to construct and has a superior ability to classify complex human activity and detect anomaly.

To facilitate the use of P-Nets, we introduce a hierarchical definition into the P-Net construction process. A special node that references a sub-P-Net is used in the conceptual

model during the construction of a P-Net. When the conceptual model is compiled into the computational model, those special nodes are substituted by the sub P-Net so that normal P-Net inference can be performed.

In activity domain, the training sequences are usually limited. Even fewer are the sequences with annotation at individual steps. P-Net can be used as a simulation machine to solve this problem. As a proof of generalization ability and the application of P-Net, we will show that, given a P-Net with a generative observation model for each node, the P-Net inference algorithm can be augmented to generate an unlimited number of positive example sequences. These large scale samples can serve as the standard testing set for any novel method.

CHAPTER I

INTRODUCTION

1.1 Motivation

Automated systems are becoming increasingly involved in our daily lives. To correctly handle a situation—either help people finish a job in a friendly environment or to prevent people from error in a hostile situation—the system must know what is happening. Human activity is a very important factor in this process. Knowledge of what people are doing can enable a wide range of assisting technologies, smart appliances, and aware environments. There have been quite a few interesting applications in various research societies.

For example, “Deja Vu Displays” [65] is trying to monitor the cooking process in the AwareHome. It watches the target by cameras and sensors to understand each operation and then uses the context information to provide predictive as well as assisting hints for elderly people so as to avoid cooking errors. “Smart Kiosk Project” in HP is designed to provide information, sell products and entertain. It needs to track and understand people in its vicinity; Donald etc [53] described the application in tracking a person in an urban area with GPS and tried to figure out the target’s normal habit and daily pattern.

Security is another major application domain that requires an understanding of activity. Video surveillance is becoming standard in all sorts of public areas as well as well the private backyard. Having a person behind each camera is simply impossible. An automated filter system as well as abnormal situation detection system has generated enough market demand for companies like VisionIQ, which offers an automated system to perform drowning detection and Brickstream, which provides shopping customer identification.

All these applications have lead to a surge of research in human activity recognition. The core problem is to model and recognize a temporal process. Having an extra dimension

of time makes the problem interesting as well as challenging. On one hand, there is more information embedded in a sequence than the information in a snapshot. On the other hand, a sequence may have much richer structure and dexterity that makes the information appearing noisy and hard to understand.

The objectives of an activity recognition system can be summarized in 3 aspects, each of which contributes the evaluation of an activity representation and recognition model. They are, 1) Classification: Which activity took place in the observed sequence? 2) Annotation: When did various components of the activity occur? 3) Anomaly Detection: Is there anything unusual in the sequence?

1.2 Activity Decomposition and Recognition

Human activity is a complex temporal process. It usually has multiple agents and lasts a relatively long period of time. To represent and analyze it, it is natural to decompose the whole process into elements. Based on the specific time scales and agents involved, we suggest a two level decomposition, motion vs. activity, in which the low level motion is the building block for the high level activity.

Motion is the single movement of an autonomous object or the absence of movement and reflects the current status of a particular object or the current relationship of a few objects. Its existence is directly verifiable from the physical description of the object(s). A typical motion with movement is “punching”, the act of a human fist moving in straight-forward direction with a high speed; a typical motion without movement is “touching”, the physical status that a hand is in contact with another object. A motion can usually be observed from the lower level perception system. A single motion usually does not have a clear meaning until combined with other motions. It is not uncommon in activity literature for low level elements to be called events. Here in our decomposition, we use “motion” to emphasize the effect that the primitive element is usually not instantaneous but a interval with duration.

Activity is the cooperation of multiple autonomous objects in a relatively long period to achieve a complex result. Activity has two important aspects. One is the building blocks: motion; the other is the coordination between those motions. This coordination exists not only in the temporal domain but also in the logical domain. Just like the natural language, the vocabulary is usually limited, but the coordination of those words in a certain formality forms the description of the real world. How to represent these relations and leverage them to accomplish recognition is critical to activity recognition.

Action is a frequently used term in this domain. According to [4], action is a movement in a context with an intention. Here the concept of activity and action are similar in time scale, but activity emphasizes the internal structure. Through this difference, we situate the research in the complex and long lasting activity domain. Simple short activity like “greeting between two people” is not our focus.

The decomposition suggests a natural architecture to the activity recognition system: to build a detector for each individual motion and a controlling layer on top of the detectors to represent the temporal/logical relationship between motions and coordinate the overall behavior. The central problem in activity recognition is sequence modeling and recognition. Many models can be used (surveyed in the next chapter). They all conform to the above structure. The elements for a sequence may vary due to the granularity, but it is generally true that the sequence of raw feature vectors are organized into elements then the sequence of elements constitutes an activity. Our system conforms to the same architecture.

Video plays an important role in activity recognition especially in surveillance applications. It is fairly easy to set up several video cameras and utilize off-the-shelf algorithms to extract information from the video stream. In this thesis, most of the information is extracted from video domain, but there is no reason that we can not use other information. After all, the raw input to the system is a stream of feature vectors whose perceptual source is irrelevant.

1.3 Contribution

The primary goal of this thesis is an automatic human activity understanding system based upon vision and other sensor input. In the system, human activity is represented by a process of multiple streams. The core of our system is a new graphical model, Propagation Net (P-Net). The major breakthrough of our model over existing methods is its ability to handle multiple parallel streams in a unified and an efficient way. Though this model is designed for human activity, we believe it can be generalized to model any complex temporal sequences, and hence constitutes a new tool to the machine learning society to handle temporal sequences.

The contributions of this thesis are:

1) Propagation Net: a new graphical model is devised to represent the stochastic process of parallel streams. Its conceptual model uses intervals rather than instantaneous events as the elemental components; it represents temporal and logical constraints between motions uniformly as a conditional probability function; it has a hierarchical definition that facilitates code reusing; its conceptual model is compiled into an efficient computational model of DBN with duration models to carry out inference and learning.

2) Local Maximal Search Algorithm: an inference algorithm on the P-Net computational model. The joint distribution on DBN with duration model is too big to be handled directly. Rather, a marginalized probability distribution is estimated and independence between history nodes are assumed to carry on DBN updating. Later, the independence assumption is excluded by iterations.

3) Discrete Condensation: a particle filter based inference algorithm on the P-Net computational model. The joint distribution of DBN with the duration model is sampled by weighted particles. The coherence of history is automatically maintained by each particle. The discrete state space within DBN enables an efficient search on the next states and thus augments the Condensation algorithm to Discrete Condensation (D-Condensation).

4) Semi-supervised learning algorithm: the motion detector set is manually coded or

trained on the fully annotated examples. Such a process is tedious as well as expertise intensive. To avoid this problem, we devise a semi-supervised learning framework to build a P-Net and its motion detectors together. Once the topology of the P-Net computational model is manually specified, the P-Net and its evidence detectors are initialized on a small set of the fully annotated examples. They are then refined together in a unified EM iteration by using more non-annotated positive examples.

5) Contrast boosting algorithm: Within the semi-supervised learning framework, we use boosted stumps as the node detectors. Due to the nature of P-Net, the detectors for the different nodes can fire at the same time, a situation in which the normal multi-class boosting algorithm cannot work. To train for a set of detectors that are different but not exclusive, we devise the *Contrast Boosting* algorithm. It keeps a weight for each dimension of the observation vector per each detector. Weights for the same dimension of different detectors are normalized together so as to force the detectors focusing on the different dimensions. All detectors are trained together in the same boosting circle. Through *Contrast Boosting* algorithm, we can get a set of detectors for patterns that exist in multiple parallel streams.

CHAPTER II

BACKGROUND

2.1 Activity Characteristics

We have defined activity to be composed of motions. Each motion corresponds to a primitive temporal interval. To define an activity, we need to further describe the coordination between motions. Consider the example of reading a book. The primitive intervals as well as the relationships between the intervals look like, “First, [A] fetch the book; next, [B] look at the book while occasionally [C] flipping the pages; finally, [D] put down the book.” From this relatively trivial example, we can notice the need for the following relationships to represent the activity:

Partial temporal order: For example, motion A happens before B. The relationships of $A > B$, $B > D$ and $C > D$ constitute the natural partial ordering of components in the temporal domain. This partial ordering is the most salient temporal constraint.

Multiple parallel streams: interval B and interval C can happen in a variety of way, including B before C, C before B and B while C. This uncertainty originates from the lack of partial ordering between B and C, which present two locally parallel streams of relatively independent intervals. These parallel streams may not exist from the beginning to the end and they can become entwined together by merging and forking.

Logical constraints: D can start only after B “AND” C have finished. Each interval is triggered only by a particular logic combination of the parent intervals. In this case, it is “AND”. Those logic conditions control the progress of the activity.

Duration of the elements: The component of activity is motion. The building block is not

an event with instant time but has the temporal extent that becomes an interval rather than a checkpoint. The length of interval matters.

Non-adjacency: a blank period can exist between intervals such that the sequenced intervals may not meet each other.

To become valid, any general activity representation method should be able to incorporate these characteristics. In so doing, it presents three requirements: (1) the method should represent an activity by a vocabulary of elements, each of which corresponds to a motion with temporal duration; (2) the method should encode temporal and logical constraints between these elements. Furthermore, any perception system is subject to noisy observations. The representation method should be able to handle noise in a systematic way. This leads to the following requirement: (3) the method should have a probabilistic model when handling observations. The following survey of existing models will compare their abilities with respect to the above three requirements.

2.2 Existing Method Overview

Several techniques in standard pattern recognition can be applied to activity recognition. Ignoring the temporal dimension and using general pattern recognition techniques is a choice, but the core problem in activity representation and recognition is how to handle a sequence with variable length and meaningful internal structure. Such a sequence has a distinctive dimension of time that makes this type of pattern recognition different and interesting. Many specialized tools are available to model this process. In the following sections, we will examine the existing techniques in detail. Some earlier reviews can be found at [51, 46, 1]

The existing models are designed to emphasize different aspects of the activity. Some are suitable for motion, some for simple activity without much internal structure, others for complex activity. In our view, generally speaking, motion field methods are used in motion recognition; Hidden Markov Models (HMMs) are used on the motion and simple activity; Finite State Machine (FSM), Bayesian Network (BNT), Dynamic Bayesian Network

(DBN) and PNF-network/Allen's algebra are tools for all levels; Stochastic Context Free Grammar (SCFG) and Petri net are mainly for complex activity recognition.

2.3 Representing Motion Field

Recovering the motion field is a fundamental problem in computer vision, but it is a separate issue from this paper. We are interested in how to model a motion field once it is generated.

Davis & Bobick introduce Motion-energy image (MEI) and Motion-history image (MHI) in [15] to represent motion field. Once MEI and MHI images are generated, seven Hu moments are used as filter on these images to extract the feature vector. In this way, the evolution information is changed to static images by MEI+MHI, then further compressed by Hu moments. To recognize an input motion, a Mahalanobis distance is calculated between the moment description of the input and each of the known actions. The key idea here is to map a pattern in a process to a pattern in static image then later on to a point in static feature space, so that the process recognition becomes a static status classification and standard pattern recognition techniques can be applied [59]. There are other methods to change the motion process to static point. For example, in [61], aided with tracking, the motion sequence is segmented. Each segment is matched with a template pose. The trajectory of such index becomes a feature vector.

Efros & Malik [18] pushed the boundary of motion detection into small videos of poor quality. Their motion descriptor separates the positive and negative factors in the motion field. Such separation facilitates computing the normalized correlation matrix of two streams of motion fields. Therefore the patterns in the motion field are transformed into the patterns of correlation matrix. This technique requires full-scale frame-by-frame correlation resulting in computational complexity only applicable to videos of small image size.

Obviously, the above techniques are most suitable in a simple setup with single object,

whereas multi-objects problems are solved by segmentation in preprocessing. Their ability is limited, as they only indirectly use the rich structure in sequence.

2.4 Finite State Machine

A finite state machine (FSM) is composed of nodes with concrete meaning and probabilistic directed links to describe the topology. Usually the topology is specified by experts while transition probabilities are obtained from the training set [32, 43, 71].

FSMs have the virtue of describing a single stream process in an easily comprehensible manner. Each node has a meaning in the high level description which has concrete support from a segment of the lower level observation. The gap between actual observation features and comprehensible meaning is usually merged by an ad-hoc method. One choice [43] is to filter the observation with a fixed length window, then stack the sequence points within the window together to become a high dimension feature vector and generate the high level concept as the standard classification problem. Usually, a simple threshold based module is enough to generate concrete meaning such as “the distance between A and B is decreasing”.

Another choice [32] is to build a naive Bayesian tree for each node. The leaf nodes represents the probability density function for an individual observation, and the root node of the Bayesian tree corresponds to the node in FSM. Therefore, standard graphical model inference procedure, i.e., junction tree algorithm, can be invoked on each tree. The posterior probability of the root node will serve as the input for node.

Once the evidence is accumulated for the nodes of FSM, the iterative updating will be invoked. The nodes in FSM with the highest probability at each time step will mark an explanation path for the observation. The standard FSM is deterministic and the inference is linear. However, to handle noisy observations, FSM is usually augmented to become stochastic FSM. The Viterbi algorithm is a standard inference algorithm on a stochastic FSM.

The stochastic FSM requires that the posterior probabilities of all nodes at any time

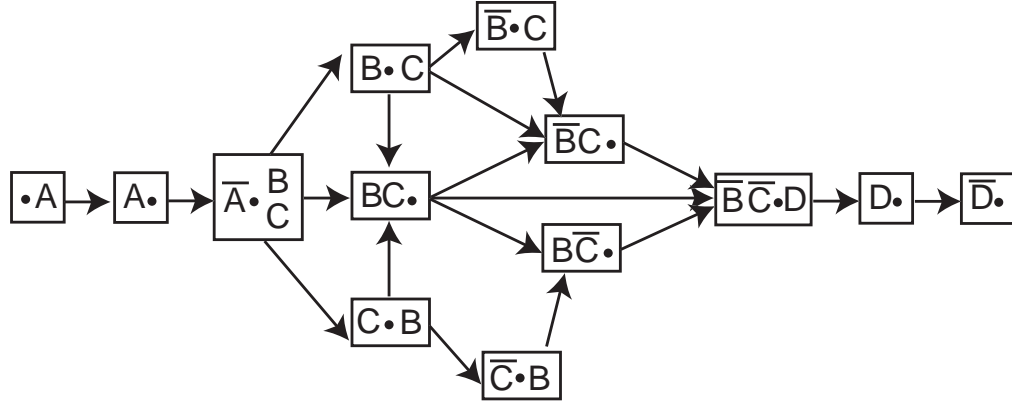


Figure 1: Example FSM to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D. A dot marks the progress of an activity, where the symbol to the left of the dot represents what is happening or has happened, and the symbol to the right marks what is expected; a hat on a symbol denotes that the motion represented by the symbol has finished.

step sum to one, and thus represents a major assumption that only one node is active at any particular time step. This condition is called “single stream assumption”, which also exists in many existing methods including HMM, SCFG. It is exactly the opposite of the second of the activity characteristics we described earlier. An activity of multiple streams can be modeled as a single stream by the power set of the possible elements, thereby leading to a magnitude increase in the complexity of FSM. Only when the activity is simple and short is such conversion bearable. An example is shown in Figure.1, where the activity has only four components, but the corresponding FSM needs 13 nodes.

Efforts to address this inability have emerged. Multi-threaded events are introduced by [33]. On top of individual FSMs, a temporal constraint layer derived from Allen’s algebra is added. Under the layer, each FSM represents a single thread event and produces all possible explanation candidates. The temporal model then searches for the best candidate to maximize the total probability. In this method, the interaction between FSMs is unclear and has yet to be formalized, which holds back this approach from complex scenarios. Furthermore, when activity becomes complex, each single stream component in the approach sometimes only corresponds to a single node. Without the constraints between the streams, the combined space of candidates produced by the lower level FSM is so huge that the search within the high level temporal constraint is impractical. We believe those high level

temporal constraints should be used to constrain the inference in lower level FSM, and we shall demonstrate how to do it in an efficient way in our work.

2.5 Hidden Markov Model

Hidden Markov Models (HMM) are a well known and used framework to model and classify dynamic behaviors. It offers automatic dynamic time warping, an efficient inference algorithm (Viterbi), a training algorithm, and clear Bayesian semantics.

Theoretically, an HMM is a probabilistic finite-state machine. Its major difference with a normal FSM is how it is constructed. Usually, FSM is designed by first having the topology and the meaning of nodes, followed by the individual training of the node for the observation model. HMM usually starts with no definite meaning of the nodes and a loosely defined topology. The meaning of nodes can be extracted after training. However, Bregler [7] uses such conceptual HMM to select different dynamics in motion, as such blurs the boundary between HMM and FSM.

Starner & Pentland used HMMs to recognize the videos of fairly complex American Sign Language recognition by the tracked hands. and the result reached 92% without grammar [63]. Wilson & Bobick also reported similar results on gesture recognition [69]. They bundled a family of HMM together with a parameter tying then trained the whole family together with EM iteration, leading to a model of better generalization ability.

“Style Machine” provides an interesting perspective on HMM [6]. It has a generic HMM model and a set of specific HMMs. Training keeps the generic HMM simple and pushes specific HMM towards details. PCA on trained HMM parameter space illustrates a vague concept of *style*. It provides a very interesting and unique way to model human perception.

2.5.1 Hierarchical HMM

Hierarchical HMM (HHMM) is a derivative of HMM. It has two kinds of states: “production state” that can only emit a single observation symbol and “internal state” that corresponds to a sub-level Hierarchical HMM which may eventually emit a string of observations [21]. The topology of HHMM is always purely hand-designed. The parameters of HHMM can be trained from the Baum-Welch algorithm. The original version of HHMM inference is based on direct extension of the Viterbi algorithm which has a complexity of $O(NT^3)$. [52] suggests a much faster algorithm of $O(T)$ complexity by converting a HHMM into a specialized DBN.

When used in the activity recognition, Hierarchical HMM is convenient to represent human concepts. By constructing elements of activities and iteratively stacking them together, it is quite easy to start with simple behavior and reach complex activity. Another aspect worth mentioning is that in the activity recognition domain the training set is usually small. With hierarchical architecture, the parameter binding enhances the training robustness. However, just like generic HMM, Hierarchical HMM is still targeting a single stream.

2.5.2 Coupled HMM

Coupled HMM (CHMM) represents the initial effort to model causal relation in parallel sequences [5]. It has a prescribed topology with a preset number of parallel HMMs. Those HMMs correspond to high level knowledge about the actual internal process that CHMM tries to model. To avoid the factorial state space problem, the correlation between HMMs is simplified by N-head dynamic programming.

Coupled HMM and its derivatives are oversimplified model of human activities. Parallel streams are assumed to be separable in CHMM, which is an assumption that rarely holds. For example, during cooking, preparing vegetable is parallel to preparing beef, here we need two HMMs to be coupled, but when they are stewed together, there is only one process which only needs one HMM. Such branching and merging will form a complex net

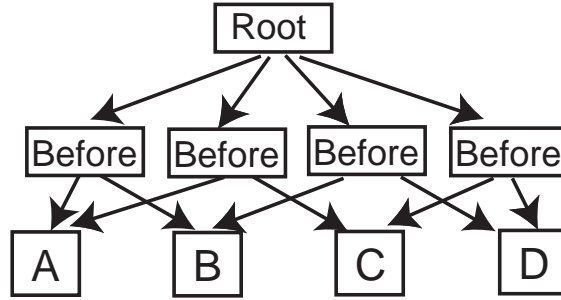


Figure 2: Example BNT to represent an activity of four elements:A/B/C/D, where A happens before B/C, and B/C happens before D

topologically different from CHMM. There is no way to change a net into multiple parallel threads to fit for CHMM. This is one of the inspirations that lead to the thesis.

2.6 Bayesian Network

Bayesian Network (BNT) is a well defined probabilistic reasoning tool. Its nodes have actual meaning and its links are derived from causal relationships. It is a very powerful and direct tool to describe the real world. Furthermore, inference on singly connected BNTs can be carried out by junction tree algorithm in polynomial time, which earns its numerous applications in AI society.

However, BNT does not have an evolving concept. It is mostly suitable to static scenarios. To extend it into temporal sequence, two approaches are suggested:

Intille & Bobick suggested using specialized nodes to incorporate temporal concept [35]. In their work to recognize football play, specialized temporal nodes for “A during B”, “A before B” and “A around B” are built with node A and node B acting as children. These temporal nodes will take in the timestamp on a child node and generate a pseudo posterior probability based on how well they match the prescribed temporal constraint. Then, A, B and their temporal relation node all serve as children for a higher level in the Bayesian tree. In this way, not only is the evidence of A and B passed on to root, but the temporal relation of A and B is also taken into account. An example is shown in Figure.2. This kind of specialized node can also be used to check the logic relation between nodes.

Another approach is to incorporate the time concept in leaf nodes [8]. Those leaf nodes become positive only when they are within a particular time interval. For example, node “going home after work” not only checks the direction of driving but also checks whether the current time is after office hour.

One can easily see that this method can represent any temporal relation with equal cost, a feature that leads to its strong representation power. But the power does not come at no cost. On one side, BNT’s capacity makes it easy for human expert to design the topology of activity representation; on the other side, the parameter learning, let alone the structure learning, on such huge space is formidable.

BNT moreover does not have a concept of process. As we explained, the process is modeled by a few hand-picked moments. But in fact, any pair of components in a natural process may have temporal relations that can facilitate recognition. Over all the possible relations, this method only uses a few salient points. Given enough prior knowledge and careful as well as painful effort to make the right choice, BNT works well in a controlled setup. However, the real world problem is usually more complex than the few handpicked checking nodes, which project great doubt on BNT’s generalization ability. If one needs to learn a BNT, finding those checkpoints is critical to success. A few BNT learning algorithms exist, but finding the temporal relationship between hidden nodes is yet to be discovered.

Another problem with the tree shaped BNT is its inability to produce a full scale explanation about how the sequence evolves. In other words, it cannot label the sequence except for the few salient checkpoints. Without such a full scale ability, developing an algorithm to discover the structure within activity is very hard. And BNT is unsuitable for online reasoning

Some effort to integrate a process concept in BNT has emerged. TimeNet [39] is one of such examples. TimeNet is truly a network rather than a tree. TimeNet assigned a node

for each event per any interesting time interval. (Notice a magnitude more nodes potentially exist for a single event.) The meaning of any node is no longer just that something happened, but something happened at certain time interval. The temporal relation is fully buried under the fact that each event has a special time. For example, “event A before B” is represented in the style of “event A has time interval $[T1, T2]$ ” and “B has time interval $[T2, T3]$ ” while the actual time relation is ignored. At each time step, there are a list of effective nodes that consist the BNT. Standard inference algorithm is used on this BNT to retrieve the posterior probability for each hidden node. No single root node to represent the activity, but the activation of nodes at each time step marks an explanation of the observation. The price for enforcing a hard timestamp on nodes is usually unbearable. To define an activity with multiple evolving options and variable durations, TimeNet needs many more nodes and the inference becomes intractable.

2.7 Dynamic Bayesian Network

Dynamic Bayesian Network (DBN) is derived from BNT. DBN models a stationary Markov process. At each time step, this process has a set of hidden nodes and evidence nodes. One way to look at DBN is to unroll DBN to the maximal window size by duplicating the set of hidden variable and evidence nodes for each time step. Variables within one time step are connected by intra-link; between the time step, hidden variables are connected by inter-link. DBN is the representative slices of this large BNT. Direct junction tree inference on this unrolled BNT is neither space not time efficient. In a stationary markov process, only variables in the previous time step influence the current variables. This property ensures the hidden variable in previous time step D-separate the future from the past. Therefore, the inference on the unrolled BNT can be completed iteratively with only two slices of variables: one slice representing the beliefs at the previous time step, the other representing beliefs at the current time step. These two slices are inter-linked together by the conditional probability functions to form a BNT. At each time step, the standard junction tree algorithm

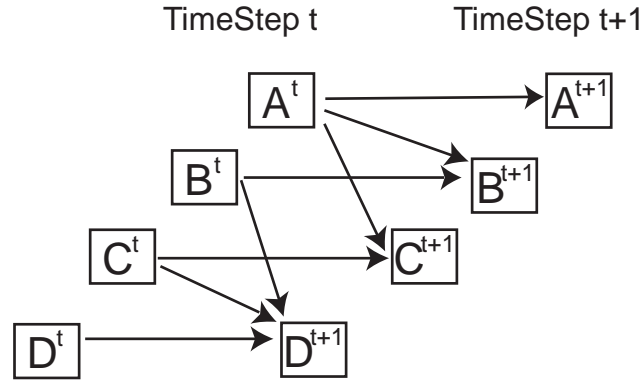


Figure 3: Example DBN to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D

on this BNT will produce posterior probability on the next time slice of nodes, given new evidence on the next time frame; then the current time slice is abandoned and the next time slice becomes the current time slice. Iteration continues until the last time step. Detailed derivation of DBN can be found in [51]. An equivalent DBN representation for activity in Figure.1 is shown in Figure.3.

DBN is used in various scenario to model a process. For example, Koller & Huang have a series of papers [40, 34] about recognizing the traffic patterns by a handcrafted DBN. Their vision system employed a contour tracker and an affine motion model based Kalman filter to extract vehicle trajectories. On top of that, a DBN based symbolic reasoning system is able to make inferences about traffic events such as vehicle changing lanes and stalling. Nicolas & Thonnat also used DBN in various surveillance applications [45].

Another important series of work is carried out by Garg, Pavlovic and Rehg [26, 54]. They extended DBN into input/output DBN with an Adaboost training infrastructure. Visual and audio streams are fused together by DBN. The parameters are initialized by standard DBN parameter estimation and then enhanced by Adaboost iteration.

Qualitative Temporal Bayesian Network (QTBN) is suggested as a combination of TimeNet and DBN [13]. It uses DBN to perform evidence-accumulating at each time step then feeds the result probability into TimeNet to do bookkeeping. It has the virtue to

achieve various time granularities. But, the interface between two components is not justified; learning on such an infrastructure is yet to be developed and the efficiency remains questionable.

Overall, DBN has the fundamental concept of an ongoing process that makes it a powerful prototype to perform inference in the activity domain. However, DBNs lack the ability to model the duration distribution other than the exponential distribution and therefore constitutes another incentive to this presented work.

2.8 Stochastic Context Free Grammar

Stochastic Context Free Grammar (SCFG) is a powerful tool to model activity, first proposed by Ivanov and Bobick [38]. They enhanced the Earley-Stolcke parsing algorithm on SCFG in the activity domain and tested SCFG in a parking lot scenario with a fairly simple tracking system. Moore & Essa [48] extended that work into indoor desktop setup and tested on a card game scenario which had a more rigorous rule. Another more recent work was done by Minnen & Essa on the problem of understanding tower Hanoi [44].

SCFG works by taking in a single terminal symbol stream and using a combined top-down/bottom-up approach to either evolve non-terminal symbols or combine terminal symbols into non-terminal symbol. Appropriate production rules are chosen to support the parsing process. Probability is attached to each production rules and the overall evaluation is then obtained by multiplying the probabilities from each individual step. It is very easy to write a strictly ordered procedure in SCFG and carry out recognition on it with a fairly reliable observation module. An equivalent SCFG representation for activity in Figure.1 is shown in Figure.4.

SCFG however has a serious limitation that prevents it from going further. To change a feature vector into symbols and place them in an appropriate sequence, a conversion layer must exist. Usually, such a layer is handmade. To force a multi-dimension feature vector into a single symbol stream, either there are a magnitude more symbol or multiple symbols

$$\begin{array}{lll}
S \rightarrow I|\epsilon S & I \rightarrow aI|aJ & J \rightarrow \epsilon J|K|L|M \\
K \rightarrow bK|bN|bO|bL & L \rightarrow \gamma L|\gamma O|\gamma R|\gamma P & M \rightarrow cM|cL|cP|cQ \\
N \rightarrow \epsilon N|O & O \rightarrow cO|cR & \\
P \rightarrow bP|bR & Q \rightarrow \epsilon Q|P & \\
R \rightarrow \epsilon R|T & T \rightarrow dT|dU & U \rightarrow \epsilon U|\phi
\end{array}$$

where

S : Starting symbol

$I/J/K/L/M/N/O/P/Q/R/T/U$: Non-terminal symbol

$a/b/c/d/\gamma/\epsilon$: terminal symbol for observation of motion A/B/C/D/B+C/blank

ϕ : null symbol

Figure 4: Example SCFG to represent an activity of four elements:A/B/C/D, where A happens before B/C, and B/C happens before D

have to be checked simultaneously. The first approach is dangerous that when the feature vector represents parallel threads, it needs a magnitude more symbols to achieve the equivalent expressing power. In the example shown in Figure.4, 13 non-terminal symbols are needed to represent such a simple activity of 4 intervals. The second approach is violating the context free assumption and computing on such grammar is known to be inefficient.

Another problem with SCFG comes from the limitation on Context Free Grammar (CFG). CFG is fundamentally a symbol based algorithm. Though it is augmented to become SCFG, SCFG still makes hard decision on choosing the next production rule. Only when there is a production can the corresponding interpretation move forward. The probability only comes afterwards. Therefore, all potential subsequent states need to be indicated by production rules. (Contrast that with DBN, in which by default, the subsequent state space is any combination of the hidden variables and does not need to be pointed out explicitly.) When input is noisy, insertion and deletion error will become a major problem in activity recognition. One way to handle an insertion or an deletion error in SCFG is to add null transition and duplicate each production rule for any possible subsequent situation. Null transition leads to an exhaustive recursive searching usually controlled by a limited search depth – another hard decision. Furthermore, the above duplication usually increases the production set size by at least one order of magnitude. If the search depth is large and there are multiple streams, the resulting production rule set can be too large for a timely

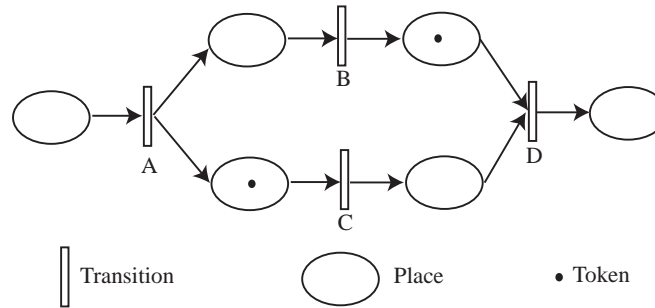


Figure 5: Example Petri Net to represent an activity of four elements: A/B/C/D, where A happens before B/C and B/C happens before D. The events are represented by transition. The current status is represented by the token.

computation. These problems should be addressed before SCFG can be applied to the real world problem of multiple agents and noisy sensors.

2.9 Stochastic Petri Net

Petri-Net is a long established tool in software engineering. It has been successfully applied to the performance analysis of concurrency and synchronization. Petri net is a directed bipartite graph [9]. It has 2 types of nodes: place and transition. Places are the holder of tokens while transitions regulate the movement of tokens. A marking of a Petri net represents the current status of the net. It is a vector, where each entry is the number of tokens in the corresponding place. Directed links connect places to transitions then back to places. There are 3 types of links: input links, output links and inhibitor links. The firing of a transition node is controlled by the input links and inhibitor links. A transition is enabled iff all of its input places contain no less tokens than prescribed by the input links and all of the inhibitor places contain fewer tokens than prescribed by the inhibitor links. Once a transition fired, tokens from input places are removed and new tokens are put into the output places. An equivalent Petri Net representation for activity in Figure.1 is shown in Figure.5.

To model the time in the underlying process, Petri net was augmented to associate firing delays to the transitions [47, 64]. Generalized stochastic Petri nets (GSPN) [42] is then introduced with further relaxation to allow immediate transition with zero firing delay,

while immediate transition has priority over delayed firing. To make GSPN analytical tractable, several different assumptions of the delay of transition are introduced, which constitute specialized GSPN, e.g., Markov SPN, Semi-markov SPN, etc [11].

When Petri net and GSPN are introduced, they are designed to model an open process and solve for the steady state and the reachability set. Most of the applications are in software engineering and network for synchronization analysis. There are some preliminary research to use Petri net as a modelling tool to represent simple human activity [27].

Petri net and GSPN are strong to model multiple-agent process. But they are not designed to activity recognition. In activity domain, most of observations are noisy. The core problem is to compute the posterior probability of the underlying process given the observation. Petri net and GSPN are designed to handle noisy observation. In fact, there is no concept of observation and underlying “hidden” process in genuine petri net and GSPN. All transition is definite, just the transition time can be stochastic. That is not desirable in activity modelling. But the idea of using tokens on the place nodes to mark the progress of process is one of the inspiration that leads to this work. Another minor problem is the genuine transition node only supports logic “AND”. This can be easily fixed by augmenting into a conditional probability function.

2.10 Symbolic Network Approach

In the artificial intelligence community, lower level perceptual input is usually filtered to generate symbolic values. The temporal and logical constraints between those symbols are represented in either frame like or network like language. The problem of activity recognition is translated into finding the appropriate instances of symbols that satisfy all the constraints. A few methods have been proposed to exploit the temporal constraints and facilitate an efficient recognition, which will be surveyed next.

2.10.1 PNF-Network

As described in Allen's algebra, there are 13 types of temporal relations between two intervals. Pinhanez & Bobick [56] suggest taking reasonable simplification to consider only Past/Now/Future relation and constructed PNF-Network. They argue in most situations, it is not necessary to know the exact time for a interval node, rather it only matters to know the simple trinary information of whether it happens in the Past, Now or Future. For each pair of interval (I_i, I_j) , there is a triple set $\langle R_p, R_n, R_f \rangle \in \langle P, N, F \rangle^3$, where R_p can be Past (P), Now (N) or Future (F), representing I_j happens in the past, now, or in the future when I_i is Past, R_n represents the value of I_j when I_i is Now, R_f represents the value of I_j when I_i is Future. Such triple set constitutes the constraint between two intervals. The prior knowledge of activity is expressed as the constraints between the elemental intervals. The enclosure of such constraints forms a PNF-network. The input for a PNF-network is binary positive evidence for each interval node. When the lower level detector reports either true for N(ow) or false for P(ast)/F(uture), the network is triggered and updated through time expansion and PNF propagation. The minimal domain that satisfies the constraints, the previous status of network and current observation will become the new status of the network. The sequence of PNF-Network states mark the explanation of the observed activity.

PNF-network is deterministic. As with Allen's algebra, it makes hard decision to narrow the search scope. This approach makes it very difficult to deal with faulty evidence. PNF-network lacks the ability to use the probability/confidence information. Instead, a hard threshold is necessary. In this way, it has to pre-allocate a set of search options for preset known number of faulty observation and exhaustively search the space. Such add-on patches will not only reduce the claimed computation efficiency but will also make choosing the search boundary in the real world problem very difficult.

2.10.2 Frame-based Method

Vu, Bremond and Thonnat proposed a frame-based language to describe activity [68, 67]. Their language has 3 parts: 1) actor as the subject for constraint, 2) constraints that consist of temporal and logical operators, and 3) scenarios that consist of actors and constraints. To recognize an activity, they follow the standard AI approach: they first filter the sensor output to generate the symbol streams, then use the description frames to instantiate the variables with the detected actors and the recognized sub-scenarios. The best candidate that fits the constraint becomes the evidence and is used for the following inference. The candidate recognized scenarios and the temporal constraints are ordered by time, so that a strictly non-overlapped order in candidate intervals can be enforced, which leads to an almost linear search. A hierarchy of at most two sub-scenarios is generated during the compiling stage, and thus narrows the search scope for the super scenario. Some other model can be found in [17].

Such frame based methods have a tremendous advantage in describing activities. By including any operators in the constraint, their expressive power is virtually equivalent to a natural language. However, this advantage in specifying the activity may become a problem when recognizing a model in such a language. When the description hierarchy becomes too huge and the scenario becomes large, the potential number of sub-scenarios can become exponentially large which may require a strong pruning process and potentially hard decision. Also, like any symbolic approach, it inherits the problems of deterministic approach in dealing with noisy sensors. All these limitations hinder its ability in a noisy real world problem.

CHAPTER III

REPRESENTING THE TEMPORAL SEQUENCES: PROPAGATION NET

Human activity is a special case of temporal sequences. To recognize it, we need to first represent it. Different methods have different perspective and assumptions. As we have surveyed in the previous sections, the following points are the characteristics of human activity but they are not represented efficiently in existing methods:

1. the natural concept of stream: to recognize an activity, a model first has to prescribe the activity based on human knowledge. It is easy to see that human description is based on time axis in a stream-like fashion. A model that keeps this stream fashion in an explicit way is easy to understand and convenient to transcribe the knowledge, which constitutes an good tool for expert to define an activity.
2. the parallel streams that coexist and entwine together: as is discussed in the previous chapter, human activity has multiple streams that have rich interactions between them. Those interactions are key elements of an activity. How to describe these interactions and use them as heuristics or constraints in inference processes efficiently is the key to recognize an activity. A naive solution that use combinatorially more states to keep track to multiple streams will hinder the inference ability in complex activity.
3. a duration distribution other than the exponential distribution: the element of activity is interval rather than individual time point. Most of the intervals have a non-exponential distribution. Traditional model with static self-link can only model exponential distribution which hinders the model ability to transcribe the prior knowledge

and lose the ability to constrain the inference process.

4. a coherent history: as described previously, the second objective of any surveillance system is to establish the annotation and understand when did each component happen. This objective is essential to the third objective of discovering the anomaly and triggering alarm/action. To achieve these 2 objectives, it requires single coherent explanation rather than a distribution of what happened at each time step. The ability to generate such an history explanation is important in posterior analysis.
5. the ability to deal with noisy sensors: real sensing technologies/algorithms have noise. Any the sensor observation can be wrong at any time step. Recognition is only possible by integrating multiple sensors and prior knowledge. How to filter these noisy observation based on the guidance of prior knowledge is vital to the success of any activity system.

The deficiency of existing models with the above questions is summarized in table.1. For example, as surveyed in previous chapter, finite state machine (FSM) has a fundamental assumption that only one node is active at a time. This assumption is equivalent to assume there is only one active stream. Using FSM to represent multiple streams is very hard because it needs to use combinatoric more states/nodes. Original FSM uses static link to maintain the duration which leads to the exponentially distributed duration. Dynamic Bayesian Network (DBN) also uses static self-link to maintain the node duration and naturally it is incapable of handling non-exponential duration distribution. DBN only keeps a distribution of the states for record rather than how each node becomes active. This is part of the markov assumption that DBN relies on to maintain its efficiency in updating. But that renders it incapable to produce a single coherent explanation history.

To address the above problems and incorporate the characteristics of daily activity, we devise a new representation model, Propagation Network (P-Net). First, we introduce the

Table 1: the deficiency of existing models

models	problems				
	natural stream concept	entwined parallel streams	non-exponential duration distribution	coherent history	Handle noisy sensors
FSM		X	X		
coupled-HMM		X	X		
BNT	X			X	
DBN			X	X	
Petri-Net				X	X
SCFG		X	X		X
symbolic network	X	X			X

conceptual model of Propagation Network and how it manages to encapsulate the characteristics of activities; then we provide the precise computational model that implements the conceptual model. The inference and learning algorithms on P-Net will be presented in the following chapters.

3.1 *Philosophy*

The philosophy behind this work is to represent and interpret activity as *openly* and as *naturally* as possible. By *openly*, we mean that the model is conceptually understandable. By *naturally* we mean that the output conforms to human explanation habit. In all, we wish to present a tool that “mimics” human, a tool whose building blocks have specific meaning understandable to humans and whose output explains the process of activity as a human would.

3.2 *Conceptual Model*

As we have described, the building blocks for human activity are temporal intervals. Each interval has a specific meaning and can be associated with some specific perceptual evidence. They constitute the vocabulary for humans to describe a process.

Between these primitives, there are temporal and logical relations that are intrinsic to the activity definition. These relations are the constraints that control the evolving process

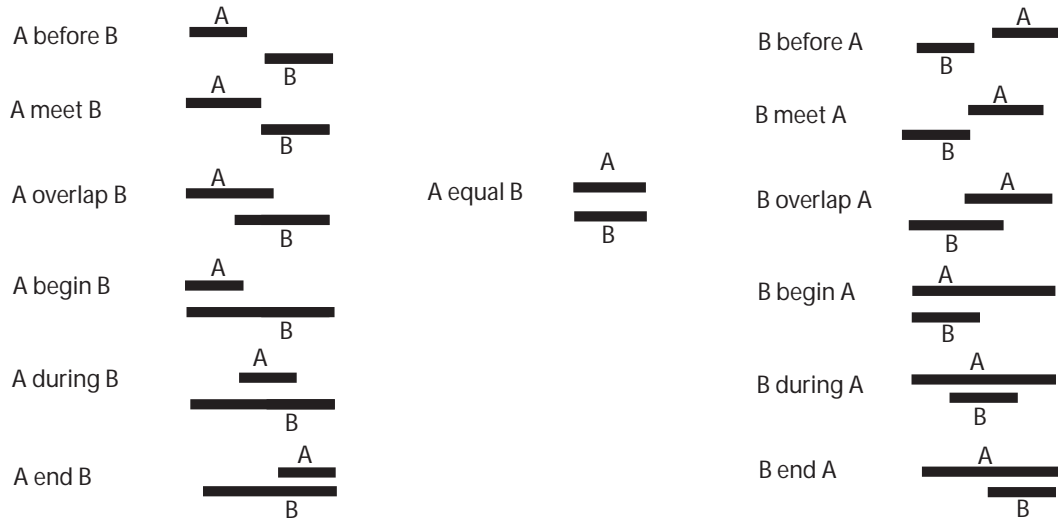


Figure 6: 13 temporal relation between 2 intervals

of activity.

3.2.1 Representing Temporal Relations

According to Allen's algebra, there are 13 legal temporal relations between two intervals. Among them, there are seven unique relationships: *before/meet/overlap/begin/during/end/equal*, whose definitions are shown in Figure. 6.

On the other hand, there exists an natural partial order between the temporal intervals. To represent the partial ordering, we only need one relationship *before*, which represents the partial order on temporal intervals as the following:

$A \geq B$: when the ending time of interval A is before the starting time of interval B

$A \leq B$: when the ending time of interval B is before the starting time of interval A

With this ordering we can generate a directed graph. The interval shall serve as the nodes while the existence of a directed link indicates a *before* relationship between the intervals.

However, the *before* relation alone cannot encapsulate the rich concept of temporal relations in activity domain. Fortunately, the remaining relationships can be encoded by a direct augmentation of the duration models and dummy nodes.

The dummy node is a fake node in the graph that does not represent any real motion. It

is introduced to mark a special time point, i.e., the beginning or ending point of a special moment. In Figure. 7, a dummy node is marked as x . Duration models can be attached to both nodes and links: the duration model of a node requires the node's activation interval to conform to a certain distribution of time while the duration model of a link prescribes the distribution of the time between the ending of a parent node and the starting of a child node. In Figure. 7, a special duration model attached to a node or link is marked by the condition on top of the symbol. Unless stated differently, the duration of dummy node is always 0, the duration of real node is always larger than 0 and the duration of a link is always no less than 0.

“A equal B” can be represented as a dummy node of duration 0 that leads to interval A and B by links of duration 0; then A and B leads to another dummy node by links of duration 0.

“A before B” can be represented as A leads to B by a link of duration at least 1.

“A meet B” can be represented as A leads to B by a link of duration 0.

“A overlap B” can be represented as a dummy node x_1 leads to A with a link of duration 0 while x_1 leads to B with a link of duration smaller than A's duration; A leads to another dummy node x_2 by a link of duration 0 while B leads to x_2 with a link of duration larger than 0.

“A begin B” can be represented as a dummy node x_1 leads to both A and B with links of duration 0; A leads to another dummy node x_2 by a link of duration bigger than 0 while B leads to x_2 with a link of duration 0.

“A during B” can be represented as a dummy node x_1 leads to A with a link of duration larger than 0 while x_1 leads to B with link of duration 0; A leads to another dummy node x_2 by a link of duration larger than 0 while B leads to x_2 with a link of duration 0.

“A end B” can be represented as a dummy node x_1 leads to A with a link of duration larger than 0 while x_1 leads to B with link of duration 0; A leads to another dummy node x_2 by a link of duration 0 while B leads to x_2 with a link of duration 0.

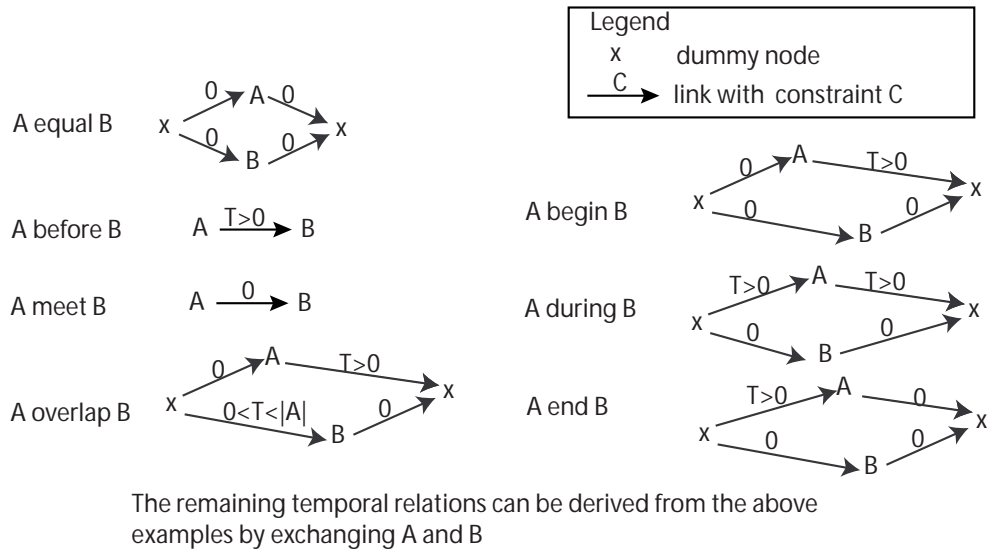


Figure 7: An example of temporal relations with dummy nodes

The actual graphs of the above notation are summarized in Figure.7.

In this way, we have the full representation power of Allen’s algebra by an augmented partially ordered graph. This type of graph shall serve as the temporal part of the conceptual model to describe the activity.

3.2.2 Representing Logic Relation

Our vocabulary to describe activity is a set of motions. Motion (by our definition) is supposed to be self-modulated, that is, once started, it is an independent straightforward process. Here we emphasize that the influence between the intervals only exists at the starting point of an interval. Therefore, during any interval, there is no place for external force. This appears to be a very strong limitation of representation power. But as we stated, motion is supposed to be elementary. If there needs to exert the influence from outside after the starting moment, the interval should be further segmented and those outside influences can be explicitly represented as the constraints on the interval’s starting point. In this sense, we are not compromising overall representation power, but improving the formality and potentially the computation efficiency of our description.

Since the effect from ancestor nodes are included in parent nodes, we take a further

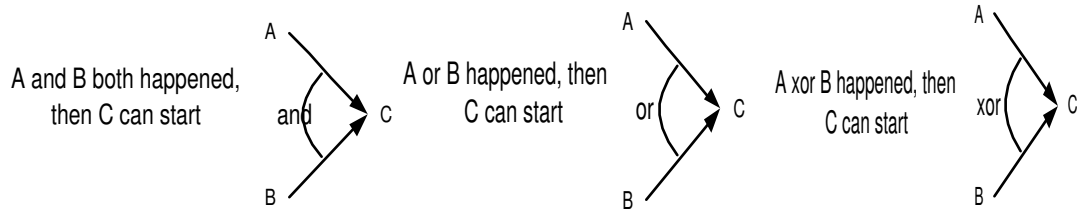


Figure 8: An example of logic relation

Markov assumption: a logic condition only exists in the adjacent intervals, which assumes a logical relation only exists between the end of parent interval and the start of child interval. Therefore, it can be carried upon *before* relation, the only temporal relation we use in our model. In such way, a unified form to integrate temporal and logic relation together is achieved and the computational efficiency by treating the two relations as the same is enabled.

One might argue some interval terminates at the condition of another interval's progress, which is a constraint on the ending point of an interval. But in fact, this kind of constraint implicitly exists as the reverse effect of a constraint on the following intervals. For example, the constraint of "A before B" has the implicit effect on A's ending point that, when B is prompted to start, A is implicitly forced to end. Explicit constraint on the ending point is not necessary, as they can be translated into the above implicit constraints. For example, "A ends once B ends" is simply "A end B" or "A equal B", which is shown in the previous section as representable by our *Before* relation.

So far, the links to represent the partial ordering are the temporal parts of the constraints on the starting point. To put the logical constraints on the links and treat the logical and temporal constraints uniformly, we follow the BNT style to define conditional probability functions: multiple links into a node imply a joint constraint on the starting point of the corresponding interval. Examples of and/or logic relations are shown in Figure.8 which are also used in the following experiments. It is well known that any logic relation can be represented by the general form of a conditional probability function.

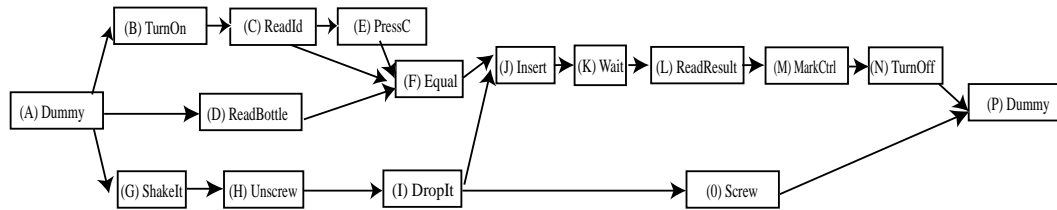


Figure 9: Conceptual diagram for a glucose meter calibration process

3.2.3 Prescribing conceptual model

A P-Net represents an activity by associating one node with each primitive motion in the activity. Two extra dummy nodes are added to mark the start and end of the activity. The state of each node, described in detail in the next section, includes the most recent starting time of that node and the corresponding duration. Links in the network correspond to the partial order constraints and the logical constraints between the pairs of motions. Figure.9 provides an example of a conceptual diagram for the P-Net that represents the blood glucose monitor calibration task that will be described in detail in section.5.6

The temporal and logical constraints regulate the activation model and the duration model for each node. The activation model describes when and with what probability the node can become active, given the parents' states. The duration model describe the probability of the node to remain active for the next time step, given the current state of the node. The two models are unified in one conditional probability function given its own current state and its parents' states. Following the standard graphical model notation, multiple links entering a node in the conceptual model implies a joint conditional probability function for the child node.

As a generative model, P-Net needs to specify what can be observed for each motion. That behavior is prescribed by the evidence component, a probabilistic observation model associated with each node. The perception system can only provide raw feature like coordination/speed/gradient. P-Net components motions have specific meaning. There exists a gap between the raw feature and the motions. Instead of letting the P-Net observation model describe the raw feature directly, we choose to use a specialized layer to convert raw

feature to indicative observation then P-Net observation model only describe the indicative feature. There is one indicative scale per each motion. This indicative scale represents how likely the corresponding motion is happening. One common choice for such a layer is to use a simple Bayesian network. Each leaf in such a BNT corresponds to a raw feature while the root represents the motion. The posterior probability of the root given the evidence shall serve as the indicative scale for the corresponding motion. Another choice is to use HMMs. The HMM posterior probability shall serve as the indicative observation for the evidence component [38]. Based upon the indicative feature, each observation model describes with what probability the corresponding indicative scale, conditioned upon the state of the node, can be observed.

A cycle on the conceptual model is allowed. It represents the multiple occurrences of the same set of primitive motions. The circle appears to be a violation of the partial ordering. But in fact, the circle only appears when the explanation path goes through the same node multiple times. Then there are multiple intervals corresponding to the same node. And for each individual interval, the partial order between the parent and child intervals is still preserved. In other words, this pseudo-circle exists only because each node in a conceptual model corresponds to all the temporal intervals that may exist for the same primitive motion. In the next section, the computational model will roll out the explanation path effectively breaking the cycle.

3.3 Computational Model

As a diagram, a conceptual model is the tool for a human expert to specify a P-Net. The actual computation of P-Net is carried out on a DBN style model, which we call the *computational model*. The definition and computation of a generic DBN can be found in [51].

The computational model of P-Net is a DBN enhanced with a duration model. In fact, it unrolls the conceptual model at each time step to form a DBN. There are two types of nodes: the hidden node and its unique corresponding observable evidence node. As a DBN,

it has two types of links: inter-slice (across time) link that updates the states over time and the intra-slice link that connects a hidden node with an evidence node.

In a traditional DBN, the state of each hidden node is binary. In a P-Net, each node state r_i^t is augmented to record the last activation, which is represented by the activation time s_i^t and the activation duration d_i^t . The inter-slice links represent the inter-slice conditional probability table, which is constructed to make sure that (1) the activation conforms to the partial order of events, (2) the activation duration increases one at a time, and (3) the activation duration as well the blank period duration between parent and child nodes conforms to a duration model. The intra-slice link connects the hidden node and its unique evidence node, thereby representing the observation model.

Formally, a P-Net is defined as $\mathcal{P} = \{R, \Phi, B, \Theta, O\}$. The definition of each component follows:

R : the random variable set. The joint distribution of R represents the states of P-Net. We define the state of each node at time t to be the tuple (s, d) where s is when the node started and d is the duration that it was (or is) active. Also, a node is `nil` if there has been no activation within some reasonable time window tracing back from the current time. Therefore, the state set for node i at time t is defined as the following $\{(\emptyset) \cup \langle s, d \rangle, d \geq 0, s + d \leq t\}$, which represents that at time t , node i is active from s to $s + d$ while \emptyset represents no activation within a reasonable window behind t . At each time step, we associate one random variable r_i^t with each node, $P(r_i^t) = P(r_i^t = \langle s, d \rangle)$. As a random variable, we naturally have $P(r_i^t = \emptyset) + \sum_{s,d} P(r_i^t = \langle s, d \rangle) = 1$

There is an important difference between an inactive state (finished and null state) and an active state in R . An active node is defined as $r_i^t = \langle s, d \rangle$, where $s + d = t$. An active node is able to determine its own state and remain active. An inactive node can serve as the parent, which can conceptually trigger a child node. Nodes are triggered, become active, and then terminate and trigger child nodes. This sequence forms a sense of propagation hence the name *Propagation Net*.

Multiple activations may exist in the history of the node. In our representation, only the most recent activation has any influence on the propagation process. We believe this Markov simplification is reasonable when there is no counting condition on the process of the activity (we can retrieve the counting information in postprocessing, but the computation process is not influenced by such counting). We agree there are situations when counting matters, but that is a compromise we make to achieve unified state representation and ensure computational efficiency. If we wish to add the counting ability, we can do so by attaching a stack to those nodes that need counting. Theoretically, it is equivalent to augment the finite state machine into the pushdown automata. Due to the application domain, but the computational complexity grows.

Φ : The causal relationship Φ_i for r_i^{t+1} defines the state transition of node i at time step t . It is defined over the joint set of r_i^t and all r_j^t where j represents the parent nodes of i .

In principal, Φ can be quite arbitrary. However, because of the duration model, the joint distribution of r_i^t and all r_j^t is exponentially big. Therefore, without proper constraints, the dimensionality of Φ function is huge even for toy level problems. Therefore, we enforce the following constraints on three mutually exclusive conditions:

1. When any of the parent nodes is still active at the last step, the child node has to be inactive. This assumption enforces a staged traversal through the network.

Formally, we define Φ_1 as following: if there is a parent j of node i that is active at time t , then the probability of node i being active at time $t + 1$ is zero.

2. When node i is active at time t (formally $r_i^t = \langle s, t - s \rangle$), Φ_2 depends only on how long the node has been active; that is, once a node is active, only its duration model impacts the likelihood of continuing its activity. Φ_2 is designed to maintain the duration distribution.

Let us define

$$\Psi_i(d) = \int_{d+1}^{\infty} D_i / \int_d^{\infty} D_i$$

This $\Psi_i(d)$ is the probability that a node will remain active at time $t + 1$ if it has been active for duration d at time t . Its compliment is the probability that the node will cease being active. Therefore, when a node chooses to remain active, Φ_2 equals Ψ and when it chooses to stop, Φ_2 equals $1 - \Psi$.

3. The remainder of the Φ PDF is called the activation function and is used when node i is yet to be active, but its parents are finished. This “triggering” probability function is restricted to be a function of only how long ago each of its parents terminated their activity. That is, it is only a function of $t - (s_j + d_j)$ for parent j rather than individual s_j and d_j . This restriction again reduces the combinatorics of this conditional probability function and allows the system to be trained on a reasonable amount of data. The node only has two choices, either remains inactive: $r_i^{t+1} = r_i^t$, or becomes active with activation duration 1: $r_i^{t+1} = \langle t, 1 \rangle$.

Formally, Φ can be expressed as following function

$$\Phi(r_i^{t+1} = \langle \hat{s}_i, \hat{d}_i \rangle \mid r_j^t = \langle s_j, d_j \rangle, r_i^t = \langle s_i, d_i \rangle) = \begin{cases} \Phi_1, & \text{if } \exists j : s_j + d_j = t \\ \Phi_2, & \text{if } s_i + d_i = t \\ \Phi_3, & \text{else} \end{cases} \quad (1)$$

$$\Phi_1 = \begin{cases} 1, & \text{if } r_i^{t+1} = r_i^t \\ 0, & \text{else} \end{cases} \quad (2)$$

$$\Phi_2 = \begin{cases} \Psi_i(d), & \text{if } \hat{s}_i^{t+1} = s_i^t \wedge \hat{d}_i^{t+1} = d_i^t + 1 \\ 1 - \Psi_i(d), & \text{if } \hat{s}_i^{t+1} = s_i^t \wedge \hat{d}_i^{t+1} = d_i^t \\ 0, & \text{else} \end{cases} \quad (3)$$

$$\text{where } \Psi_i(d) = \int_{d+1}^{\infty} N_i(u_i, v_i) / \int_d^{\infty} N_i(u_i, v_i)$$

$$\Phi_3 = \begin{cases} \Phi_3(r_i^{t+1} | t - s_j - d_j), & \text{if } r_i^{t+1} = r_i^t \vee (\hat{s}_i^{t+1} = t \wedge \hat{d}_i^{t+1} = 1) \\ 0, & \text{else} \end{cases} \quad (4)$$

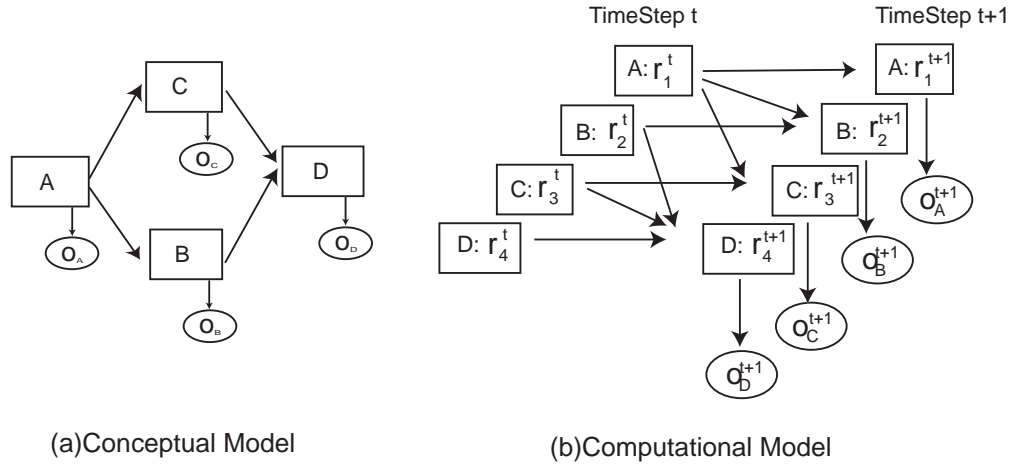


Figure 10: Comparison of a conceptual model and its corresponding computational model

$O : o^t = \{o_i^t\}$, the actual observation sequence for P-Net. For each node i , there is a unique indicative observation $o_i \in [0, 1]$, and o_i^t represents the integral evidence for node i being active at time t . O is an N -dimension sequence of length T , as there will be one observation for each node at each time step.

B : Every node has an observation model

$$B(o_i^t) = P(o_i^t | r_i^t) = \begin{cases} Q_i(o_i^t), & \text{if } r_i^t \text{ is active} \\ \overline{Q}_i, & \text{else} \end{cases} \quad (5)$$

In the rest of the thesis, B bears the property that, for node i , the observation of all active states conform to a unified distribution Q_i , while the observation of all the inactive states conform to another distribution \overline{Q}_i . In this way, the actual activation time and the duration are ignored, dramatically reducing the parameters needed to specify a P-Net.

Θ_i : the initial distribution of r_i^0 , $\Theta_i = P(r_i^0)$. When the sequence is long enough, the effect of Θ is minimal.

This formalizes our computational model, as illustrated by Figure.10.

3.4 Complexity

The complexity of a P-Net consists of two parts: the state complexity and the parameter complexity. The state complexity is the space needed to store the current status of the model. The parameter complexity is the number of parameters needed to construct such a model.

The current status of a P-Net is represented by the joint distribution of R . Therefore, a P-Net's state complexity is precisely the size of the joint distribution. If we let the maximal duration of any node and link be C , then the number of the different states in one node is $o(C^2/2)$; let us further denote the number of random variables in R to be N , the size of the joint distribution of R is $o(C^{2N}/2^N)$. Given an observation sequence O^T , this joint distribution at time step t can be obtained iteratively as the following:

$$\begin{aligned} P(R^{t+1}|O^{t+1}) &= \sum_{R^t} [P(R^t|O^t) \cdot P(R^{t+1}|R^t)P(o^{t+1}|R^{t+1})] / P(o^{t+1}) \\ &= \sum_{R^t} [P(R^t|O^t) \cdot \prod_i P(r_i^{t+1}|r_j^t, \forall j)P(o_i^{t+1}|r_i^{t+1})] / P(o^{t+1}) \end{aligned} \quad (6)$$

Unfortunately, the dimension of this joint distribution is too large. Summing over this distribution or even finding the maximal point over this distribution is impossible. No ordinary inference algorithm on DBN can work without a simplification. How to simplify it and still carry on an effective computation is the core problem for an inference algorithm. We have two solutions that will be presented in the next two chapters.

The parameters in the P-Net include Φ , B and Θ . Since we enforce that every P-Net has a starting point and we are only interested in the whole process, the initial states of a P-Net can always be set as: only the starting dummy node is active, so $|\Theta|$ is $O(1)$. The complexity of B depends on the actual observation model for the individual node. If we choose the Gaussian model for all nodes, for each node we need to specify two 1-dimension Gaussian models for active and inactive observations, therefore $|B|$ is $O(4N)$. Φ is the major part of the parameters, which is defined by three parts Φ_1, Φ_2, Φ_3 as shown in Equation.1. The parameters needed for Φ_1 are simply the parent information. If we let

the number of links in P-Net be M , then $|\Phi_1|$ is $O(M)$. Φ_2 depends on the choice of the duration model for the individual node and link. If using a Gaussian distribution, then for each node and link we need to specify one 1-dimensional Gaussian distribution, therefore, $|\Phi_2|$ is $O(2M + 2N)$. The triggering function Φ_3 depends on the actual topology of the P-Net. Let the maximal number of parents be E , then for one node there are $o(C^E)$ different parent states. Since the new activation only starts from duration 1, for each node we need $o(C^E)$ parameters. Therefore, $|\Phi_3|$ is $O(N \cdot C^E)$. So the total parameter complexity is

$$O(1) + O(4N) + O(M) + O(2M + 2N) + o(N \cdot C^E) = O(M + N \cdot C^E)$$

When C and E are large, this equation indicates the computation is still too big to be manageable. In the rest of the thesis, we choose to use noisy-or/noisy-and for Φ_3 , thereby reducing $|\Phi_3|$ to $O(N \cdot C \cdot E)$ and the total parameter complexity then becomes $O(M + N \cdot C \cdot E)$

CHAPTER IV

P-NET RECOGNITION BY LOCAL MAXIMAL SEARCH ALGORITHM

P-Nets have been designed as a tool for activity recognition. To fulfill the task, we need methods to address the following problems:

1. To classify an observation sequence O^T into one of the activities defined by P-Nets, we have to calculate the probability of O^T given a P-Net \mathcal{P} .
2. To determine the time and the duration of the individual primitives that compose the activity, we have to compute the most likely internal state sequence given a P-Net \mathcal{P} .

These two problems can be addressed together by finding the most likely internal sequence. Then the observation probability given the most likely internal sequence can be used to approximate the total observation probability. This state sequence can be computed from:

$$\begin{aligned} P(R^{t+1}|O^{t+1}) &\approx \text{Max}_{R^t} P(R^t|O^t)P(R^{t+1}|R^t)P(O^{t+1}|R^{t+1})/P(o^{t+1}) \\ &\approx \text{Max}_{R^t} P(R^t|O^t) \cdot \prod_i P(r_i^{t+1}|r_j^t, \forall j)P(o_i^{t+1}|r_i^{t+1})/P(o^{t+1}) \end{aligned} \quad (7)$$

In Equation.7, r_i^{t+1} actually depends on the joint distribution of all parents $\{r_j^t, \forall j\}$. Because of the dimensionality of the joint distribution, it is not tractable to estimate the full joint distribution, let alone to carry out the standard Bayesian inference. In this thesis, we introduce 2 algorithms to approximate Equation.7. In this chapter, we describe Local Maximal Search Algorithm (LMSA), a Viterbi-like iteration based algorithm. In the next chapter, we introduce Discrete Condensation, a particle filter based algorithm.

4.1 Forward-Backward Calculation

Finding the maximal point over the distribution in Equation.7 is too expensive. To simplify the calculation, here we will assume the independence of all parent variables, i.e. use the product of the marginal distribution of the parent variables to represent the joint distribution. Once this is done, finding the maximal point in the joint distribution can be reduced to find the maximum for each parent, making the computation manageable. However, this independence assumption is too strong to be true; it will be relaxed in the next section by iteration.

Inspired by the Viterbi algorithm, we select the most likely tuple $\langle s_j, d_j \rangle$ in r_j^t as the parent state and use it to compute r_i^{t+1} . That is we substitute $P(r_i^{t+1}|r_j^t, \forall j)$ with $P(r_i^{t+1} | \langle s_j, d_j \rangle = \text{argmax}_{\langle s_j, d_j \rangle} P(r_j^t = \langle s_j, d_j \rangle), \forall j)$, and then record r_i^{t+1} with its parent information $r_j^t = \langle s_j, d_j \rangle$.

Here, we use maximum point in the joint parent distribution to generate the subsequent states. Another way is to let each point in the parent distribution generate the subsequent states and sum them together then find maximum point. The second method provides the possibility of generating an strong answer by summing up a few weak parents. However, this method requires iterating through the full joint distribution and it is intractable. Also, as explained before, our target is to provide one valid state path to explain the evidences just as a human would. Summing up multiple weak parents is conceptually unacceptable. Therefore, we believe our choice is suitable for the situation.

Altogether, Equation.7 is simplified as

$$\begin{aligned}
 P(R^{t+1}|O^{t+1}) &\approx \prod_i P(r_i^{t+1}|O^{t+1}) \\
 P(r_i^{t+1}|O^{t+1}) &\approx \text{Max}_{\{r_j^t, \forall j\}} P(r_i^{t+1}|r_j^t, \forall j) P(o_i^{t+1}|r_i^{t+1}) \cdot \prod_j P(r_j^t|O^t) / P(o_i^{t+1}) \\
 &\approx P(r_i^{t+1}|r_j^t, \forall j) P(o_i^{t+1}|r_i^{t+1}) \cdot \prod_j P(r_j^t|O^t) / P(o_i^{t+1}), \\
 &\text{where } r_j^t = \text{argmax}_{\langle s, d \rangle} P(r_j^t = \langle s, d \rangle)
 \end{aligned} \tag{8}$$

When all evidence is processed, the state of the maximal marginal probability on the dummy ending node marks the maximal explanation path. From this point, we go backward to retrieve the history path. The following recording function will iteratively trace back from the state of node i at time $t + 1$, $r_i^{t+1} = \langle s^{t+1}, d^{t+1} \rangle$ to parent states $r_j^t = \langle s^t, d^t \rangle$ and generate the candidate maximal path:

$$\{(t, j, s^t, d^t)\} = \text{History}(r_i^{t+1} = \langle s^{t+1}, d^{t+1} \rangle)$$

4.2 Iteration

The forward calculation gives us one local maximal explanation path through a P-Net, but when we assume the independence between the parents, the correlation between the parent nodes are ignored. The true joint distribution is not the multiplication of the parents' probabilities. The maximal point found by the above forward computation may not be the maximal point in the true joint distribution, as illustrated in Figure.11. To be even worse, this hallucinated maximal point may even be a invalid path. Two nodes may select different time/duration states on their mutual ancestors for its local maximal point, which causes the inconsistency in the maximal path. For the activity shown in Figure.10, child B may want A to terminate at t_1 , while child C may want A to terminate at t_2 . They both assume they succeed. Then the grandson D will obtain a inconsistent path from B and C. This inconsistency will be discovered when there is conflicting activation for a single node, i.e. node "A".

To unify the different state assumptions on the inconsistent node, we choose to iterate on top of the forward-backward calculation. Within each iteration, a candidate path is generated backwards from the ending nodes. Then we freeze any node that has no inconsistent assumption on itself and its ancestors. By "freeze" we mean that in the next iteration we restrict the node from generating any state other than the one within the frozen path. Then we find the most forward inconsistent node, which is the node that has an inconsistent

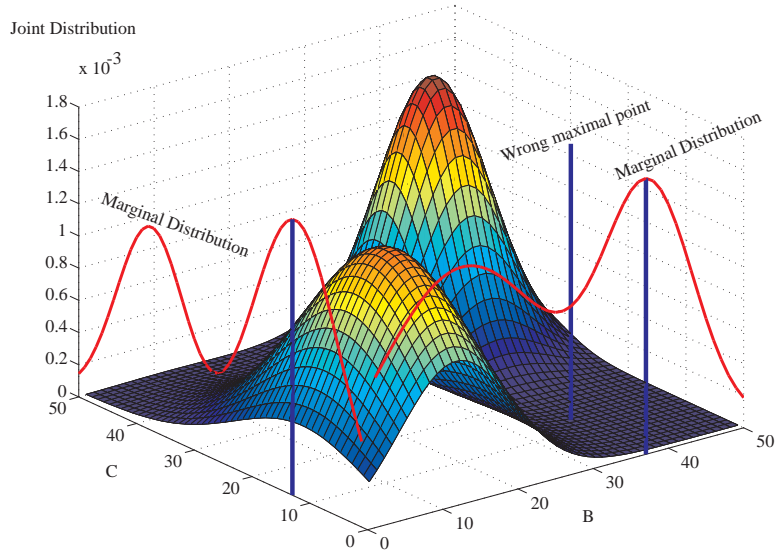


Figure 11: The independence assumption in forward computation may lead to a fake maximal point in true joint distribution.

assumption on itself but not on any of its ancestors. This node is where the inconsistency starts. To resolve it, we will assume that one activation prevails at a time and iterate through all the candidate activations. To do that, we enlist a stack to store the freezing options. At the beginning of each iteration, the stack will pop up a freezing node list so that a few nodes are frozen and the rest are left for forward-backward computation. After the forward-backward computation, we have either a valid explanation path or several freezing node lists to be pushed into the stack. For the same inconsistent example mentioned above, at the beginning of LMSA, the stack is initialized with NilConstraint of no freezing node; the first forward-backward computation comes up with the most forward inconsistent node A with two freezing option: $\langle s_1, d_1 \rangle$ and $\langle s_2, d_2 \rangle$; 2nd iteration goes forward after freezing A as $\langle s_1, d_1 \rangle$ and finds a valid path $Path_1$; 3rd iteration freezes A as $\langle s_2, d_2 \rangle$ and finds another valid path $Path_2$; now as the stack is empty, iteration terminates and the better of $Path_1$ and $Path_2$ becomes the final explanation path. The whole Local Maximal Search Algorithm (LMSA) on P-Net is summarized in Figure.12.

If there is inconsistency in the current forward-backward step, we will find a list of

Given: P-Net \mathcal{P} and the observation O
 Push NullConstraint into Q
 Repeat
 Pop freezing constraint c from Q
 Freeze P-Net with c
 for time $t=1$ to T
 Update Marginal Distribution $P(r_i^{t+1}|O^{t+1})$ according to Equation.8
 Keep history information for each state
 end
 Find max state $r_N^T = \langle s^T, d^T \rangle$ of the dummy end node N
 Trace back from r_N^T to retrieve history path $Path$
 if $Path$ has Conflict
 find the earliest conflict node k
 assemble freezing constraint candidates $\{c_i\}$ that are not k 's descendants
 push $\{c_i\}$ into Q
 else
 if $P(Path|O) > P(maxPath|O)$
 maxPath=Path
 end
 end
 Until Q is empty
 Output maxPath as the final explanation

Figure 12: Local Maximal Search Algorithm (LMSA) for P-Net inference

inconsistent nodes. Since the node activation interval has a partial ordering, within those inconsistent nodes, there is at least one node that is not child of the rest. That node will be frozen in the next iteration. Since it is frozen, there won't be any inconsistency at this particular node in the next forward-backward computation. Therefore, it is easy to see that if one node is frozen, then it will be kept frozen in the rest of the iterations. So in each iteration at least one more node will be frozen until all nodes are frozen at which point a valid consistent path has been found. When the path is consistent, the independence approximation is avoided and we have a valid explanation path with precise posterior probability. The iteration will terminate once all the candidate freezing options are explored. After generating all candidate local maximal paths, the path with maximal likelihood is deemed as the overall interpretation.

The iteration is caused by the inconsistency in forward-backward computation. Those

inconsistency can only happen at the nodes that branch into multiple children. In our experiment, such nodes and hereby the iterations are very limited. For each iteration, LMSA is polynomial and the overall speed is extremely fast. One major problem of LMSA is that it is only applicable when the whole process has been observed, which renders causal inference impossible. That is not desirable in surveillance scenarios. Another problem is, LMSA only considers local constraints and only explores the local maximal explanation. Therefore the result would not improve as more computational resource is allocated. These two problems shall be solved in the next chapter by a particle filter based algorithm.

4.3 Complexity

LMSA uses iterations to find the maximal explanation path. For each iteration, at each time step, the marginal distributions will be estimated from the parent node distributions. For each node, there are $o(C^2/2)$ states, where C is the maximal duration; for each state, we need to pick one state from each parent to get the posterior probability, that is $o(C \cdot E)$ computation, where E is maximal number of parents for any node. Therefore, there are at most $o(C^3 \cdot E/2)$ computations per iteration.

Since for each iteration we can resolve one more inconsistency and freeze that node, there are at most N iterations. Therefore, the total computation complexity to find a valid path in LMSA is $o(N \cdot C^3 \cdot E/2)$. Notice it is linear to the number of the nodes and polynomial to the complexity of the P-Net topology represented by C and E .

The computation buffer for LMSA is the marginal distribution and the history information for each node. The size of the marginal distribution is $o(N \cdot C^2/2)$. The history for each state at any time step is an array of parent states, whose size is $O(E)$; therefore for a sequence of T steps, LMSA needs at most $O(T \cdot N \cdot E \cdot C^2/2)$ space for history information.

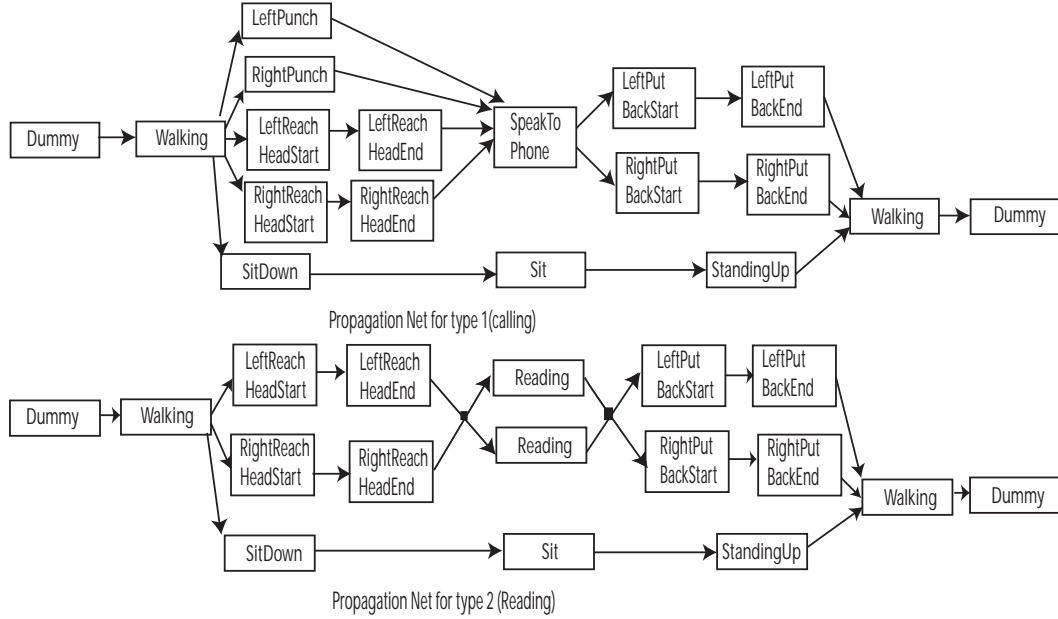


Figure 13: Propagation Nets for “calling” and “reading”

4.4 Experiment

To test our system, we constructed a simple indoor vision tracking system that uses simple color and shape information to track the hands and head of single person in a static indoor environment. This vision tracking is very noisy. (We could build more robust one, but that falls out of the research scope while the ability to utilize the noisy information and generate the correct answer is what this thesis concentrates on.) The lower level features include: motion of hands, distance of hands and head, orientation of head. On top of these features, we try to classify two types of daily activity. In type 1, “calling”, one walks in, sits on the chair, makes a phone call then walks out; in type 2, “reading”, one walks in, sits on the chair, reads a few pages of book, then walks out. We take 20 sequences of each type. Each sequence is 20-60 seconds long and sampled at 10 frames a second. The P-Net representations for type 1 and 2 are illustrated in Figure.13. The detector for each node is a handcrafted bayesian tree over a few threshold based classifiers. A modeltic graph of the P-Net based recognition system is shown in Figure.14

For each category, we use five out of 20 samples to obtain parameters, the remaining

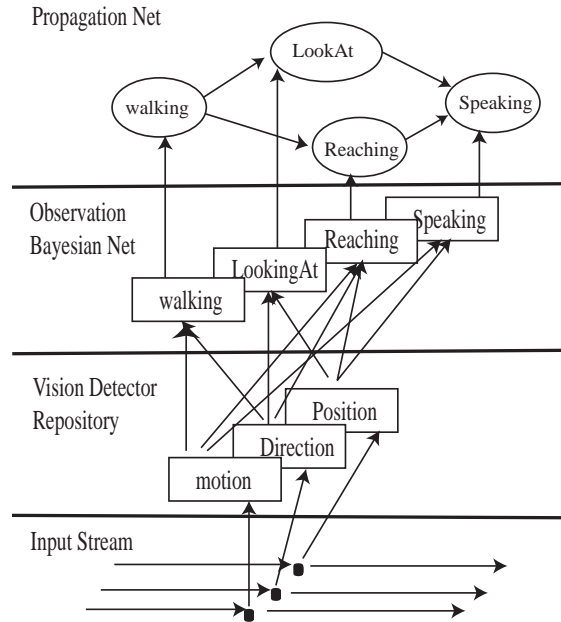


Figure 14: Architecture of a P-Net based activity recognition system

15 serve as testing set. As shown in Figure.15, all 30 testing samples can be correctly classified. The middle level output from the Bayesian network is quite poor, as the lower level detector is too simple and has many false alarms. The temporal constraints of the P-Net significantly improve the mid-level labelling. An example comparing middle level output and P-Net labelling on the `LeftPutbackStart` component is shown in Figure.16. We can see from the graph, P-Net successfully filter out most of the false alarms and pin down the actual activation interval.

Finally, to demonstrate the response of P-Net on the temporal information, we temporally scramble the data to maintain the same low level evidence but in the wrong order. The scrambled observation sequences are processed by the same P-Net. From the comparison with the original sequence in Figure.15, the impact of the temporal information become obvious. One can see that the correct interpretation is always the best, though the margins are not as high as needed to perform a robust detection. One reason is the two activities are quite similar in motions and thus hard to distinguish in terms of low level descriptions.

Since LMSA is based on each node's local maximum, its speed is polynomial. The

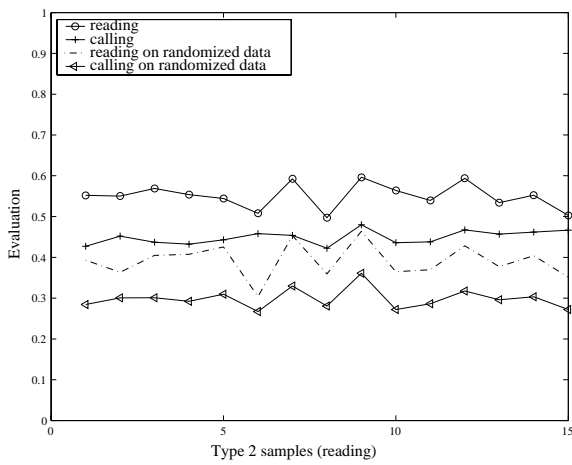
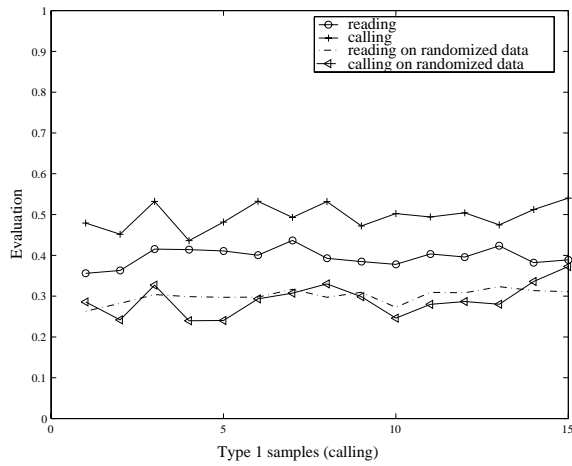


Figure 15: Results on “reading” and “calling” P-Nets

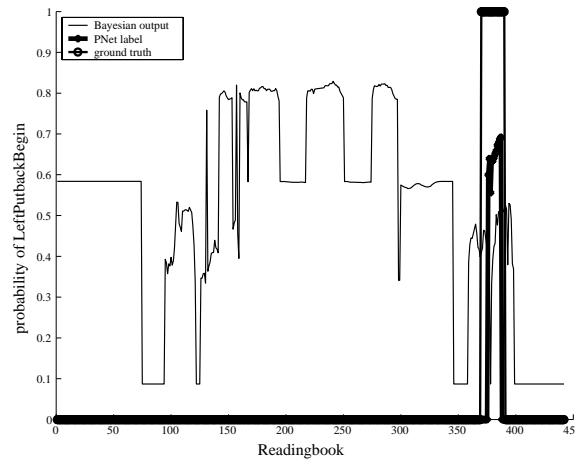


Figure 16: Comparison of straight Bayes net detector vs. P-Net labelling for LeftPutbackBegin running time for one iteration of 400 frame sequence is less than five seconds on a Pentium4 2Ghz workstation. The total time depends on how many inconsistent nodes may exist and how many iterations it takes to resolve the inconsistency. In our experiments, the iterations never exceed five.

These experiments show LMSA is a viable method to handle the joint probability distribution in P-Net. The expressing and detection power in P-Net are preserved, which enables us to handle noisy activity. However, LMSA still lacks the ability to do online recognition. In the next chapter, we will introduce a different solution to the inference problem in P-Net.

CHAPTER V

P-NET RECOGNITION BY DISCRETE CONDENSATION

5.1 Particle Filter

Particle filter, also known as Sequential Monte Carlo method, is a sophisticated technique for recursively estimating dynamic bayesian distribution [16]. There are numerous Monte Carlo sampling methods developed over the past decade, e.g. Condensation [37], bootstrap [29]. The key idea is to represent the posterior distribution by a set of random weighted samples and estimate the next distribution based upon these samples. These random samples are also called particles. When the true underlying distribution is Gaussian, the posterior distribution can be compute analytically by Kalman filter. But in most problems, such convenience is not available. Instead, large number of particles are spread over the possible state spaces and weighted frequency is the approximation to the distribution. As the number of particles increases, particle filter can approximate any distribution with an infinitely small difference. Another advantage of particle filter is that it can control the computational resource directly by controlling the particle numbers. Estimating the posterior distribution over a state space requires traversing the prior distribution. If the underlying distribution does not have an analytical form, such traversing is computationally impossible on state space of large dimensionality. In particle filters, the posterior estimation given a particular sample point is simple. The total computational complexity is now linear to the number of particles and becomes manageable.

A P-Net defines such a high dimensional space over the joint distribution of all nodes. In the previous chapter, we used the marginalized distributions to represent such a distribution

by assuming independence, however the iteration required to achieve a consistent solution makes the LMSA solution only viable as a batch job.

Instead, we now introduce a weighted particle filter to sample this joint distribution. The state of each node constitutes a separate dimension of the distribution. According to P-Net definition, the state of each node has the unique notion of active vs. inactive. At a particular point in the joint distribution, a few nodes are active while the rest are inactive. The active nodes marks the progress of the whole activity.

For conceptual reason, we use token to mark these active nodes. We can construct the particle as following: each weighted particle is comprised of at least one and potentially several tokens, and each token represents one active node in the parallel motion streams. The token maintains the current node's state (*i.e.*, $\langle s_i, d_i \rangle$) and its history. The particular set of tokens within a particle constitutes one sampling point in the giant state space which corresponds to one assumption about how the activity is evolved. A set of such particles constitute an estimation of the joint distribution. At each time step, all we must do is to re-estimate the new posterior distribution $P(R^{t+1}|O^{t+1})$ given the observation O^{t+1} based upon the current particle set $\{R_k^t \forall k\}$, which can be achieved by the following:

$$\begin{aligned}
 P(R^{t+1}|O^{t+1}) &\approx \sum_{\forall \text{particle } k} P(R_k^t|O^t) \cdot P(R^{t+1}|R_k^t)P(O^{t+1}|R^{t+1})/P(o^{t+1}) \\
 &= \sum_{\forall \text{particle } k} Weight_k \cdot \prod_i P(r_i^{t+1}|r_{jk}^t, \forall j)P(o_i^{t+1}|r_i^{t+1})/P(o^{t+1})
 \end{aligned} \tag{9}$$

5.2 Discrete State Space

The Condensation algorithm is a standard method to propagate the particles forward through time and generate the new sample particles based upon Equation.9 [36]. In the initial experiment, we follow the standard Condensation algorithm. At each time step, for each current particle, we sample the consequent state space and generate a fixed number of following particles and group them together. Then we re-sample this particle set by their posterior

probability to generate K new particles for the next time step. We found a practical problem with the performance of Condensation algorithm operating in our discrete state space. The re-sampling process in Condensation will quickly force all of the particles to be the same or nearly the same as the most likely particle. As the state space is too big, it is hard to find a successful fast random spreading mechanism to deal with this problem. Because the observation is noisy, the particle on the correct explaining path may not be the top K choice. A bad random spreading mechanism can easily cover the state space but still lose the track of correct explaining path. This over-clumping problem, also known as sample impoverishment, is also observed in other's research [37, 2]. This problem renders the standard Condensation algorithm unsuccessful in our setup: exploration of the state space is very slow, and a huge number of particles are required to explore low probability paths. In normal setup where I-Condensation is successful, a good importance function is used to avoid the above problem. However, since there are $O(C^{2N}/2^N)$ possible states in our network with N nodes and a window size of C , it is extremely difficult to find a reasonable importance function.

Here we introduce Discrete Condensation (D-Condensation). Rather than using importance function, we take advantage of the limited branching factor of the discrete state space to improve efficiency. For example, when node i is active at time t , $r_i^t = \langle s, t - s \rangle$, the next step must be either it remains active with a longer duration $r_i^{t+1} = \langle s, t - s + 1 \rangle$ or it becomes finished $r_i^{t+1} = \langle s, t - s \rangle$. We need not explore other state possibilities. The decision process for subsequent states is shown in Figure.17. For any particle, we may generate at most $O(2^J)$ subsequent particles, where J is the size of the largest cut set in the P-Net that separates the dummy *start* node from the dummy *end* node. So in each time step, for K initial particles, there are at most $O(K \cdot 2^J)$ subsequent particles. If we only keep K particles for the next time step, the total computation in D-condensation is $O(T \cdot M \cdot 2^J)$. In practice, 2^J is controlled by the topology of the underlying P-Net which reflects the complexity of the activity. So far in our experiments, the branching factor is

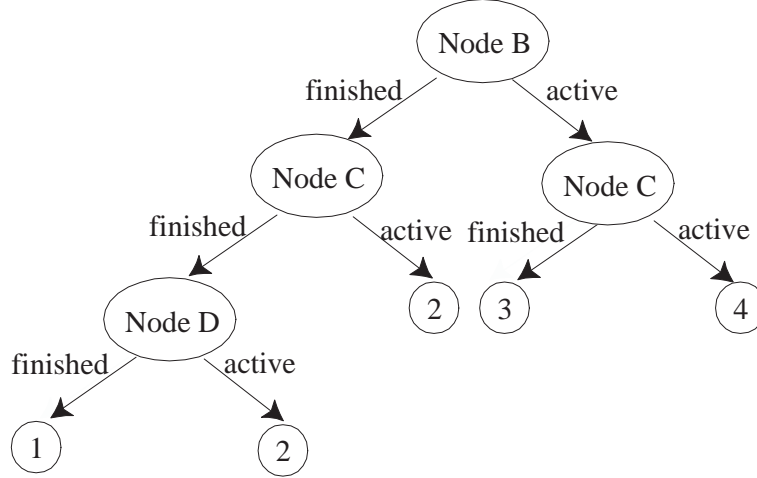


Figure 17: Potential consequent states when node B/C are active in last time step in the P-Net of Figure.10

limited that the P-Net interpretation is fast enough for real-time computation.

Accordingly, the weight for the new particle q at time step $t+1$ derived from particle k at time step t can be obtained from the following equation:

$$Weight_q = Weight_k \cdot \prod_i P(r_i^{t+1} | r_{jk}^t, \forall j) P(o_i^{t+1} | r_i^{t+1}) \quad (10)$$

In this manner, we can iteratively obtain the distribution of $P(R|O)$, and the best probability sequence will be the history path of the particle of the highest posterior probability.

5.3 Round-Robin Selection for D-Condensation

A particle filter does not help eliminating the combinatoric state space – it just provides a way to sample the space in a limited time. Discrete branching factors only enable us to efficiently explore the next state. How to maintain an efficient particle representation in the long term is simply another form of the combinatoric problem. In other words, how to select K out of $o(K \cdot 2^J)$ possible particles and still keep an accurate estimate of the whole distribution in the long term is the key to the solution.

Within each of these $o(K \cdot 2^J)$ possible particles, there is a set of currently active nodes, which we term the *frontier* of current explanation. The differences in *frontiers* are

Given: P-Net \mathcal{P} and the observation O
 Put initial particle $p_0 = \{r_0^0 = \langle 0, 1 \rangle\}$ in list L^1
 for time $t=1$ to T
 for each particle χ_i^t in L^t
 for all χ_i^t 's subsequent particle χ_j^{t+1}
 compute χ_j^{t+1} 's weight by Equation.10
 put in χ_j^{t+1} into temporary list TL
 end
 end
 sort TL into round-robin list RL by the active node list
 repeat
 for each list item RL_j in RL
 remove the top particle and put it in L^{t+1}
 end
 until $|L^{t+1}| > ParticleCapacity$ or $RL = empty$
 end
 Output L^{T+1} as the final posterior distribution

Figure 18: Discrete Condensation (D-Condensation) for P-Net inference

what really matters in future explanations, which should have better impact on survival of the particles rather the pure probability. For example, if there is only slot left, a particle with a different *frontier* but lower probability should have higher priority to survive than a particle of the *frontier* that is already sampled by the already selected particles.

Inspired by this idea, we introduce the Round-Robin method to select particles for propagation. First, we generate a list of the different active nodes set, i.e. each item in the list is a set of active nodes such that at least one current particle has exactly that set of nodes as active nodes. Second, we sort the particles into that list by their active nodes set and rank them by probability. Third, we select the particle at the top of each list and remove them from the lists. The selection stops when all K particles are selected or the list is empty. That completes Discrete Condensation (D-Condensation), which is summarized in Figure.18.

As we can see, the Round-Robin selection is enabled by P-Net's unique notion of active vs inactive node state. Active nodes mark the progress of the explanation path. The

difference in frontier rather than the difference in history makes the explanation path conceptually different, and that constitutes a convenient way to differentiate the particles not only by their probability but also by their active/inactive attributes. By putting those particles together and prioritizing the particles with different *frontier*, we successfully avoid the over-clumping problem.

Furthermore, after such selection, particles with the different active nodes will survive despite their relatively low probability. Each of those particles represent a distinctive explanation. When the input is noisy, if only ranked by probability, the particle of the true explanation path may not always fall in the top K selection, even though it is No.1 at the end of the process. However that particle is usually the highest of all the particles that have the same active nodes. Through Round-Robin selection, it has a much better chance of survival through the noise and stands out at the end.

Meanwhile, the total number of particles that survives is directly controllable. When more computational resource are available, the bound K can simply be raised to increase the precision of the sampling and explore a larger space. K necessary for a successful inference depends upon how many different active node set exists, which in turn depends upon the complexity of the P-Net. Generally, for a single stream, the number of different active node sets are simply the number of nodes; for several pure parallel streams, it will be the multiplication of node number in each individual stream. The actual total number of different active node set can be obtained by traversing the P-Net conceptual model.

5.4 Classification of Noisy Input

In real world applications, people often make errors and deviate from the ideal activity described by the P-Net in the form of the insertion errors (unexpected elements), and deletion errors (skipped elements). These lead to three possible classifications for an observation sequence: (1) a perfect example, (2) an almost right example, a sequence that matches the majority of the activity definition but has a few deviation, and (3) a negative example.

P-Nets are capable of dealing with “insertion error” and “deletion errors”. The inference algorithm will ignore the input segment corresponding to “inserting error” that does not contribute to sequence interpretation with a penalty; P-Nets deal with “deletion errors” by taking unlikely observation data as the real observation to push through the missing node. This is called hallucination.

Hallucination can be detected by checking the history path. Given an activation interval $\langle s, d \rangle$ for node i in a history path, if the corresponding observation is more likely to be generated by an inactive node rather than an active node, this activation interval is labelled as hallucination:

$$\begin{cases} \text{node } i \text{ is hallucinated:} & \text{if } \prod_{s \leq t \leq s+d} P(o_t | R_i) < \prod_{s \leq t \leq s+d} P(o_t | \emptyset) \\ \text{node } i \text{ has real observation:} & \text{else} \end{cases} \quad (11)$$

In our set up, no real activation is shorter than 2 frames. Any interval length less than two is also labelled as hallucination.

If the best particle can not reach the final dummy node or reaches the final dummy node with a probability below a learned threshold, the sequence is classified as *wrong*. Otherwise, if a further check through the history of the particle reveals no hallucinations, then it is labelled as a *perfect* sequence. If hallucinations do exist, then the sequence is considered *almost right*. Thus, P-Nets can not only label the whole sequence in three categories, but also label each frame and automatically detect any missing nodes.

5.5 Complexity

The D-Condensation algorithm estimates the joint distribution by K particles. At each time step, for each particle, there are at most $O(2^J)$ subsequent particles, where J is the size of the largest cut set in the P-Net that separates the dummy *start* node from the dummy *end* node. There are $O(2^J K)$ new particles. For each particle, there is only one fixed set of parent states. We need to estimate the state for every node, each of which depends upon

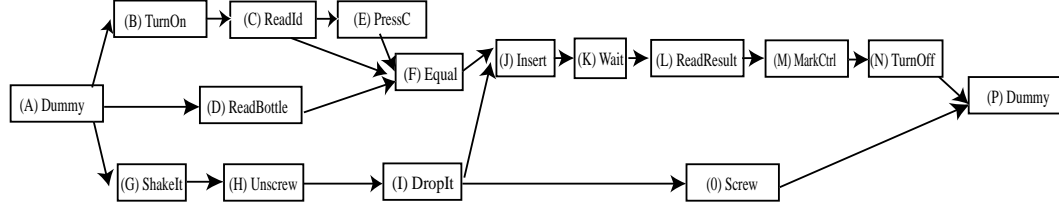


Figure 19: Conceptual diagram for a glucose meter calibration process

at most $O(E)$ parents. Therefore, for each particle, the computation is only $O(N \cdot E)$. To generate all candidate particles, there is $O(2^J K \cdot N \cdot E)$ computation. Sorting one candidate particle into list needs constant time to look at the state and a small sorting within each list item. Approximately, it is much less than sorting all particles together: $O(2^J K N E \log(2^J K N E))$

Therefore the total computation complexity is $O(T \cdot 2^J K N E \cdot \log(2^J K N E))$

The computation buffer for D-Condensation is K particles per time step and the maximal potential candidate $O(2^J K)$. To keep the history, total space is $O(KT + 2^J K)$.

5.6 Experiment: Glucose Meter Calibration Task

Discrete Condensation algorithm is tested on a glucose monitor calibration task.¹ The process is defined by the P-Net in Figure.19. The user interaction with the glucose meter represents a typical medical process in which an automated computer system that recognizes the user's activity can guide the user through the task. This domain is of current interest as we explore the application of technology to elder care.

We build a 16-node P-Net representation for the standard glucose meter calibration procedure. Then three subjects come in and perform a total of 21 perfect sequences, 10 missing_1_step sequences and 10 missing_6_steps sequences. For training purpose, we use six perfect sequences. The rest serve as the testing set.

As in the experiment in the previous chapter, the middle level output from the Bayesian network is quite poor, as the shared lower level detector library is too simple and has many

¹The work in this section is based on the tracking system built by Yan Huang

false alarms. But, the temporal constraints of the P-Net compensate for the noise and make the final labelling much better. An example comparing the noisy observations and the final labelling of the “insert” component is shown in Figure.20.

During our experiment, if we set the largest number particle to survive for the next iteration as $M = 300$, we find the largest number of subsequent particles that are ever generated by D-condensation is 1967 and the distinctive active states set is 238. This suggests the average 2^J in the most expensive round of iteration is $1967/238 \approx 8.3$. In other words, in the worst round, each particle generates on average eight subsequent states. Considering any node has a window of at least 50, which makes the possible state of a single node to be 1250, that is a small enough number as compared to the particle number needed to make standard Condensation work in such large space. D-condensation is very fast. The overall frame rate on a pre-generated observation data file is over 122 frames per second, more than sufficient for a real time surveillance system. The computation statistics are summarized in table 4.

Table 2: Overall Evaluation

Sequence Category	Total	Perfect	Almost Right	Negative
Training	6	100%	0%	0%
Perfect	15	100%	0%	0%
Missing One	10	20%	80% [†]	0%
Missing Six	10	0%	50% [‡]	50%

[†] All 8 claim missing that step; 2 of 8 claim missing an extra step; 1 claims missing extra 2.

[‡] 3 claim missing 5 nodes, 2 claim missing 6; all 5 at least claim 3 actual missing steps.

The final result is shown in Table.2. All the perfect sequences are labelled correctly. We label as “almost right” 8 out of 10 “missing_one_node” sequence; the other two are labelled as “perfect”. Comparing with the ground truth, we find the errors are caused by insertion errors in the vision module and the output of the insertion errors is statistically indistinguishable; we label five of “missing_6_steps” testing data as “totally wrong”, five as almost right while the system correctly identifies most of the missing steps.

Table 3: Labelling individual node

Individual Node	Overall Success †	Correct Positive ‡	Correct Negative *
B:TurnOn	0.9999	1.0000	0.9999
C:RdIdScreen	0.9901	0.9956	0.9897
D:RdIdSstrip	0.9893	0.9333	0.9909
E:PressC	0.9787	0.2344	0.9998
F:Equal	0.9847	0.9267	0.9908
G:Shakelt	0.9590	0.6003	0.9738
H:Unscrew	0.9563	0.5041	0.9857
I:DropIt	0.9827	0.8584	0.9941
J:Insert	0.9878	0.8643	0.9961
K:Wait	0.9964	0.9987	0.9958
L:ReadResult	0.9966	0.9847	0.9991
M:MarkCtrl	0.9983	0.9720	0.9993
N:TurnOff	0.9967	0.8997	0.9997
O:screw	0.9476	0.6629	0.9617
Average	0.9839	0.8709	0.9914

† Overall Success is the average of all nodes

‡ Correct Positive = number of correctly positively labelled frames over number of all positive label frames for node i

* Correct Negative = number of correctly negatively labelled frames over number of all negative label frames for node i

The result can also be checked by labelling each frame. For each frame, depending on whether 16 of the nodes are on or off, there are 16 of 0/1 labels. Compared with ground truth, we collect overall-correct-ratio as $[correct_positive + correct_negative]/[all_frames]$, correct-positive-ratio as $[correct_positive]/[all_positive]$ and correct-negative-ratio as $[correct_negative]/[all_negative]$. The statistics for the individual perfect type sequence and all sequence are available in Table.3. Though individual labelling ratios range quite a bit, the overall correct ratio for any sequence is very high ($> 96\%$), and the average correct-positive-ratio is 87.1%.

D-Condensation also provides us the ability to perform online labelling. As a P-Net sees evidence coming in, it performs online updating of the state distribution. Figure.21 shows snapshots of this process. As the video continues, the particles propagate through the P-Net. The marginal distribution on the nodes shows the concentration of the particles.

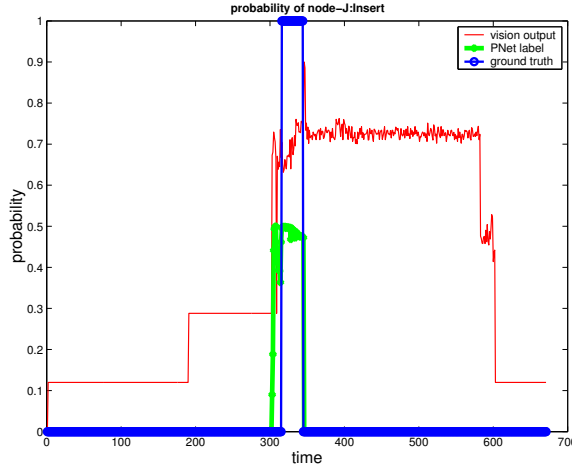


Figure 20: Comparison of PNet label with ground truth and lower level vision output.

We can see it matches well with the ground truth.

Table 4: Overall Computation Performance

Measure	Data
Sequence Length Range	[232, 928]
Average Speed (frames/sec)	122.7
Maximal No. of Distinctive Active Node Lists	238
Maximal Subsequent States	1967

5.7 Comparison with SCFG

To provide a comparison to P-Nets, we created a model of the blood glucose calibration task using a stochastic context-free grammar [38, 48, 44].² We associated one event in the grammar with every node in the P-Net. Events were detected by finding intervals of high probability in the corresponding low-level observation probability signal, $p(O_i)$. For each interval, an event was generated every 45 time steps and inserted into the event stream.

A stochastic grammar must explicitly represent all valid event orders. For this experiment, we enumerated all 1624 event sequences implicitly represented by the P-Net. They were encoded in a stochastic grammar as a single rule with many equally likely production alternatives.

²The work for this section is based on collaboration with David Minnen.

The stochastic parser found valid parses for all 21 of the correct sequences. We measured accuracy by calculating the percentage of symbols in the most likely parse that fell within the correct range according to the ground truth data. Over the 21 sequences, 62.8% of the symbols were within the correct range. We attribute this relatively low performance to the lack of duration models within the grammar, which causes the parser to be more susceptible to noisy events and to accept temporally unrealistic interpretations.

Computational complexity concerns made parsing the erroneous sequences difficult. Unlike a P-Net, recovering from deletion errors in a stochastic parser incurs an exponential penalty. Thus, parsing the missing-six data set was infeasible. The parser was able to find valid parses for all 10 missing-one sequences, but only after a restricted grammar (35 alternatives) was used and deletion recovery was limited to only one consecutive event hallucination.

Here in SCFG, we do not include a duration model on each non-terminal. Otherwise, either the probability on each grammar rule will not be static, or there will be exponentially more non-terminals to represent the duration. The second choice is totally impractical. The first choice may be possible. If we can augment non-terminal symbols in SCFG with a variable to represent the duration and make the grammar probability depend on those variables, SCFG may have a similar modelling ability as P-Net. But it needs further elaboration as defining non-static probability function is not an easy job. Without a duration model, all that SCFGs can model is an exponential distribution of activation interval and that is usually not the case in the daily activity.

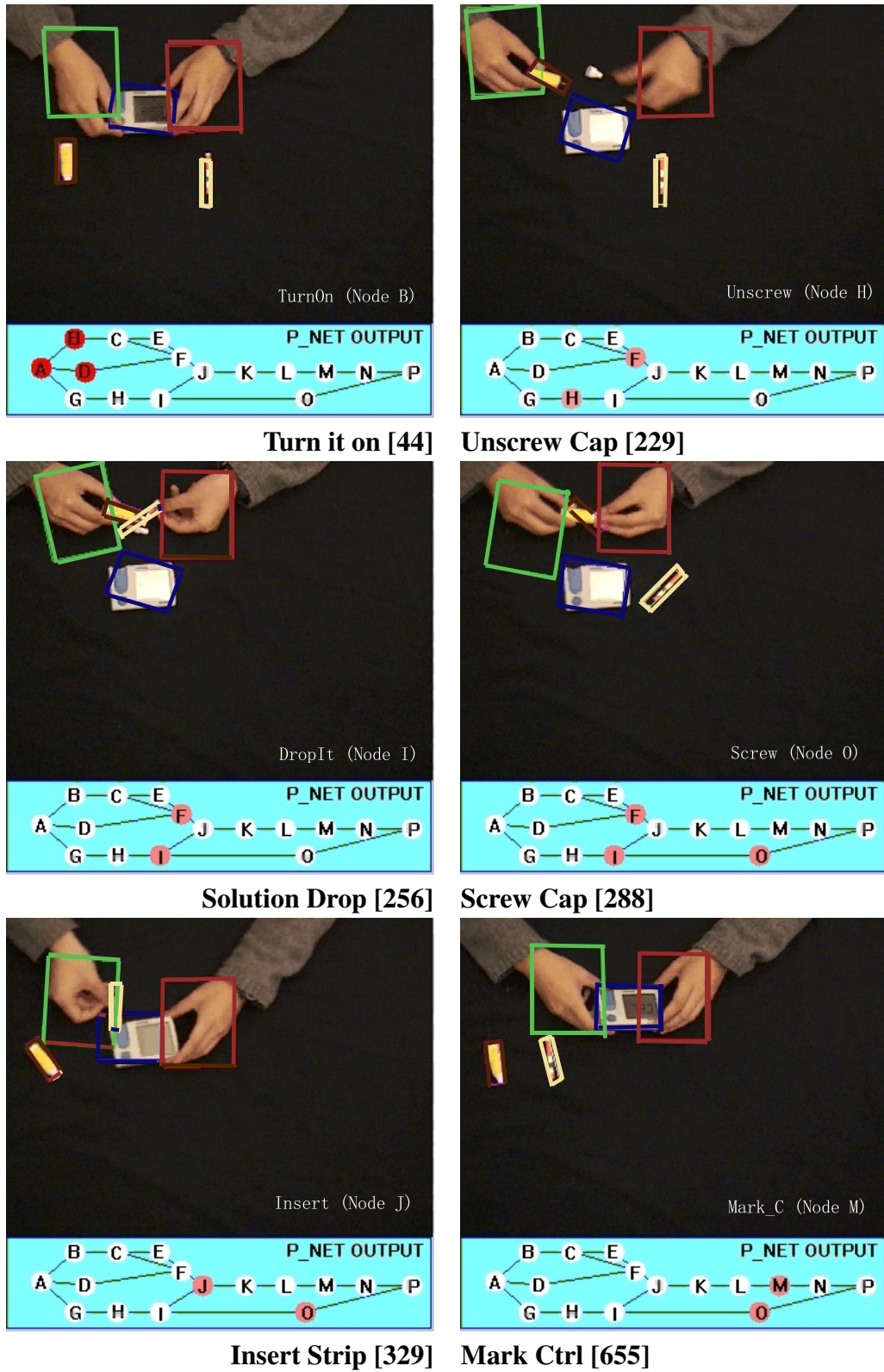


Figure 21: The online labelling of test sequences by P-Net. [] shows the frame number in the sequence. The color of the nodes in P-Net shows the marginal distribution that represents the P-Net belief on whether the corresponding motion is occurring. (Refer to Figure.9 for the motions represented by the nodes.)

CHAPTER VI

LEARNING

6.1 Network Parameter Learning

When P-Nets are designed for the first time, they are manually constructed. First, a conceptual model is drawn by an expert, then all parameters are specified to be consistent with the fully annotated training examples. We wish to automate this process as much as possible. Here our goal is not to come out with an unsupervised solution. Instead, we combine expert knowledge with the machine learning techniques to construct a P-Net.

It is hard to learn the causality relation in the activity domain. One major reason is the limited number of training sequences. Normal graphical model structure learning techniques work only when the training data fairly spreads out the potential state space so that the probabilistic relation can be estimated. But the training set we have is typically too small to learn such structure.

To simplify the problem and provide a solution viable with a real world data set, we further split the causal relation Φ into two parts: the topology part and the parameter part. The topology part describes under what condition a node can become active. The parameter part fits a model to precisely specify the topology part. The conceptual model represents the topology part of causal relation, which includes

1. what are the building blocks of the activity: a set of nodes
2. how to detect those motions: detectors for each nodes=
3. what is the logic condition that an interval can start: and/or combination of parent nodes

4. what is the observation model: a distribution model for the corresponding detector output when the node is active, another distribution model for the output when the node is inactive
5. what is the duration model: the distribution of the activation time

Experts will specify the topology of the above items (item 1-3) then the parameter part (item 4-5) will be extracted from the training sequences. As we usually do not have many training sequences, all models have to be simple and stable. We choose Gaussian for all distributions. We choose the noisy-and/noisy-or function to specify the activation model ϕ_3 (Equation.4): each link from a parent node has an independent Gaussian model to specify the probability given the interval between the parent termination and the intended child activation; the probabilities from all parent nodes are then integrated by the noisy-and/noisy-or function.

For example, if the logic condition is *either A or B finishes then C starts*, and the Gaussian model for link A to C is $G_a(x)$ and $G_b(x)$ for link B to c, then the activation model for C will combine $G_a(x)$ and $G_b(x)$ together by noisy-or function, which can expressed by the following segment function form conditioned on the state of A and B:

$$\phi_3(r_c = \langle t, 1 \rangle | r_a = \langle s_a, d_a \rangle, r_b = \langle s_b, d_b \rangle) = \begin{cases} G_a(t - s_a - d_a), & \text{if A finishes : } s_a + d_a = t - 1 \\ G_b(t - s_b - d_b), & \text{if B finishes : } s_b + d_b = t - 1 \\ G_a(t - s_a - d_a) + G_b(t - s_b - d_b) - G_a(t - s_a - d_a) * G_b(t - s_b - d_b), & \text{if both A and B finish : } s_a + d_a = t - 1 \cap s_b + d_b = t - 1 \\ 0, & \text{else} \end{cases}$$

There are two sets of model parameters to be learned from training data: the duration model and the observation model. Since we choose a Gaussian distribution for these models, there are only two scales per Gaussian—mean and covariance—to be estimated from the positive training examples. They can be easily retrieved once the explanation paths are

Given: fully labeled observation O_0 and its ground truth path L_0 ,
unlabelled positive observation set O_1, \dots, O_n ;
Initialize $PNet_0$ as the following:
Set Gaussian mean of the duration model in node i as the duration s_i for node i in L_0
Set Gaussian mean of the observation model according to O_0 and L_0^i
Set variance as a large enough constant For $u=1, \dots, U$
For $v=1, \dots, n$
 Obtain MAP path L_v^u on observation O_v from $PNet^u$ >
end
Fit $PNet^u$ on $\langle \{O_0, \dots, O_n\}, \{L_0, L_1^u, \dots, L_n^u\} \rangle$
break if $\sum_v P(L_v | O_v)$ is decreasing
end
Output: the best $PNet^u$

Figure 22: Learning the duration and observation model from the positive training set obtained. This suggests an EM approach: use the current P-Net to generate the best explanation path for each of the positive training examples and then use the explanation paths to estimate the models.

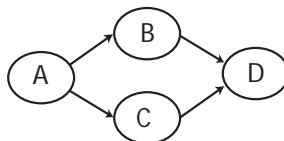
To obtain the initial position, we manually annotate one positive example. With the true explanation path and the observation matrix, the initial mean of each Gaussian can be easily obtained. The covariance of each Gaussian is set to a moderately large number so that it can converge to the correct model. The whole parameter learning process is illustrated in Figure.22.

The P-Nets in the experiments in the previous chapter are obtained in the above manner. The performance in those experiments proves the validity of this cooperative approach to construct a P-Net.

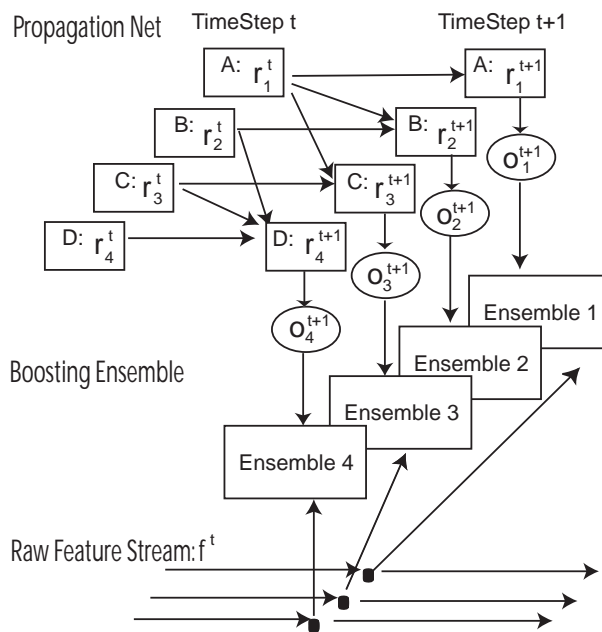
6.2 *Detector Learning*

6.2.1 Framework

Thus far, the detectors for the nodes in P-Net are constructed by hand. Some are manually coded and others are trained on the fully annotated data. This manual process is tedious and requires extensive domain expertise. On the other hand, purely unsupervised methods



(a) Key events (A/B/C/D) and their partial orders



(b) the Corresponding P-Net and boosted ensembles

Figure 23: Example of Propagation Net with the boosted event detectors.

(such as HMMs) can be trained automatically but yield models whose internal structure—the nodes—are difficult to interpret semantically. Here we compromise between the two methods by introducing a Semi-Supervised approach.

First, a manually structured P-Net is initialized from a small amount of fully annotated data, and then it is refined by an EM-based learning method in an unsupervised fashion. During node refinement (the M step) a boosting-based algorithm is employed to train the evidence detectors of individual nodes. Experiments on a variety of data types—vision and inertial measurements—in several tasks demonstrate the ability to learn from as few as one fully annotated example accompanied by a small number of positive but non-annotated training examples.

We intend to use the boosted stumps as the node detector. The general machine learning

paradigm of boosting [19] is robust to noise [22] and has proven useful in feature selection [70]. However, in the activity recognition domain, the temporal context is also important. Boosting the naive classifier on an individual feature vector without the context will not provide much help in recognizing a sequence. To address this problem, [62] proposes to use the immediate neighbor frames as the context, which only supports simple activity. In our approach, we need to deal with much longer temporal context. It is a natural choice to use a P-Net to describe the context and train a set of boosting ensembles together. We call this framework P-Net+Boosting. Figure.23 illustrates our approach of combining a priori defined P-Net structure with a boosting-based learning method to train the motion detectors.

Once the learning is finished, the recognition system works in the following way: for each node in P-Net there is a boosted detector; at each time step, the underlying perception system will provide a raw feature vector composed of raw information, e.g. the position and the velocity of the candidate objects; the boosted detector works on the raw feature vector and produces a confidence scale $o_i^t \in [0, 1]$ of whether the corresponding node is happening; the observation model of the corresponding node will generate the observation probability $P(o_i^t|r_i)$; the P-Net inference algorithm will find the Maximum a Posterior (MAP) path through P-Net that marks out the most likely coherent activations.

6.2.2 Semi-supervised Learning

One natural solution to the construction problem is to apply the standard EM algorithm as the following: (1) randomly select the initial position for P-Net, (2) use P-Net to label sequence while assuming all boosting detector output is neutral, (3) use the Maximum a Posterior (MAP) explanation path to set the ground truth to train the boosting ensemble, (4) use the MAP path + real boosting output to train P-Net, (5) find the MAP path on new boosting output and go back to step 3. A caveat is in order as there are at least two parts of this EM training that can exhibit uncontrolled behavior. First, EM training is

Given: fully labeled sequence SEQ_0
 and its ground truth path L_0 ,
 unlabelled positive sequence set SEQ_1, \dots, SEQ_n ;
 Initialize: Use $\langle SEQ_0, L_0 \rangle$ to train Detector Set $\{D_i^0\}$
 Use $\{D_i^0\}$ to compute P-Net input O_0^0 on SEQ_0
 Fit $PNet^0$ to $\langle O_0^0, L_0 \rangle$
 For $u=1, \dots, \text{maximal iteration}$
 For $v=1, \dots, n$
 Obtain MAP path L_v^u on sequence SEQ_v from $\langle PNet^u, \{D_i^{u-1}\} \rangle$
 end
 Use $\langle \{SEQ_0, \dots, SEQ_n\}, \{L_0, L_1^u, \dots, L_n^u\} \rangle$ to train Detector $\{D_i^u\}$
 Use $\{D_i^u\}$ to compute input $\{O_0^u, \dots, O_n^u\}$ on $\{SEQ_0, \dots, SEQ_n\}$
 Fit $PNet^u$ on $\langle \{O_0^u, \dots, O_n^u\}, \{L_0, L_1^u, \dots, L_n^u\} \rangle$
 break if total probability of $\{L_0, L_1^u, \dots, L_n^u\}$ is decreasing
 end
 Output: the best $\langle PNet^u, \{D_i^u\} \rangle$

Figure 24: Semi-Supervised Learning Algorithm to train P-Net and its underlying detectors in unified EM iterations.

very sensitive to the initial position conditions. Second, EM training is by nature greedy. Without annotation, it often accidentally locks onto a pattern that has no semantic meaning but seems to match the segregation.

Expert prior knowledge can alleviate these problems. By having one fully annotated positive sequence SEQ_0 for each category, the initial position can be obtained. As all frames have labels, the detector ensemble can be trained and the mean of the duration model can be set as the actual duration in SEQ_0 . The observation model can be trained on the output of the boosting ensemble with manual annotation as ground truth. In this way, we can obtain a good starting point close to human perception.

With a few more unannotated positive sequence SEQ_1, \dots, SEQ_n , we can continue the normal EM training. To avoid the second problem in EM, we will put SEQ_0 in the training set and take the human annotation as the MAP path for SEQ_0 , then we resume the normal EM training cycle. By involving the fully annotated SEQ_0 , we have an anchor point in the EM training. This anchor point will ensure the pattern that the detector converges to is not far from human perception. Therefore, the semi-supervised EM cycle becomes a controlled greedy search for the patterns that have better generalization ability than the initial position.

Consequently, we have a training algorithm that requires just a little more human effort than the unsupervised learning while keeping the advantage of a P-Net with clear meanings. The whole algorithm is illustrated in Figure.24.

In our learning process, only the positive sequences are used. That is because in most surveillance scenario, we only know what is supposed to be right, anything falls out of the scope is wrong. In that case, the objective is to build a generative model which can draw a fine boundary by itself. Using negative examples and discriminative training may bias the model toward the particle examples shown up in the training set which is not desirable. However, for each detectors, there are negative frames as well positive frames. How to use them effectively is presented next.

6.2.3 Contrast Boosting

In our model of activity, there can be multiple parallel streams of motion. Multiple nodes can be active at the same time, which makes the nodes non-exclusive. The objective of standard n -class boosting algorithm is to find the ensembles that always classify a feature point to one and only one of n class. In other words, the ideal result ensembles are mutually exclusive, which makes it inappropriate in our setup. One obvious solution is to train n independent 1_vs_n-1 ensembles by the standard boosting algorithm [19]. Such training ignores the fact that, though not strictly exclusive, the events are distinctive. The detectors for such events are preferred to be as different from each other as the real data warrants, which necessitates a mechanism to push the individual detector apart during training. In other words, we need to train the detectors together in a non-exclusive but repulsive fashion.

One way to do that is to train n independent 1_vs_n-1 ensembles by assigning smaller initial weights to the feature points of overlapped portion so that the ensemble is biased towards the unique owned portion. This can be easily implemented by standard Adaboost algorithm. The only augmentation is to assign the initial weights as the following: first, for each detector, assign a unit weight to the frames that are claimed in the MAP path; then

normalize the weights so that the sum of weights that belong to the same raw feature equals 1 if any detector claims it, otherwise 0; finally normalize the weights so that the sum of the weights that belongs to the same detector equals 1.

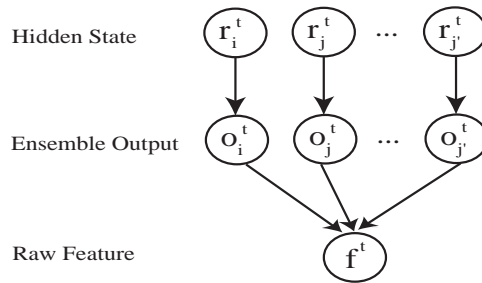
This linear assignment of initial weights, however, ignores the fact that P-Net provides a probability on how likely this particular detector output o_i^t can be observed given the hidden state r_i^t : $P(o_i^t|r_i^t)$. Multiple nodes claiming the same feature f^t form the network shown in Figure.25(a). A boosting ensemble simply maps f^t to o_i^t , approximately we can use $P(o_i^t|r_i^t)$ in substitution of $P(f^t|r_i^t)$, where the result is a Bayesian network shown in Figure.25(b). As we do not have enough information to estimate the joint conditional probability on node f^t , we will just assume it to be the noisy-or joint conditional probability: $P(f^t|r_i^t, r_j^t) = P(f^t|r_i^t) + P(f^t|r_j^t) - P(f^t|r_i^t) \cdot P(f^t|r_j^t)$. We further assume the equal prior: $P(r_i^t = active) = P(r_i^t = inactive)$. When computing the weight w_i^t on raw feature f^t for detector i , we can assume the rest of the hidden nodes are observed as the states in the MAP explanation path. Then computing $P(r_i^t|f^t, r_j^t, j \neq i)$ is easy. This posterior probability can be used as the initial weight for Adaboost training w_i , shown in Equation.12.

$$\begin{aligned} \frac{P(r_i^t|f^t, r_j^t, j \neq i)}{P(\bar{r}_i^t|f^t, r_j^t, j \neq i)} &= \frac{P(f^t|r_i^t, r_j^t)}{P(f^t|\bar{r}_i^t, r_j^t)} \\ &= \frac{P(f^t|r_i^t) + P(f^t|r_j^t) - P(f^t|r_i^t) * P(f^t|r_j^t)}{P(f^t|\bar{r}_i^t) + P(f^t|r_j^t) - P(f^t|\bar{r}_i^t) * P(f^t|r_j^t)} \end{aligned}$$

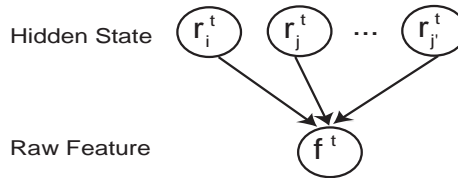
Since $P(r_i^t|f^t, r_j^t) + P(\bar{r}_i^t|f^t, r_j^t) = 1$, we have

$$w_i \equiv P(r_i^t|f^t, r_j^t) = \frac{P(o_i^t|r_i^t) + P(o_j^t|r_j^t) - P(o_i^t|r_i^t)P(o_j^t|r_j^t)}{(P(o_i^t|r_i^t) + P(o_i^t|\bar{r}_i^t))(1 - P(o_j^t|r_j^t)) + 2P(o_j^t|r_j^t)} \quad (12)$$

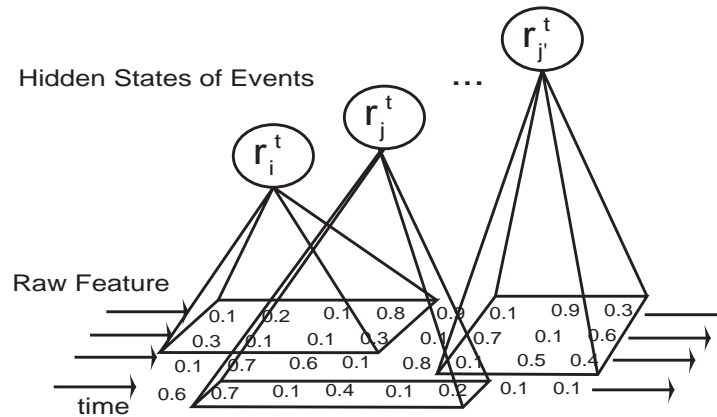
Boosting is proven to be useful in feature selection [70]. If the naive weak classifier works on a single dimension, e.g. the decision stump, the first few levels of the boosting ensemble will point out the most discriminative dimensions in classification. This idea can be borrowed and applied to our setup: different events can overlap in time, but the relevant feature dimensions shall be different. That is to force the detectors to pick up different feature dimensions if they overlap in time (illustrated in Figure.25(c)). This idea can be



(a) Multiple hidden nodes claiming the same raw feature



(b) Ignoring the mapping functionality of the ensemble, (a) becomes a Bayesian network.



(c) Multiple hidden nodes claims different dimensions of the same raw feature.

Figure 25: Example of P-Net with the boosted ensemble as event detectors

Given: $(f_i, g_i), \dots, (f_m, g_m); f_i \in \mathcal{X}, g_i \in \{-1, +1\}$

Initialize $w_{1i} = 1/m$

For $t=1, \dots, T$

- Train weak learner using distribution $W_t = \{w_{ti}\}$
- Get weak hypothesis $h_t : f \rightarrow R$
- Choose $\beta = \frac{1}{2} \ln\left(\frac{1+\epsilon}{1-\epsilon}\right)$
 where $\epsilon = \sum_i w_{ti} [h_t(f_i) \neq g_i]$
- Update: $w_{t+1,i} = w_{ti} \exp(-\beta_t g_i h_t(f_i)) / J_t$
 where J_t is a normalization factor

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^T \beta_t h_t(x)\right)$$

Figure 26: A generalized version of ADABOOST and corresponding α from [19]

Given: training set $\{ \langle f_i, g_i, w_i \rangle \}$,

where $f_i = [f_{i1}, \dots, f_{iN}]$ is $1 \times N$ observation vector,

g_i is the ground truth $\{-1, 1\}$,

$w_i = [w_{i1}, \dots, w_{iN}]$ is corresponding weight

For training circle $t=1, \dots, T$

For each dimension $v=1, \dots, N$

Set $\{p_i = w_{iv} / \sum_{j=1}^N w_{jv}\}$

Train 1D classifier on $\{ \langle f_{iv}, g_i, p_i \rangle \}$;

Get hypothesis $h_v^t : f \rightarrow [-1, 1]$;

Calculate error of $h_v^t : \epsilon_v^t = \sum_{i=1}^M p_i |h_v^t(f_{iv}) - g_i| / 2$

end

Select the best h^t from $\{h_1^t, \dots, h_N^t\}$ with lowest ϵ_v^t .

For each dimension $v=1, \dots, N$

Set $p_i = w_{iv} / \sum_{j=1}^N w_{jv}$

Calculate error of $h^t : \bar{\epsilon}_v^t = \sum_{i=1}^M p_i |h^t(f_{iv}) - g_i| / 2$

Set $\beta_v^t = \frac{1}{2} \ln\left(\frac{1+\bar{\epsilon}_v^t}{1-\bar{\epsilon}_v^t}\right)$

Update the weight: $\{w_{iv} = w_{iv} \exp(-\beta_v^t g_i h^t(f_{iv})) / J_t\}$

end

end

Output: the final hypothesis

$$h_{final}(f) = \text{sign}\left(\sum_{t=1}^T \beta^t h^t(f)\right)$$

Figure 27: Contrast Boosting with weight on each dimension

implemented by assigning an weight for each dimension of each feature vector. The initial weight can be computed by a computation similar to Equation.12. The only difference is to add an extra subscript to denote the different dimension for each symbol.

The standard Adaboost training, illustrated in Figure.26, is altered accordingly to take into account the weight on each dimension as opposed to just one weight per feature point. The modification is simple. Since we use weak classifiers that work on only one dimension, we can iterate on all dimensions using the corresponding weight vector and find the best weak classifier on one particular dimension. We then update the weight for each dimension independently as we would in normal Adaboost. The overall Contrast Boosting training is shown in Figure.27.

Plugging the Contrast Boosting method into semi-supervised learning algorithm, we have completed an temporal sequence modelling and recognition system that is easy to construct.

6.3 Experiments

We apply the semi-supervised learning algorithm to three data sets obtained from vision and sensor platforms: 1) a vision based indoor activity data set, 2) a vision based glucose monitor calibration data set, 3) a sensor based simple weight-lifting exercise data set ¹.

The indoor activity data set consists of two types of activities. In type 1, “calling”, one walks in, sits on the chair, makes a phone call then walks out; in type 2, “reading”, one walks in, sits on the chair, reads a book then walks out. We take 16 calling sequences and 15 reading sequences. The video streams are processed by a simple color/shape based vision tracking system to obtain the 2D trajectories of hands, a head and an object. Variations in viewing condition and the non-rigidity of the objects results in trajectories with significant noise. The key events and their partial order are shown in Figure.28.

As described in previous chapter, the glucose monitor calibration data set has only

¹Thanks David Minnen for providing the data set

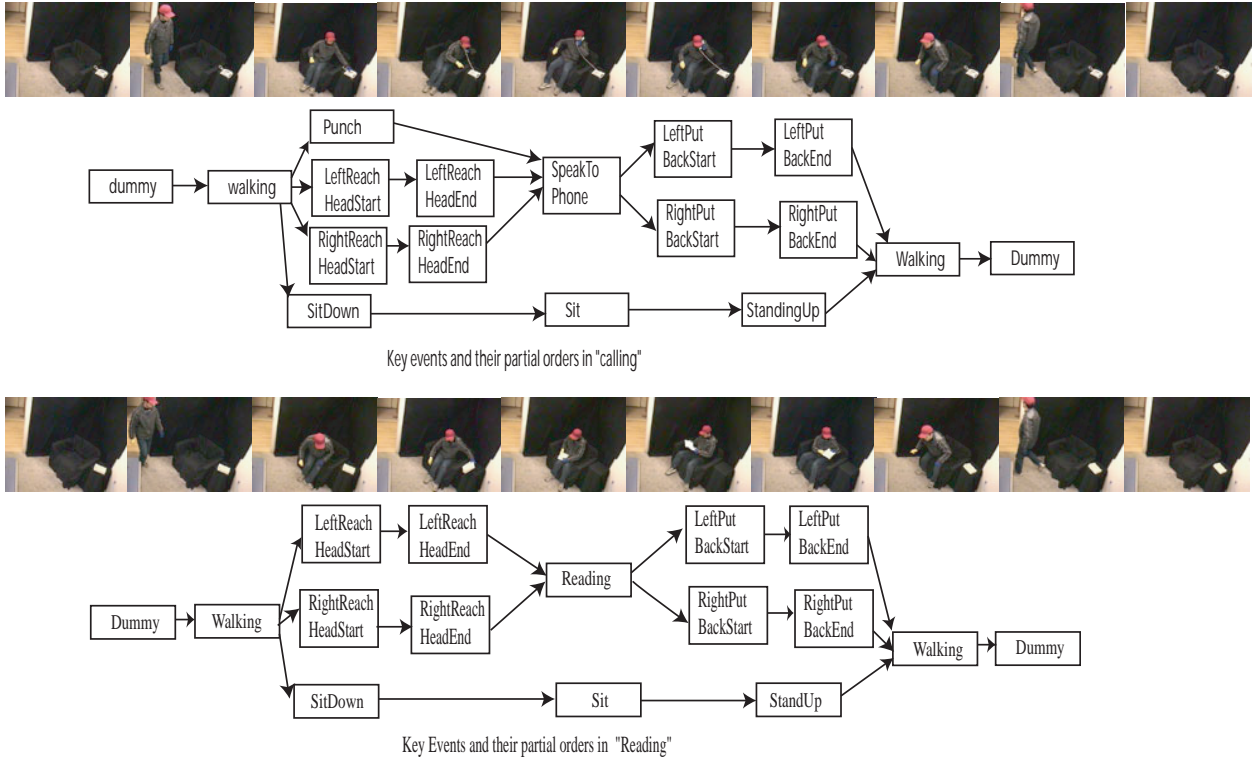


Figure 28: Vision tracked indoor activities: “reading” and “calling”.

one type of activity: calibrating the glucose by a user. The process involves 14 steps, whose semantics imply only a partial ordering. The data set is designed to detect anomaly. We recorded 21 correctly performed sequences, 10 missing-one-step sequences and 10 missing-six-steps sequences. The video is processed by a vision based tracking system which provides 2D trajectories of hands, a glucose meter, a test stripe and a test bottle. The tracking result is fairly stable, but because all events happen in a fairly close environment and the hands are deformable, finding the key event from just 2D trajectories is not easy. The key events and their partial order are shown in Figure.29.

The exercise data set has six types of simple dumbbell exercises, each with a few example sequences, i.e., 72 flat-curl sequences, 72 shoulder-extension sequences, 48 back sequences, 36 twist-curl sequences, 35 shoulder-press sequences and 60 tricep sequences. A 3D accelerometer, a 3D gyroscope and a 3D magnetometer sensor are attached to the hand when the exercise is performed. The continuous 3D sensor readings are recorded as the raw observation vector.

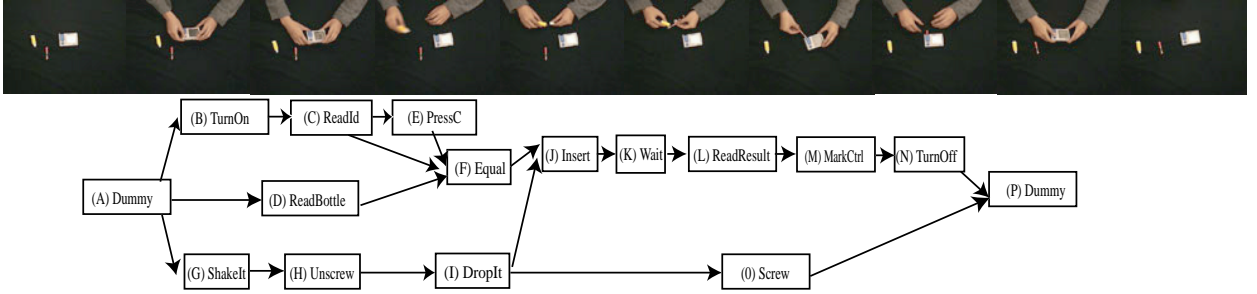


Figure 29: Vision tracked glucose monitor calibration process.

We choose to compare our system with the most widely accepted HMM, mainly because both methods are trained with little human interference once the topology is decided. We understand there are many other graphical models more powerful than HMM, but those models all require either hand-crafted detectors or large annotated training set, which is exactly the opposite of our purpose.

As observed in [62], an event can be learned by the feature vector built upon the trajectories. In the vision system, the trajectory is composed of 2D positions; in the sensor data set, the trajectory is of 3D vectors. Based on these trajectories, we construct the raw feature vector in an uniform style with the following composition: the raw position of trajectory, the distance between each pair of trajectories, the first moment of a trajectory, the second moment of a trajectory. These raw features are computed before training.

6.3.1 Classification

Classification is the basic requirement for any recognition system. We will train one PNet+Boosting per category on the training set. On the testing set, the classification result is the category whose MAP path has the highest probability normalized by the path length. As a comparison, HMM will also be trained on the same set, and the probability of the Viterbi path is used for classification. To bias these results as little as possible, we experimented with a variety of HMM topologies, settling upon the one that yields the best results in cross-validation. The best topology was a left-right, skip-one ahead model, which is used in the following experiments. The total number of nodes in HMM is also searched

to reach the best performance which happens to be 7.

We test classification ability on the indoor activity data set of “calling” and “reading”. First K sequences are chosen from each category as the training set while the rest are used as a testing set. The first training sequence in each category is annotated and provided to our system as SEQ_0 while the remaining training sequences are used as the unlabelled positive examples. We vary the training set size K to see how the system benefits from more training data. Due to the limited number of total sequences, we perform 10 fold experiments. The testing results are averaged as the overall result. Since random guessing can reach 50% accuracy, we use the classification ratio as the indicator, which is defined as $(\text{No.of.Correct}-\text{No.of.Wrong})/\text{Total}$. The results are shown in Figure.30. As we can see, for any number of training sequences, our proposed system is superior to HMM and quickly becomes 100% accurate as the training set becomes large enough.

6.3.2 Anomaly Detection

A P-Net can also detect and absorb missing observations and events, as it has probabilistic observation models. Another ability of P-Net is that it knows whether the explanation path reaches the ending node which represents the activity has finished. Therefore, a totally wrong sequence will have a much lower MAP probability that may even not reach the ending node, while a MAP path for an almost-right sequence will reach the end but with a lower probability. In this way, our system can distinguish between a right sequence, an almost-right sequence and a wrong sequence. Since an anomaly cannot, by definition, be collected ahead of time, only one HMM can be trained on the positive sequences. To evaluate how well the HMM can detect anomalies, we pick the best possible threshold to classify right sequences versus the rest.

We use the indoor activity data set to compare anomaly detection capabilities. We vary the training set size as described before. The results are shown in Table.5. We can notice our system is much better than HMM, plus the our performance improves as more training

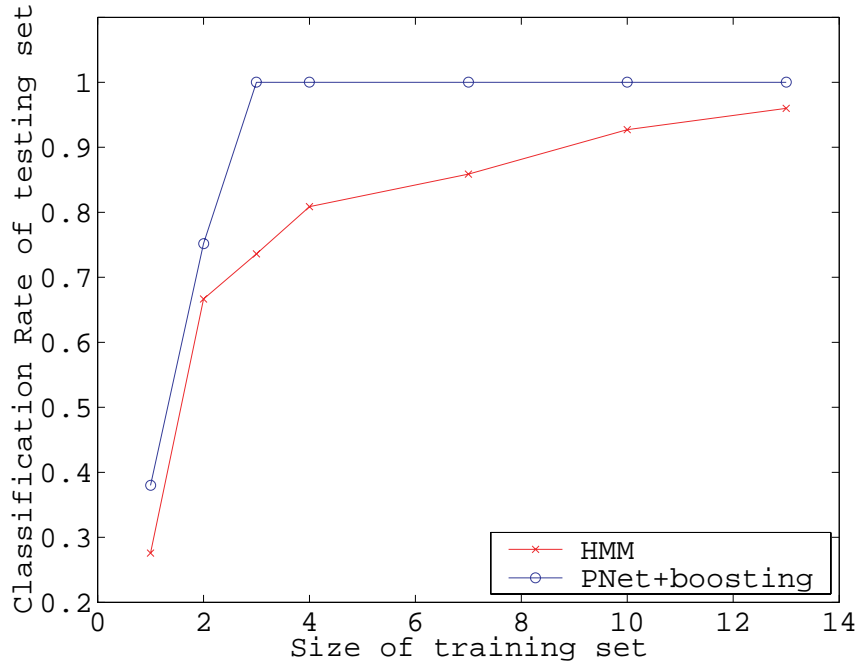


Figure 30: Classification results on Indoor activity dataset, Classification Rate = (No.of.Correct-No.of.Wrong)/Total

sequences are provided and is almost perfect when there are merely six training sequences per category.

6.3.3 Individual Event Detection

One of the design goals of our system is to keep the semantic interpretation of an individual node and detector. For each testing sequence, we can compare the labels of the MAP path generated by the correct P-Net with the ground truth. The comparison can be shown in three ratios: 1) correct positive = the number of true positive labels/total positive ground truth labels; 2)correct negative = the number of true negative labels/total negative labels; 3) overall success=the number of correct labels/sequence length. In the experiment, the performance of an individual detector as well as the performance of whole detector sets are recorded.

Table.6(a) shows the labelling ability on the indoor activity data set when seven training sequences are used. We notice the labelling ability is well above 90%. Considering there is no specially designed feature in the raw feature set, these results are extraordinarily

Table 5: Abnormity detection results on the glucose calibration dataset

	training size=1	training size=3	training size=5	training size=7
Classification	0.6750	0.7368	0.7500	0.7655

(a)HMM Classification Rate with best possible threshold

	detected as		
	perfect	almost right	wrong
perfect seq	0.4286	0.5714	0
miss-one-step seq	0.3000	0.7000	0
miss-6-step seq	0.4000	0	0.6000

(b)PNet+boosting detection result when training set size is 1

	detected as		
	perfect	almost right	wrong
perfect seq	0.6667	0.2857	0.0476
miss-one-step seq	0.5000	0.5000	0
miss-6-step seq	0.4000	0.1000	0.5000

(c)PNet+boosting detection result when training set size is 3

	Detected as		
	perfect	almost right	wrong
perfect seq	0.8571	0.1429	0
miss-one-step seq	0.8000	0.1000	0.1000
miss-6-step seq	0.1000	0	0.9000

(d)PNet+boosting detection result when training set size is 6

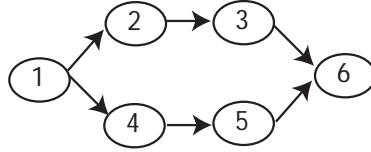


Figure 31: As there is no obvious structure in the exercise data set, we select a topology that has two sub-streams to start the training.

encouraging. Table.6(b) shows the labelling ability on the glucose calibration data set when six training sequences are used. It is less perfect in finding the positive labels for a short interval, e.g. “Equal”, “MarkCtrl”, but the overall labelling is effective.

6.3.4 Unknown Topology

When the activity is too simple, there is hardly any human pre-definable event and topology. That reduces our semi-supervised learning algorithm to a mostly unsupervised learning algorithm. This is also the situation when expert knowledge is not available. The comparison between our system and an HMM will show the baseline ability of our system to model temporal sequences in a blind situation. To get the algorithm running, we choose a left-right topology (Figure.31) and skip the training steps that involve the fully labeled sequence. A PNet+boosting classification system can still be constructed by the simplified version of Figure.24.

The sequences in exercise data set are simple and short (<100 frames). There is not much internal structure, which makes it the right testing set for our purpose. Besides, it shows the classification ability on multiple classes. Varying the training set by 10, 20 and 30 sequences per each category while keeping the rest as the testing set, we obtain the results shown in Table.7. Notice our system is still superior to HMM. This proves the P-Net+boosting framework is at least no worse than HMM even when there is no prior expert input.

Table 6: Result of detecting individual nodes in indoor activity data set and glucose activity data set

Individual Node	Correct Positive †	Correct Negative ‡	Overall Success *
walking	0.9989	0.9675	0.9717
Punch	0.3530	0.9922	0.9656
LeftReachStart	0.4983	0.9761	0.9650
RightReachStart	0.8675	0.9758	0.9751
RightReachEnd	0.4493	0.9828	0.9771
LeftReachEnd	0.3179	0.9900	0.9811
SpeakToPhone	0.9666	0.9543	0.9573
LeftPutStart	0.4180	0.9751	0.9698
LeftPutEnd	0.4784	0.9851	0.9761
RightPutStart	0.1837	0.9645	0.9586
RightPutEnd	0.0275	0.9883	0.9802
SitDown	0.2644	0.9880	0.9630
Sit	0.9795	0.9492	0.9700
StandUp	0.6068	0.9905	0.9765
WalkingBack	0.8869	0.9972	0.9855
Reading	0.9902	0.9813	0.9841
Average	0.9048	0.9815	0.9739

(a) Event detection in the indoor activity dataset

Individual Node	Correct Positive †	Correct Negative ‡	Overall Success *
TurnOn	0.3571	0.9915	0.9874
ReadId	0.9646	0.8379	0.8441
ReadBottle	0.6144	0.9917	0.9813
PressC	0.2795	0.9879	0.9694
Equal	0.0939	0.9953	0.9284
ShakeIt	0.3009	0.9536	0.9417
Unscrew	0.2971	0.9844	0.9527
DropIt	0.5843	0.9726	0.9487
Insert	0.8942	0.9115	0.9106
Wait	0.9747	0.9755	0.9753
ReadResult	0.7521	0.9973	0.9719
MarkCtrl	0.1733	0.9972	0.9639
TurnOff	0.2128	0.9976	0.9733
Screw	0.5876	0.9281	0.9094
Average	0.6631	0.9679	0.9502

(b) Event detection in the glucose calibration dataset

† Overall Success is the average of all nodes.

‡ Correct Positive: number of correctly labeled positive frames over number of all positive frames for node i

* Correct Negative: number of correctly labeled negative frames over number of all negative frames for node i

Table 7: Classification ability on the exercise data set

	training size=10	training size=20	training size=30
HMM	0.7681	0.8462	0.8473
PNet+Boosting	0.7795	0.8881	0.9310

CHAPTER VII

EXTENSION AND OPEN QUESTIONS

7.1 *Simulation Machine*

In the activity recognition domain, one major problem is the shortage of annotated data. A typical data set includes at most a few dozen examples per activity type, and often there is no ground truth for each activity component at every time step. This shortage prevents a detailed analysis on algorithm ability. Though in the real world an algorithm is required to work on a small training set, a large full annotated data set may provide many accurate statistics to evaluate the performance. A simulation machine is justified to generate an unlimited number of annotated examples.

As a generative model, P-Net is capable of describing a complex activity, and that makes building a simulation machine upon a P-Net viable. Generally, we modify the D-Condensation algorithm to explore a P-Net without observation vectors and collect particles that reach the dummy end node. For each of these particles, one positive path is obtained from the particle history information. This path represents which node is active at each individual time step. The generative observation model is then sampled to obtain the observation vector.

In the modified D-Condensation process, there is no influence from the observation model. The duration model and the interlink model control the particle likelihood. Therefore, the weight for new particle q at time step $t + 1$ who is derived from particle k at time step t can be obtained from the following equation:

$$Weight_q = Weight_k \cdot \prod_i P(r_i^{t+1} | r_{jk}^t, \forall j) \quad (13)$$

In the original round-robin selection in D-Condensation, we aggregate the particles by

```

Given: P-Net P
Initialize: Put particle  $\langle s = 0, d = 0 \rangle$  at dummy start node
For t=1 to Maximal activity length T
    Find all possible subsequent particles in P
    Update the particle weight according to Equation.13
    sort the particles into round-robin list
    for each list item
        select random particles according to particle weight
    end
    keep the particle that reaches the dummy end node in  $\Omega$  end
for each particle in  $\Omega$ 
    retrieve the history explanation path  $\zeta$ 
    sample  $\zeta$  to generate the observation path and put it in  $\Upsilon$ 
end
Output: the sample observation sequence set:  $\Upsilon$ 

```

Figure 32: P-Net based simulation machine

their active nodes into a list so that each list item is a set of particles of the same active nodes sorted by their probability. We always select the particles from the top of the each list item to find the most probable explanation path. This, however, can not keep the distribution in the sampling process. Instead, within each round of selection, for each list item, we select a particle by its probability. The rest of the D-Condensation remains the same. The simulation machine hence is constructed as shown in Figure.32. The generated examples successfully preserve the marginal duration distribution on the nodes and the links. With this tool, we can easily generate thousands of annotated examples.

7.2 Hierarchical Definition

As observed in [35], activities in a similar setup usually consist of a limited number of components. Components available directly from perception form the most basic activity vocabulary. Human tends to group those components together to become larger components. If those larger parts are used on multiple occasions and become mutually understandable building components, they too become part of the vocabulary. Such a hierarchy is standard in the natural language and makes the description clear and concise. P-Net is a

description tool for activity. As a bridge between versatile natural human description and precise computation model, it must be able to translate the hierarchical definition of components in human concept into a precise model, thereby directly leading to the demand for a hierarchical definition in P-Nets.

A hierarchical conceptual model of P-Net can be borrowed directly from Hierarchical HMM. There are two kinds of nodes: “production node” and “internal node”. Production nodes are the terminal nodes that can generate observations. Internal nodes are like non-terminal nodes that incur a sub-level P-Net. When a token is passed onto an internal node, it is simultaneously transferred to the dummy starting node of the sub-level P-Net and an extra propagation is initiated to move the token out of the dummy node. When the sub-level P-Net reaches the end dummy node, a simultaneous vertical transfer will move the token back to the parent internal node and mark the parent node as finished so that the subsequent node can be activated immediately. In this way, tokens are still guaranteed to reside only on the production nodes while the observation model and corresponding probability computation is unaffected.

Here, we implicitly place an extra constraint on any sub-level P-Net: there is only one entrance point and one exit point. This constraint has two impacts: first, once the sub-level P-Net is activated its progress is only controlled by its internal constraint and there is no place for an external influence, second, this sub-level P-Net can only influence others by a single exit point. Thus it rules out the possibility to coordinate between elements of two sub-level P-Nets. This is consistent with our selection of elementary units to guarantee the autonomous and continuous attribute of motion-node. We believe this constraint is necessary for conceptual clarity as well as implementation efficiency. Since every P-Net already has one dummy start node and one dummy ending node, the above constraint does not incur extra work to construct the conceptual model except to name each P-Net so that they can be referenced and reused in the following construction.

Unfortunately, the hierarchy in conceptual model can not be directly translated into

the computational model. Both the state computation in LMSA and the particle selection process of D-Condensation require global comparison, and consequently requires each sub-level P-Net to be open rather than just a black-box node in the conceptual model. There are two options to extend the computational model for hierarchical P-Nets:

1. We can flatten the hierarchical P-Net by duplicating any sub-level P-Net while attaching extra tags to ensure parameter binding between the duplicated nodes.
2. We keep the hierarchy in the computational model but augment each token/state with a stack to keep trace of where it comes from.

The second option maintains the hierarchy in computational model Figure.33(c). Each sub-level P-Net has one extra link coming from the corresponding high-level node and another link going to the node. The duration model on the vertical link will spike at 0 so that all tokens are immediately transferred with no cost from the high level node to the low level dummy start node and from the end node back to high level node. Similar hierarchical graphical model can be found in Kevin Murphy's thesis work [51]. All the states in sub-level P-Net must be augmented with a stack to keep the trace of the corresponding node at the high level. The inference algorithms are changed to distinguish the particle on the same node but with different stack information. The training process is simple as all duplications leads to the only copy of nodes and the parameter binding in training process is realized automatically.

The first option, as illustrated in Figure.33(b), uses a conceptual model as buffer so that the hierarchical definition exists only in the user interface and the conceptual model. We need a translation process to map between conceptual and computational model. Each node in the high-level P-Net that corresponds to a sub-level P-Net will be replaced by the sub-level P-Net itself; all links that point to the node will now point to the dummy starting node; all links that come from the node will now come from the dummy ending node; all parameters within the sub-level P-Net will be augmented with a tag to mark the duplication;

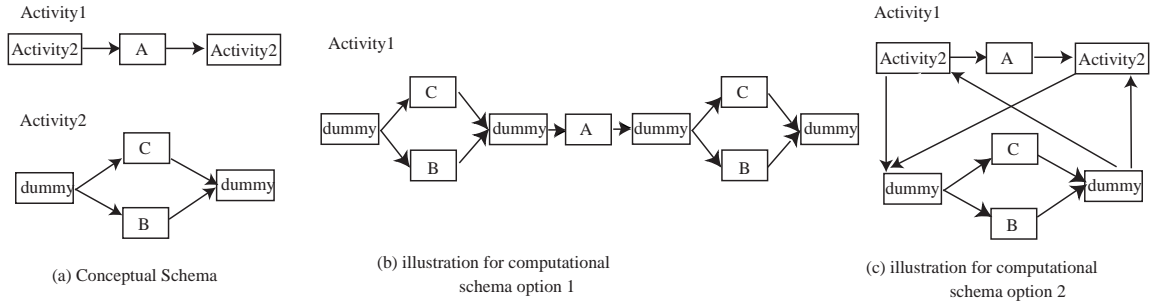


Figure 33: Hierarchical definition of P-Net

the rest of the model remains intact.

Within the computational model, the inference algorithms (LMSA and D-Condensation) are unaffected. There is no overhead computation to maintain the hierarchical information, but the training algorithm will become complex. In each explanation path, there may be multiple occurrences of the same sub-level P-Net. Those occurrences can be discovered by checking the attached tags, and they shall be pooled together to estimate the parameters.

We implement the first option as our target is to facilitate real time recognition. A simple compiling interface is constructed to handle the hierarchical conceptual model. In this way, we trade the higher training complexity for less recognition complexity so that the computational performance is unaffected and the specification within the construction process becomes easier.

7.3 Open Questions

There are still a few unsolved questions about how to use P-Net in activity recognition, namely, structure learning, classification of anomaly versus outlier and interleaved activities. The author has some ideas but yet to have a full solution. These three questions are general to activity modelling and recognition. There are discussed in this section not only as the potential improvement of P-Net but also as the inspiration for possible new activity models.

7.3.1 Structure Learning

In the activity recognition domain, the ideal situation will be the following: for any complex real world problem, given a small set of labelled positive/negative training examples, the system is able to construct the representation and use it to classify the unknown example and label its individual steps. Once a list of components are specified, the causal/topology relation between those components can be extracted to form a generative model of activity. This corresponds to the question : to learn the structure relation and build the conceptual model for P-Net.

There are some research on learning various structure models. Major literature falls in learning context free grammar and learning graphical models.

The inside-outside algorithm is the primary learning method for context free grammar. Originally, inside-outside algorithm is used to estimate the probability of each production rule given a large corpse and a set of grammars [3]. It is augmented in [41] to find the grammars given the terminal symbols and the corpse.

As described in [41], the Inside-Outside algorithm works on the Chomsky Normal Form. A Chomsky Normal Form (CNF) is a context free grammar whose rules are of the following form:

$$i \rightarrow jk \text{ or } i \rightarrow m$$

where i,j,k are unique non-terminal symbols and m are terminal symbol. It is proven any context-free grammar can be reduced to CNF [10]. A stochastic CNF is described by two matrices A and B, where:

$$a[i, j, k] = P(i \rightarrow jk|G)$$

$$b[i, m] = P(i \rightarrow m|G)$$

$$\sum_{j,k} a[i, j, k] + \sum_m b[i, m] = 1$$

$a[i,j,k]$ is the probability of non-terminal i will generate the pair of non-terminal symbols j and k ; $b[i,m]$ is the probability that non-terminal symbol i will generate a terminal symbol

m.

Similar to forward and backward probability in Baum-Welch algorithm, the inner probability e and outer probability f are defined to estimate matrix A and B , where

$$e(s, t, i) = P(i \Rightarrow O(s) \dots O(t) | G)$$

$$f(s, t, i) = P(S \Rightarrow O(1) \dots O(s-1), i, O(t+1) \dots O(T) | G)$$

. Once the parsing tree is established, e can be iteratively computed bottom-up from A and B ; then f can be computed top-down from e . Once e and f are obtained upon existing A and B , the newer rule probability can be estimated as

$$\begin{aligned} \hat{a}[i, j, k] &= P(i \rightarrow jk | i \text{ used}) = \frac{P(i \rightarrow jk, i \text{ used})}{P(i \text{ used})} \\ &= \frac{\frac{1}{P} \sum_{s=1}^{T-1} \sum_{t=s+1}^T \sum_{r=s}^{t-1} a[i, j, k] e(s, r, j) e(r+1, t, k) f(s, t, i)}{\frac{1}{P} \sum_{s=1}^T \sum_{t=s}^T e(s, t, i) f(s, t, i)} \\ \hat{b}[i, m] &= P(i \rightarrow m | i \text{ used}) = \frac{P(i \rightarrow m, i \text{ used})}{P(i \text{ used})} \\ &= \frac{\frac{1}{P} \sum_{t: O(t)=m} e(t, t, i) f(t, t, i)}{\frac{1}{P} \sum_{s=1}^T \sum_{t=s}^T e(s, t, i) f(s, t, i)} \end{aligned}$$

To construct the CNF out of a training corpse, first, the probability matrix A and B are initialized; second, the standard inside-outside algorithm can estimate the newer probabilities \hat{A} and \hat{B} given the corpse; third, rules with too low probability is trimmed and the new set of rules are used for the next round of iteration.

The complexity of generic inside-outside algorithm is $O(n^3T)$ where n is the number of symbols and T is the iteration time. Comparing with $O(n^2T)$ in Baum-Welch algorithm, this is very expensive for any moderate size grammar. There are a few work to constrain the iteration process. Pereira etc suggested to use bracketed corpse to constrain how parsing tree and therefore limits the choice of potential rules and non-terminal symbols [55]; Clark etc suggested to use prefix and suffix distribution to discover non-terminal and use Minimum Description Length to guide the selection of rules [12]. Yet the complexity is still on magnitude of $O(n^3)$. And the initial starting point is very sensitive to the final result. There is no guarantee that the result will be humanly understandable. Furthermore, the above

algorithms are all based noise free data. So far there is no success story on how to handle errors in training set.

Structure learning in graphical model is another domain that receives extensive attention. One of the pioneer structure learning algorithm in Bayesian network is K2 algorithm [14]. K2 algorithm constructs a bayesian network from a complete data set. A complete data set has no hidden variable and no missing value. All random variables are ordered. The parent nodes are assumed to be the nodes with lower order. The network is augmented from zero node by including next lowest order node. All possible parent links are explored and the structure with the highest joint probability of data set and structure is kept. Under a few assumptions, K2 reduces the joint probability $P(B, D)$ analytically, (B is structure, D is the data set). The detailed assumptions can be looked up in [14], while the most important ones are data independence, complete data, model independence and conjugate prior. Under slightly different assumption, a few other structure evaluation criterions are suggested, e.g. BD in [31] and BIC in [60].

To handle data with missing values and hidden variables, Friedman introduces Structural EM learning algorithm (SEM) in [23, 24]. SEM combines structural and parametric modification within a single EM process. Conceptually, Structural EM use the current network to find the expectation over data with the missing values. BIC score is employed to evaluate the candidate structure. SEM is shown to find local optima defined by the scoring function that combines the likelihood of the data with a structural penalty that discourages overly complex network. Later, SEM is extended to learn Dynamic Bayesian Net in [25]. Still there remains the open problem to identify the need for new hidden variables and how to effectively link them into the network.

A P-Net is more complex than a DBN. On one hand, the duration model makes its joint state space much bigger; on the other hand, P-Net is designed to have nodes with explicit meaning and explicit temporal/logical constraints which makes the candidate topology much less. How to automatically construct a P-Net with these features remains a

challenge. Generic structure learning algorithm will find some local maxima topology that usually only suitable to the training set. In activity domain, the training set is usually small, which cause the final result to be very poor in generalization. Another problem is all nodes in P-Net are hidden, while generic structure learning is shown to be effective with less than 30% hidden nodes. The third problem is the noisy observation. Generic structural learning algorithms all assume a noise free training set and all data points are explainable, but that is not true in our scenario, which prevents the direct application of the above algorithms.

Some preliminary exploration shows that a semi-supervised learning may be suitable for P-Net structure learning. First, we require a fully annotated positive training set T_1 with several examples. For each example in T_1 , not only the overall activity type is available, but also the individual node states at each time step are labelled. From this fully labelled training set, we hope to generate a good initial structure. As an noise free instance of the true topology, this initial structure has more constraints than the real structure. For example, if the true partial orders are A before B and A before C, the observed sequence may be A before B and B before C, which is over-constrained on C. Therefore all subsequent structure search should focus on constraint relaxation. Then, Structural EM based exploration with BIC/BD scores on a larger positive unlabelled training set may provide a way to discover the true underlying topology. Tentative incremental structure change will be kept if it increases the performance. But we have not found a reliable way to handle the noise in observation, which often leads to a wrong topology and get the EM iteration stuck. How to learn the structure in P-Net remains an open question.

7.3.2 Anomaly vs. Outlier

One of the major applications of activity recognition is surveillance, where the primary objective is to detect anomaly. As a machine learning task, it is comparably easy to distinguish between the normal sequences which are represented by the training set and the rest sequences. But a sequence not frequent in the training set does not mean it is anomaly. It

can be outliers but acceptable example. An ideal system should draw a fine line between anomaly and outlier. There are a few research works on trying to identify the anomaly [72, 30], where anomaly is identified by the less frequency and less similarity with other examples in testing set as well as the training set. But it has so far yet to answer our question.

The differences between anomaly and outlier are not the statistics in the sampling set, where they are both rare and fall outside the boundary of the normal activity. Rather it is the outcome. For example, if the delivery task is always carried out by a pickup, an ordinary activity model may select the feature about pickup and associate it with the activity. One day, if a limousine comes in for the delivery, it will be deemed as anomaly, but actually it is an outlier. On the other hand, a pickup come in to bring something out is a real anomaly. Here the fine line is in outlier the objective of the delivery is achieved by a different method while fails in the anomaly.

As a generative model, P-Net is strong to describe what is normal. Matter of fact, the learning methods of P-Net in this thesis relies on the statistics in the training set, which makes it a natural victim to the confusion of anomaly and outlier. We have some rudimental ideas on how to augment the P-Net to handle the difference.

The objectives of the activity are not easily revealed by the statistics in the training set. In the delivery example, both the arrival of new item and a pickup are universal in the training set. But the arrival of new item is the objective and pickup is just coincidence. Here the human knowledge of what matters is very important. Once we know what is critical, the implementation is simple. In the situation of P-Net, such objectives can be represented by a specialized node that checks the finishing condition. But this approach just shifts the burden of learning to human and makes the solution less interesting.

Another way is to leverage the idea of “common sense”. In the delivery example, let’s hypothesize that the system has the notion that a pickup and a limousine fall both in the scope of vehicle and they both can be used as a carrying tool. Then even though in the

training set, it picks up the feature for pickup, when it sees the example of limousine, it can leverage the equivalence of the two and correctly identifies it is an outlier. In this way, we actually increase the generalization ability by the “common sense” and encapsulate the outlier in the acceptable scope. This “common sense” is actually an expert system. One way to represent “common sense” is to prescribe the activity ontology. But, so far there is no commonly accepted ontology, let alone to integrate it with the P-Net reasoning process efficiently.

7.3.3 Interleaved Activities

In real world problem, multiple activities of multiple persons can happen together. Though they are not directly related, otherwise it will be one activity, they can be loosely associated. Specifically, on the perceptual level, the feature vector is jointly influenced by all the active nodes in multiple activities that are in process. To interpret a sequence of potential multiple activities, an naive solution is to run multiple P-Nets at the same time. Each P-Net will find its best solution independent of the others. This solution may be acceptable in most situations.

But it would be better to coordinate the interpretations between the parallel activities. The underlying perceptual evidence are shared between the activities. For example, if the current step of activity A leads to loud noise, when the evidence shows there is noise, it will be less likely that some other activity happens to cause noise. This is the common effect known as “explain-away”. This effect is much easier to handle if the observation is definite and forms a single stream. For example, if observation is a stream of symbols from a finite syllabus, two parallel SCFGs can claim each symbol. If a symbol is owned by one grammar, it is easier to let the other grammar to say it is an “insertion” error for itself and just ignore it.

But for the situation of a P-Net, the observation has multiple dimensions. Different nodes can claim different dimensions of the observation. The concept of “explain-away”

is blurred by the independence assumption between the different observation dimensions. Actually, we find this “explain-away” happen at a higher conceptual level and require a coordination layer on top of P-Net. After constructing all P-Nets, we need to further retrieve the covariance between all pairs of the nodes. But the training set only contains single activity. There is no direct way to find the precise statistics for node i in activity A and node j in activity B. So far, we can hypothesize to use the positive segment for node i and test how likely detector for node j will turn on and vice versa, then use the result to find the covariance. The coordination layer will use the covariance as the indication for potential “explain-away” effect. But the details to materialize the the “explain-away” effect from the approximate covariance and implement such an coordination is not a trivial job. All we have so far is a conceptual solution.

Furthermore, when there are multiple activities, the detectors can really use some help from the high level explanation, i.e., to provide the detector with the focus area. Pushing the information downwards within the inference rather than just passively integrating the output from detectors is another aspect that needs to be addressed in the coordination layer.

CHAPTER VIII

CONCLUSION

This thesis analyzes the human activity and considers the following points essential to the description: 1) partial temporal order 2) parallel streams 3) logical constraints 4) duration of elements 5) Non-adjacency. To model a process with these characteristics, we introduce a new graphical model: Propagation Net (P-Net).

P-Net is prescribed first as a conceptual model, then compiled into a computational model. Its conceptual model uses nodes with duration models to represent elemental intervals and uses links to represent temporal and logical constraints. The computational model augments a Dynamical Bayesian Net with duration models and defines a joint distribution to represent the status of the process. The temporal and logical constraints are wrapped together uniformly in conditional probability functions.

The joint distribution is so big that normal Bayesian inference is not trackable. Therefore, we introduce two algorithms to handle inference on P-Nets: Local Maximal Search Algorithm (LMSA) and Discrete Condensation (D-Condensation). In LMSA, an extra independence assumption segments the joint distribution into marginal distributions and simplifies the normal exponential inference to a polynomial inference; later, iteration removes the independence assumption to find a consistent MAP explanation path. In D-Condensation, the joint distribution is sampled by weighted particles, each of which keeps its full history. P-Net topology enables an efficient search on the discrete states. Of all the possible particles for the next states, the round-robin selection keeps a fixed number not only by probability but also by topological difference.

Once the topology of the P-Net is specified by the expert, the parameters within the P-Net can be trained from a few positive examples. Furthermore, the detector for each motion

can be trained together with P-Net by our semi-supervised learning algorithm. A small number of fully annotated positive examples provide the initial position and the anchor point for P-Net, then the P-Net is refined by a few non-annotated positive examples in the EM iteration. A boosted ensemble is used as the general form for the detectors. Within the P-Net framework, the detectors can fire together, which renders the normal n-class boosting inappropriate. Instead, we introduce Contrast Boosting as a solution, as it keeps a weight for each dimension of the observation vector for each detector. Weights for the same dimension of different detectors are normalized together so as to force the detectors to focus on different dimensions. All detectors are trained together in the same boosting circle. Such training leads to a set of patterns that are different but may overlap in time if necessary.

P-Nets constitute the core engine for an automatic human activity recognition system. It excels in combining lots of noisy sensor inputs into a coherent explanation for a complex temporal process. Furthermore, it enables real-time control in the temporal domain. It can tell what is happening, and more importantly, it can predict what is expected to happen. This is a very powerful tool for focusing the sensors on a highly interested area and extract the relevant information.

So far, P-Nets have proven to work well in indoor activity monitoring. We can foresee its application generalizing into the outdoor the surveillance domain. The type of activity that P-Net can describe is wide. But we can point out some characteristics of the activity in which P-Net may do better than other models:

- generally speaking P-Nets are contextual filters that leverage the high level prior knowledge to guide and filter the low level component detectors. The more regulated the activity is, the more topology constraints can be transcribed into P-Nets, the more efficient P-Net will be in processing the noisy data. This very essential point makes P-Net strong in the activity of richer structure, e.g. football play and military activity of a team.

- P-Nets are designed to represent the cooperation of multiple agents. Multiple agents usually lead to multiple entwined streams that may be a problem in other models. But in P-Nets, such a situation is welcomed. In fact, when there is only one agent, there is only single stream, P-Net will reduce to finite state machine and lose its power.
- P-Nets are based upon intervals. Duration models are critical components. If the activity is a memoryless markov process, P-Net won't work too well. But if the activity is time dependent, e.g. cooking needs to be careful about the heating time, P-Net is a good choice.
- P-Net is a graphical model that is designed to handle noise. It thrives in noisy world. It handles insert/deletion error automatically. If the observation of activity components is definite, there is no need to use a P-Net, instead an augmented Petri-Net may be a good choice.

We hope, with all these abilities, P-Nets will draw more attention from surveillance industry and that one day, P-Net will serve not only as a research tool, but also as a part of a product that facilitates our daily lives.

REFERENCES

- [1] AGGARWAL, J. K. and CAI, Q., “Human motion analysis: a review,” in *Computer Vision and Image Understanding: CVIU*, vol. 73, pp. 428–440, 1999.
- [2] ARULAMPALAM, S., MASKELL, S., GORDON, N., and CLAPP, T., “A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking,” 2002.
- [3] BAKER, J., “Trainable grammars for speech recognition,” in *Speech Communication Papers for the 97 Meeting of the Acoustical Society of America*, pp. 547–550, 1979.
- [4] BOBICK, A., “Movement, activity, and action: the role of knowledge in perception of motion,” in *Phil, Trans Royal Society London B*, vol.352, pp1257-1265, 1997.
- [5] BRAND, M., “Coupled hidden markov models for modeling interacting processes,” in *CVPR*, 1997.
- [6] BRAND, M. and HERTZMANN, A., “Style machines,” in *Siggraph*, pp. 183–192, 2000.
- [7] BREGLER, C., “Learning and recognizing human dynamics in video sequence,” in *CVPR*, 1997.
- [8] CHARNIAK, E. and GOLDMAN, R. P., “A bayesian model of plan recognition,” in *Artificial Intelligence*, vol. 64, pp. 53–79, 1993.
- [9] CHIOLA, G., MARSAN, M., BALBO, G., and CONTE, G., “Generalized stochastic petri nets: A definition at the net level and its implications,” in *IEEE Trans Software Engineering*, vol. 19, pp. 89–107, February 1993.
- [10] CHOMSKY, N., “Three models for the description of languages,” in *IRE Trans. on Information Theory*, vol. 2, pp. 113–124, 1956.
- [11] CIARDO, G., GERMAN, R., and LINDEMANN, C., “A characterization of the stochastic process underlying a stochastic petri net,” *Software Engineering*, vol. 20, no. 7, pp. 506–515, 1994.
- [12] CLARK, A., “Unsupervised induction of stochastic context-free grammars using distributional clustering,” 2001.
- [13] COLBRY, D., PEINTNER, B., and POLLACK, M. E., “Execution monitoring with quantitative temporal bayesian networks,” in *6th International Conference on AI Planning and Scheduling*, 2002.
- [14] COOPER, G. F. and HERSKOVITS, E., “Bayesian method for the induction of probabilistic networks from data,” in *Machine Learning*, vol. 9, pp. 309–347, 1992.

- [15] DAVIS, J. W. and BOBICK, A. F., “The representation and recognition of human movement using temporal templates,” in *CVIU*, pp. 928–934, 1997.
- [16] DOUCET, A., DE FREITAS, N., and GORDON, N., *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
- [17] DOUSSON, C., GABORIT, P., and GHALLAB, M., “Situation recognition: Representation and algorithms,” in *IJCAI*, pp. 166–172, 1993.
- [18] EFROS, A., BERG, A. C., MORI, G., and MALIK, J., “Recognizing action at a distance,” in *ICCV*, pp. 726–733, 2003.
- [19] E.SHAHPIRE, R. and SINGER, Y., “Improved boosting algorithms using confidence-rated predictions,” *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
- [20] F.ALLEN, J., “Towards a general theory of action and time,” in *Artificial Intelligence*, 1984.
- [21] FINE, S., SINGER, Y., and TISHBY, N., “The hierarchical hidden markov model: Analysis and applications,” in *Machine Learning*, vol. 32, pp. 41–62, 1998.
- [22] FRIEDMAN, J., HASTIE, T., and TIBSHIRANI, R., “Additive logistic regression: a statistical view of boosting,” *Annals of statistics*, vol. 38(2), pp. 337–374, 2000.
- [23] FRIEDMAN, N., “Learning belief networks in the presence of missing values and hidden variables,” in *Proc. 14th International Conference on Machine Learning*, pp. 125–133, Morgan Kaufmann, 1997.
- [24] FRIEDMAN, N., “Bayesian structural em algorithm,” in *UAI*, 1998.
- [25] FRIEDMAN, N., MURPHY, K., and RUSSELL, S., “Learning the structure of dynamic probabilistic networks,” in *UAI*, pp. 139–147, 1998.
- [26] GARG, A., PAVLOVIC, V., REHG, J., and HUANG, T., “Audio–visual speaker detection using dynamic bayesian networks,” in *Intl Conf. Automatic Face and Gesture Rec*, 2000.
- [27] GHANEM, N., DEMENTHON, D., DOERMANN, D., and DAVIS, L., “Representation and recognition of events in surveillance video using petri nets,” in *Event Detection Workshop in CVPR*, 2004.
- [28] GONG, S. and XIANG, T., “Recognition of group activities using a dynamic probabilistic network,” in *IEEE International Conference on Computer Vision*, pp. 742–749, 2003.
- [29] GORDON, N., SALMOND, D., and SMITH, A., “Novel approach to nonlinear/non-gaussian bayesian state estimation,” in *Radar and Signal Processing IEEE Proceedings F*, vol. 140(2), pp. 107–113, 1993.

- [30] HAMID, R., JOHNSON, A., BATA, S., BOBICK, A., ISBELL, C., and COLEMAN, G., “Detection and explanation of anomalous activities: Representing activities as bags of event n-grams,” in *CVPR*, pp. 139–147, 2005.
- [31] HECKERMAN, D., GEIGER, D., and CHICKERING, D. M., “Learning bayesian networks: The combination of knowledge and statistical data,” in *KDD Workshop*, pp. 85–96, 1994.
- [32] HONGENG, S., BREMOND, F., and NEVATIA, R., “Representation and optimal recognition of human activities,” in *CVPR*, pp. 818–825, 2000.
- [33] HONGENG, S. and NEVATIA, R., “Multi-agent event recognition,” in *ICCV*, pp. 84–93, 2001.
- [34] HUANG, T., KOLLER, D., J. MALIK, G. O., RAO, B., RUSSELL, S., and WEBER, J., “Automatic symbolic traffic scene analysis using belief networks,” in *AAAI*, 1994.
- [35] INTILLE, S. and BOBICK, A., “Recognizing planned, multiperson action,” in *Computer Vision and Image Understanding*, vol. 81, pp. 414–445, 2001.
- [36] ISARD, M. and BLAKE, A., “Contour tracking by stochastic propagation of conditional density,” in *ECCV (1)*, pp. 343–356, 1996.
- [37] ISARD, M. and BLAKE, A., “Icondensation: Unifying low-level and high-level tracking in a stochastic framework,” in *ECCV*, pp. 893–908, 1998.
- [38] IVANOV, Y. A. and BOBICK, A. F., “Recognition of visual activities and interactions by stochastic parsing,” in *Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 852–872, 2000.
- [39] KANAZAWA, K., “A logic and time nets for probabilistic inference,” in *AAAI*, pp. 360–365, 1991.
- [40] KOLLER, D., WEBER, J., HUANG, T., J. MALIK, G. OGASAWRA, RAO, B., and RUSSELL, S., “Towards robust automatic traffic scene analysis in real-time,” in *ICPR*, 1994.
- [41] LARI, K. and YOUNG, S. J., “The estimation of stochastic context-free grammars using the inside-outside algorithm,” in *Computer Speech and Language*, vol. 4, pp. 35–56, 1990.
- [42] MARSAN, M. A., BALBO, G., and CONTE, G., “A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems,” in *Trans on Computers System*, 1984.
- [43] MEDIONI, G. G., COHEN, I., BRMOND, F., HONGENG, S., and NEVATIA:, R., “Event detection and analysis from video streams,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 873–889, 2001.

- [44] MINNEN, D., ESSA, I., and STARNER, T., “Expectation grammars: Leveraging high-level expectations for activity recognition,” in *CVPR*, (Madison, WI), 2003.
- [45] MOENNE-LOCCOZ, N., BREMOND, F., and THONNAT, M., “Recurrent bayesian network for the recognition of human behaviors from video,” in *ICVS*, pp. 68–77, 2003.
- [46] MOESLUND, T. B. and GRANUM, E., “A survey of computer vision-based human motion capture,” *Computer Vision and Image Understanding: CVIU*, vol. 81, no. 3, pp. 231–268, 2001.
- [47] MOLLOY, M. K., “Performance analysis using stochastic petri nets,” in *IEEE Trans Computers*, vol. C-31(9), pp. 913–917, September 1982.
- [48] MOORE, D. J. and ESSA, I. A., “Recognizing multitasked activities from video using stochastic context-free grammar,” in *AAAI/IAAI*, pp. 770–776, 2002.
- [49] MOORE, D. J., ESSA, I. A., and HAYES, M. H., “Exploiting human actions and object context for recognition tasks,” in *ICCV*, pp. 80–86, 1999.
- [50] MORET, B., *The theory of computation*. Addison-Wesley, 1998.
- [51] MURPHY, K., *Dynamic Bayesian Network: Representation, Inference and Learning*. PhD thesis, UC Berkeley, 2002.
- [52] MURPHY, K. and PASKIN, M., “Linear time inference in hierarchical hmms,” in *Neural Information Processing Systems*, 2001.
- [53] PATTERSON, D. J., LIAO, L., FOX, D., and KAUTZ, H., “Inferring high-level behavior from low-level sensors,” in *Proc. of the 5th Conference on Ubiquitous Computing (UBICOMP)*, 2003.
- [54] PAVLOVIC, V., GARG, A., and REHG, J. M., “Multimodal speaker detection using input/output hidden markov models,” in *Int’l Conf. Multimodal Interfaces*, 2000.
- [55] PEREIRA, F. C. N. and SCHABES, Y., “Inside-outside reestimation from partially bracketed corpora,” in *Meeting of the Association for Computational Linguistics*, pp. 128–135, 1992.
- [56] PINHANEZ, C. and BOBICK, A. F., “Human action detection using pnf propagation of temporal constraints,” in *CVPR*, 1998.
- [57] RABINER, L. and HWANG JUANG, B., *Fundamentals Of Speech Recognition*. PTR Prentice-Hall, Inc., 1992.
- [58] RAO, C., YILMAZ, A., and SHAH, M., “View-invariant representation and recognition of actions,” in *International Journal of Computer Vision*, vol. 50, pp. 203–206, 2002.

- [59] ROSALES, R. and SCLAROFF, S., “3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions,” in *CVPR*, 1999.
- [60] SCHWARZ, G., “Estimating the dimension of a model,” in *The Annals of Statistics*, 1978.
- [61] SIDENBLADH, H., BLACK, M. J., and SIGAL, L., “Implicit probabilistic models of human motion for synthesis and tracking,” in *ECCV*, pp. 784–800, 2002.
- [62] SMITH, P., DA VITORIA LOBOA, N., and SHAH, M., “Temporal boost for event recognition,” 2005. *ICCV*.
- [63] STARNER, T., “Master’s thesis: Visual recognition of american sign language using hidden markov models,” Tech. Rep. VISMOTR 316, MIT Media Laboratory, 1995.
- [64] SYMONS, F. J. W., *Modeling and Analysis of Communication Protocols Using Numerical Petri Nets*. PhD thesis, University of Essex, Great Britain, 1978.
- [65] TRAN, Q. and MYNATT, E., “Cook’s collage: Two exploratory designs.,” in *Position paper for the Technologies for Families workshop at CHI*, 2002.
- [66] VASCONCELOS, N. and LIPPMAN, A., “A bayesian framework for content-based indexing and retrieval,” in *Data Compression Conference*, pp. 556–571, 1998.
- [67] VU, V.-T., BREMOND, F., and THONNAT, M., “Temporal constraints for video interpretation,” in *ECAI’2002, W9: Modelling and Solving Problems with Constraints*, 2002.
- [68] VU, V.-T., BREMOND, F., and THONNAT, M., “Automatic video interpretation: A novel algorithm for temporal scenario recognition,” in *IJCAI*, 2003.
- [69] WILSON, A. D. and BOBICK, A. F., “Recognition and interpretation of parametric gesture,” in *ICCV*, pp. 329–336, 1998.
- [70] YIN, P., ESSA, I., and REHG, J. M., “Asymmetrically boosted hmm for speech reading,” in *CVPR*, (Washington DC, USA), pp. pp II755–761, June 2004.
- [71] ZHAI, Y., RASHEED, Z., and SHAH, M., “A framework for semantic classification of scenes using finite state machines,” in *International Conference on Image and Video Retrieval*, 2004.
- [72] ZHONG, H., SHI, J., and VISONTAI, M., “Detecting unusual activity in video,” in *CVPR*, 2004.