This work considers the problem of recognizing activities in surveillance video. Activities are high-level non-atomic semantic concepts which may have complex temporal structure. Activities are not easily identifiable using image features, but rather by the recognition of their composing events. Unfortunately, these composing events may, generally, only be observed up to a particular certainty.

Approaches to classification/ recognition in computer vision rely on the availability of large corpora of training data. This data can be input into learning algorithms, yielding a classifier for future examples. The domain of activity recognition in surveillance video, generally does not have this luxury. Interesting activities are usually rare, and few (if any) training examples are available for training.

This limitation limits the usefulness of probabilistic state space models such as Hidden Markov Models (HMM), for activity recognition. These type of models generally require a number of examples on the order of the state space. For these reasons, The activity recognition community has thus turned to using formal specifications of domain knowledge to model activities in video.

Thus, the leading approaches to activity recognition construct a model describing the activity model using some choice of formal methods (e.g. Petri Nets, Propagation Networks, Temporal Constraint Logic). This formal specification is then translated into a Bayesian framework in which the activity progress can be estimated using (uncertain) observations of events.
One major failing of current approaches is that they fail to separate the physical constraints of the scene (which we call context) and the constraints of the activity. The context constraints must then be modeled explicitly within the specification of each activity. This leads to an increase in the size of the model, and thus state space over which we must perform probabilistic estimation.

This paper describes FPFPN (Factored Particle Filter Petri Net), an activity recognition process which addresses the issue described above. That is, the context state and activity state are separately modeled. This intuition yields simpler activity models, that are able to achieve better recognition results (as we demonstrate experimentally). Furthermore, the factored nature of our approach, results in a reduced state space. The particle filter literature has shown that less particles are required to perform estimation when the state space is smaller [5]. In our experiments we validate this result, showing that indeed a smaller number of particles can be used to achieve a the maximal recognition rate using our proposed approach. As the complexity of particle filter approaches is a (linear) function of the number of particles used, our construction yields a more efficient recognition algorithm. Another, benefit of our approach is the modularity of its design. Unlike previous approaches, which rely on an ad hoc construction of activity models and require an in-depth understanding of the activity formalism, our approach affords a structured activity model construction which only requires knowledge of the temporal structure of the activity.

1 Introduction

This work considers the problem of activity recognition in surveillance video. An activity [21, 8, 9] is defined as a set of events with partial temporal interval ordering. An event is defined as an atomic occurrence restricted to a continuous temporal interval. The problem of activity recognition is determining whether a particular activity occurs in a video sequence. The active surveillance scenarios that motivate our work imply an online approach to activity recognition. That is, video data is processed frame by frame and activities are recognized as they occur. Recently several works have framed the problem of activity recognition as a constraint satisfaction problem combined with reasoning under uncertainty. A
typical constraint approach such as Store Totally Recognized Scenarios propagation (STRS) \([35]\) accepts
the observation of a new event as input and determines whether this occurrence is consistent with existing
activity models (specified as set of constraints). If the event occurrence is consistent the activity model
state is updated. When all the constraints in the activity model have been met, the activity is recognized.
When uncertain event observations are taken into account this process is complicated. Now instead of a
certain event observation the activity recognition process is given a certainty associated with each event
observation. The activity recognition approach must then propagate this certainty from the event level
to the activity level, while still taking into account the activity constraints. Initial attempts to combine
constraint satisfaction with uncertain reasoning \([22, 33]\) cast the constraint propagation approach into
a Bayesian framework and then utilize well-studied approaches to perform inference. These approaches
translate the specification of the activity into a set of random variables whose dependence on one another
is determined by the temporal constraints of the activity. Furthermore, some dependency between the
observed events and the current state of the activity is assumed.

The critical flaw in this type of modeling lies in the distinction between modeling all possible event
occurrences, and modeling all possible event occurrences consistent with the activity. In other words, this
type of modeling assumes that all events in the video sequence occur within the context of the activity. This
assumption is not valid for general video sequences in the surveillance video domains. For instance in the
Bank scenario considered in our experiments, clearly the bank is not always being robbed. It is possible for
the customer to enter the bank, the cashier to stand behind the counter, and even the cashier to enter the
safe during the course of completely legitimate activities, even though these events are all components of a
“Bank Attack” activity. Current approaches have dealt with this discrepancy by introducing mechanisms
to reduce confused detections caused by “non-activity context” events. \([33]\) require a model of both the
ordering and duration of each event in the activity. In \([22]\), “non-activity context” events are modeled into
the activity models as additional transitions.

In this work, we approach the problem from a different direction which is based on the intuition that
uncertain event observations are dependent on the state of the scene (which we call the context). Thus, we can use our event observations to estimate the context. The progress of the activities that we are interested in is also dependant on the evolution of this context. That is, given the context, the activity state is independent of the event observations themselves. Thus an estimation of the context can be used to estimate the progress of the activity. In most cases the constraints of the scene are few, compared to the constraints of an activity, and can be modeled in a straightforward fashion. This approach also simplifies the modeling of activities since each activity is now not burdened with modeling of events outside the activity context.

This paper is organized as follows: Section 2 gives an overview of related work in activity recognition. Sections 3 and 4 review some background on Petri Nets and Particle Filter ideas. Section 5 introduces our approach to activity recognition. Section 6 illustrates the application of our approach to an example scenario. Section 7 describes our experiments and provides discussion of their results. Finally, we conclude the paper in Section 8.

2 Related Work in Activity Recognition

We distinguish activity recognition (sometimes called scenario recognition [35]) from the related field of event recognition, the labeling of event occurrences in video sequences. Event recognition approaches output a certainty measure quantifying the confidence in each event recognition. This measure must take into account the uncertainties inherent in the video data. We informally group these uncertainties into two categories: observation uncertainty and semantic uncertainty. Observation uncertainty refers to the limitations of sensors (such as cameras) to provide exact measurements, as well as to the noise element associated with many systems. Semantic uncertainty refers to the ambiguity inherent in human definitions. For example, given a location of two objects, their “closeness” is an ambiguous semantic concept.

Each event recognition approach must make a decision on how to combine the various types of uncertainty
in order to calculate the measure of certainty. This decision is often made in an ad-hoc manner. In many cases, a separate certainty evaluation is applied for each type of event. The event recognition literature contains many approaches to the calculation of measures of certainty, including Bayesian Networks [33], Scene Statistics [30, 29], and Fuzzy membership functions [13, 14]. Alternatively, events may be detected in a binary fashion [3], or made binary by thresholding [35]. In other approaches to event recognition, the semantics of an activity domain can be formalized to enable the system to better localize event occurrences [10, 19, 25, 34].

We further distinguish work in activity recognition from the popular research domain of action recognition [7, 32]. Activities, unlike actions, have a large variation in appearance and are not sufficiently characterized by image features such as color, texture, edge orientation. Additionally, approaches to action recognition often rely on the availability of a large corpus of training data, which may be used to construct classifiers for unlabeled examples. The domain of activity recognition in surveillance video generally does not have this luxury. Interesting activities are usually rare, and few, if any, training examples are available. This constraint also limits the usefulness of probabilistic state space models such as Hidden Markov Models (HMM), which generally require a number of examples on the order of the state space. For these reasons, the activity recognition community has turned to using formal specifications of domain knowledge to model activities in video.

Since there is no standard way to specify activities, each work is obliged to define a formal representation of an activity. Thus, there are two main components of existing approaches to activity recognition: (1) a modeling formalism - allowing formal specification of activity models by a human domain expert. (2) a recognition algorithm - an approach to processing video input in combination with the formal activity specification to output an activity label. Unfortunately, in most cases these two components have strong coupling and it is difficult (although theoretically possible) to evaluate the merits of a particular formalism independently of the recognition algorithm that is attached to it.

Many well-studied formalisms have been applied to activity model specification. Several approaches have
specified activities using logical formulas [2, 10, 19, 25, 34, 35]. Each of these works typically defines several predicates and relational operators to enable reasoning about temporal relationships between events.

Formal grammars have also been used to describe activity composition [17, 31]. Petri Nets [3, 20, 23, 22, 29] are a useful formalism for modeling activities which have multiple streams of events with partial ordering constraints between them (see Section 3).

Other formalisms have been designed with the problem of activity recognition in mind. The Situation Graph Tree (SGT) formalism (used in [13, 14]) is robust in representing generalization and specialization hierarchies. The Propagation Net (P-net) formalism [33] allows modeling of multiple “streams” of events within the activity by constructing a graph representing the temporal structure of the activity. A parameterized duration model is also associated with each node and edge in the graph. A related formalism, ADBN, which adds an element of hierarchy between activities, is proposed in [24].

Deterministic algorithms for activity recognition, rely on the resolution of ambiguity at lower levels of processing. An algorithm called Store Totally Recognized Scenarios (STRS) [35], the leading deterministic recognition algorithm has been applied to several real world domains. This approach is used to recognize activities specified using temporal constraint logic. Upon observation of a new event, this algorithm stores the event and determines if it completes a (sub)activity. If so, the completed activity is stored and checked against higher level activities to determine if they were completed. A common deterministic algorithm which uses a Petri Net activity specification [3, 11, 15, 23, 29] is detailed in section 3.

Stochastic algorithms for activity recognition generally maintain a probability distribution over the state space of the activity. A certain subset of the state space are those states where the activity has been recognized. the probability assigned to this subset of the state space is the certainty of the activity recognition. In most cases the state space is described in terms of a set of random variables. The joint distribution over these variables is often simplified with assumed independence relationships (e.g. the Markov assumption) which break this distribution into a product of factors. These assumptions reduces the complexity of storage and enable efficient computation of the joint and appropriate marginal probabilities.
D-Condensation is a stochastic algorithm, based on sampling, applied to activities specified as Propagation Nets [33]. This algorithm has been shown to be effective in such applications as quality control of glucose monitoring. To allow simplification of the joint distribution it is assumed that nodes (events) in the net become active (when the event occurs) in a restricted order. Furthermore, this approach assumes a separate distribution over the duration of each event and the time between events is known in advance. The algorithm for recognizing activities formalized as ADBNs [24], uses similar assumptions, but assumes only a duration model for the time between events. [2] proposes a generic framework for probabilistic logical entailment.

In [22] a Petri Net activity model is constructed for each activity of interest in the scene. These models aspire to model all activity and non-activity context. However, this approach results in complex activity models which require time-consuming manual construction by an expert who is familiar with both the activity domain and the Petri Net formalism. These complex models also require more resources to perform efficient inference (see experiments section).

3 Petri Nets

Petri Nets (PN) are specified as a directed bipartite graph (see Figure 1). Graphically, place nodes are represented as circles and transition nodes are represented as rectangles. Each place node holds zero or more tokens and transition nodes specify the movement of tokens between place nodes when a state change occurs. Those place nodes connected by directed arcs into the transition node are the input place nodes of the transition. Similarly, those place nodes connected by directed arcs out of the transition are called the output place nodes of the transition. A transition node is enabled if all of its input place nodes hold tokens. A special type of arc called an inhibitor arc requires that an input place node must not hold a token in order for the transition to become enabled. Enabled transition nodes may fire, altering the distribution of tokens in the Petri Net. When an enabled transition node fires, the tokens held in the input place
Conditional transition nodes can have an enabling rule applied to them which imposes additional conditions on the enabling of the transition. A PN model marking is defined as the instantaneous configuration of the tokens held in the various place nodes in the PN graph. For further details on the PN formalism interested readers are referred to [18, 26].

The prevalent approach to modeling video activities with Petri Nets is as follows: The Petri Net fragment is laid out as a “plan” left to right. The initial marking of the Petri net consists of a single token in the left most ‘source’ Place node. A single token in the right-most ‘sink’ place node represents the recognized state of the activity. All remaining place node represent “waypoints”, intermediary states in the progression of the activity. Each transition in the activity Petri Net is conditioned on the observation of an external event. This transition is only allowed to fire once this external event has been observed. The arcs in the activity net connect the transitions in such a way as to enforce the temporal constraints of the activity. Using the Petri Net formalism, we are able to model all temporal interval relations as defined in [4]. For further details on the construction of Petri Net activity models, readers are referred to the literature [29, 3, 18, 23].
4 Particle Filter

In a Bayesian approach to analyzing dynamic systems, the goal is to estimate the posterior distribution $P(x_t|y_{1:t})$ over the system state at time $t$, denoted $x_t$, taking into account all observations up to the current time, denoted $y_{1:t}$. The Bayesian Recursive Filter (BRF) is an approach appropriate for online problems in which $P(x_{t-1}|y_{1:t-1})$, the previous estimation of the posterior, is used in a recursive fashion to derive an updated estimation with each new observation. Particle filters (PF), also known as sequential Monte Carlo methods, are techniques for probability density estimation within the BRF framework based on sampling. The PF approach maintains a set of $N$ hypotheses of the current state called particles, denoted $X_t = \{x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(N)}\}$. Each particle $x_t^{(i)}$ is associated with a weight, denoted $w_t^{(i)}$. The weights are used to approximate the posterior distribution as follows:

$$P(x_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta(x_t, x_t^{(i)}) \tag{1}$$

where $\delta$ denotes the Dirac delta function.

The BRF framework consists of two major components. The dynamic model, denoted $P(x_t|x_{t-1})$, describes the evolution of states over time. The measurement model, denoted $P(y_t|x_t)$, specifies the relationship between state and observation variables. After initialization a BRF approach proceeds in two stages, prediction and correction. In the prediction stage, the previous estimation of the posterior is used to arrive at an estimation of possible future states. The PF algorithm samples each new particle, $x_t^{(i)}$, from $P(x_t|x_{t-1}^{(i)})$. This distribution, called the proposal (or importance) distribution, is often used as an approximation to the posterior. In the correction stage, the most recent observation, $y_t$, is used to adjust the prediction, increasing (decreasing) the probability of states consistent (inconsistent) with the observation. In the PF algorithm each particle weight, $w_t^{(i)}$, is multiplied by $P(y_t|x_t^{(i)})$. The weights are then renormalized to sum to one. Particle filter algorithms are used in many application fields including visual tracking of objects [16]. For more on particle filter approaches the reader is referred to [12, 5].
5 Our Approach

In this section we will provide the details of our model construction and inference procedures. Our model construction creates a Petri Net which is segmented into several fragments. These fragments are of one of two types. Context fragments model all relevant events that can occur in a particular surveillance domain. Activity fragments model temporal constraints on event ordering within activities. Once these fragments are defined, a Bayes recursive framework (BRF) can be inferred. Within this framework we apply a particle filter based estimation of the state. This estimation is divided into two parts estimation of the context, and estimation of the activity state.

5.1 Constructing the Petri Net Activity Model

5.1.1 Constructing Context Fragments

Context fragments model all relevant states an object can take on as well as the relevant events in the event domain. These fragments capture the physical constraints of the domain, independent of any activities than can occur. Each transition node, in a context fragment is labeled with an associated event that may be observed during the processing of a video sequence. Each place in the event context fragment can also be associated with a semantic meaning. Often there will be several independent context fragments representing different facets of the scene object state (e.g. which area of the scene the object is in, how fast the object is moving, how close an object is to other objects, etc.). These fragments will be unconnected.

5.1.2 Constructing Activity Fragments

Each relevant activity is represented by several Petri Net fragments. These fragments represent the temporal constraints on event ordering that the activity contains. Hence we refer to these as constraint fragments. Each such constraint fragment will make use of a template fragment which represents the particular type of temporal constraint. The template fragment is then specialized according to the parameters of the constraint (i.e the specific events participating in the constraint). This specialization is achieved by labeling
the appropriate transition nodes in the constraint fragment according to the constraint’s dependence on
the context fragment marking.

5.1.3 The Template Fragments

Figure 2 shows the most common template fragments used in constructing our activity models. These
relations correspond to commonly used interval relations: Before, Overlap and During. The General
Overlap relation is meant for the case in which two events are constrained to occur at the same time but
their starting and ending points are not relevant to the activity definition. Although this relation could
be constructed by combining a number of other fragments, having a separate simpler relation fragment
makes modeling this type of constraint less demanding on the recognition process.

In the paradigm of activity recognition we utilize an online approach to relation detection. That is,
we would like to recognize a particular activity and all its composing constraints as soon as they occur.
Consider the interval relation such as A During B, where B is significantly longer. In an online recognition,
evaluated at every frame, waiting until interval B is concluded would result in a delayed recognition of the
interval relation. A relaxed approach to the during relation, which permits recognition of the relation after
interval A ends (under the assumption that interval B ends eventually), results in a much earlier detection
of the relation. Using this relaxed approach temporal constraints can be validated earlier than they would
be under the strict approach, and thus earlier recognition of activities is possible.

5.1.4 Chaining Relations

When modeling activities in the surveillance domain it is often necessary to chain relations. That is, to
define a relation between the occurrence of a relation and the occurrence of an event interval (or another
relation). For example, we may require event A to occur before event B, and this before relation to occur
during event C. We facilitate this modeling functionality by connecting the fragment of the inner relation
to the fragment of the outer relation in the appropriate way. More specifically, for each
Figure 2: The template fragments used for the various temporal relations.
relation template we define the markings for which the relation is "on" (in progress) and the markings for which the relation is "off". We can now substitute these markings, for the markings in the context fragment, which primitive relations rely on. Figure 3 illustrates this construction.

5.2 Constructing the Probabilistic Model

In order to apply particle filter techniques to our approach we have to translate our formulation of an activity, specified as a Petri Net, into the language of the Bayesian Recursive Filter (BRF). More specifically, we have to define the space of possible states our system can take on at each time step, and the space of possible observations at each time step. Furthermore, we must define the dynamic model, $P(x_t|x_{t-1})$, which describes the evolution of states in time. As well as the measurement model, $P(y_t|x_t)$, which defines the likelihood of a particular observation given the system is in a particular state. We will take advantage of the disjoint fragments of our Petri Net model construction in order to construct a BRF framework.
where efficient state estimation is possible.

5.2.1 Notation

To keep our formulations concise we introduce the following notation to denote vector valued variables:

\[ i^j x^t_i \]

Here \( i \) denotes the factor, \( t \) denotes the time, and \( j \) denotes the particle id. To indicate a particular component of a vector we add the additional index \( c \):

\[ i^j x^t_{i(c)} \]

some of these indices may be omitted as appropriate, but their configuration is kept consistent throughout the paper.

5.2.2 Preliminaries

As we have detailed in previous section, the activity models in our approach are composed of multiple Petri Net fragments. In this section we will refer to these fragments as factors, each with its own state, which in combination form the system state. Like their corresponding fragments, these factors are divided into two groups, context factors and activity factors. We will denote the set of all factors as \( F \), the set of all context factors as \( C \), and the set of all activity relation factors as \( A \). Clearly \( C \cup A = F \) and \( C \cap A = \emptyset \). The set of activity labels in the model is denoted \( \text{activities} \). We also define the function \( \delta : A \rightarrow \text{activities} \), as a map from each activity relation fragment to the activity it is involved in. Given a particular Petri Net fragment \( i \), coupled with an initial marking, we can derive the set of all reachable markings. We denote this set using the letter \( S_i \). The fragment \( i \) also defines the set of all relevant events that may be observed. These events are those used to label the transition nodes in the Petri Net fragment. Note that
since multiple transitions may have the same event label, the number of events is not necessarily equal to
the number of transitions. We denote the set of all events in fragment \( i \) by \( O_i \). Since multiple events may
be observed at the same time we define another set, \( M_i \), as a subset of the powerset of \( O_i \). This set serves
as the space over all relevant event sets that may occur during the video analysis. Note that \( M_i \), trivially
contains \( \emptyset \), the empty set.

5.2.3 The State Space

In order to model Petri Net dynamics in this paradigm we must consider the (possibly empty) set of
events that occur at each frame as part of the state. The state of our model is conveniently divided into
independent factors which allows us to estimate the joint state over all factors by estimating the state of
each factor separately.

Thus, \( x_t \), the state variable at frame \( t \) of the video sequence, will be factored into several variables
representing the various factors of the activity model. That is \( x_t = < i^1 x_t, i^2 x_t, \ldots, |F| x_t > \). Each of these
components \( i x_t = < i^x t(m), i^x t(e) > \), will take on a value with two components. The first component,
\( i^x t(m) \), will denote the marking of PN factor \( i \) at time \( t \). The second component, \( i^x t(e) \), will denote the
event set relevant to factor \( i \), that occurs at time \( t \).

More formally, \( i^x t(m) \in S_i \) and \( i^x t(e) \in M_i \). Thus, the state space of \( i^x t \) is \( S_i \times M_i \).

5.2.4 The Observation Space

At each frame we will observe a certainty value for each relevant event in set \( O \). Thus the observation
is a vector, \( y_t \in [0,1]^{[O]} \). Since the observation is independent of the activity state given the context
state (see next section), \( y_t \) will be factored according to the various context factors. That is, \( y_t = <
^y t, ^y 2, \ldots, ^y C > \). Each \( ^y t \in [0,1]^{[O]} \) is a vector. The \( j \)-th entry in vector \( ^y t \), which we will denote
\( \text{\( ^y t(j) \)}, \) denotes the certainty in the observation of \( O_t(j) \) the \( j \)-th event in set \( O_t \), according to an arbitrary
pre-defined order.
Figure 4: The conditional independence relations between the context, activity and observation variables. The activity state is independent of the observation given the context.

5.2.5 Conditional Independence

In addition to defining the space of our state and observation variables, we also assume some conditional independence relationships among the variables. First, we make use of the (first-order) Markov Assumption, which asserts that the state at any frame \( t \) is independent of the state at all previous frames, given the state at frame \( t - 1 \). Hence, it is only necessary to consider the dependence between the states at frames \( t \) and \( t - 1 \).

Recall that our state space for each factor is decomposed into two components: marking and event set. Each marking at time \( t \) is dependent on the marking at time \( t - 1 \) as well as those event set that occurred at time \( t - 1 \). Furthermore, the event set that occurs at time \( t \) is dependant only on the marking at time \( t \). This assumptions are a natural modeling of the Petri Net dynamics and are illustrated by a graphical model in Figure 5.
Another assumption made in our model is that the context state fully determines the observation. In other words, the observation is independent of the activity state given the context state. These assumption is illustrated in the simple graphical model shown in Figure 4.

Now let us take a closer look at how the activity state is determined. Like the context state, the activity state is composed of two components: the marking and event set. Similar to the context state, each activity state marking at time $t$ is determined by the activity state marking at time $t-1$ as well as the events that occurred at time $t-1$. However, unlike the context state, the activity state events that occur at time $t$ are not determined only by the activity state marking at time $t$, but rather by a combination of the activity state marking and the context state marking at time $t$. These dependencies are shown in the graphical model in Figure 6.

### 5.2.6 The Dynamic model

In this section we will derive a Dynamic model from the previously defined Petri Net Model Structure. In doing so we will translate transitions labeled with events into a probability of whether these transitions have fired (upon observation of the corresponding event). As such we will mildly abuse terminology by
Figure 6: The conditional independence relations between the context and activity variables. The context at time $t$ (denoted $C_t$) is a factor in determining which events occur at time $t$ in each activity factor.

referring to enabled transitions as enabled events. Similarly we may say “all transitions in an event set are enabled”, instead of the more precise “all events in the event set have a corresponding transition node which is enabled”

In constructing the dynamic model, $P(x_t|x_{t-1})$, let us consider it in two decomposable pieces, the context dynamic model, denoted $P_c(x_t|x_{t-1})$, and the activity dynamic model, denoted $P_a(x_t|x_{t-1})$. We will make use of a simplifying assumption that these pieces are independent of one another such that:

$$P(x_t|x_{t-1}) = P_c(x_t|x_{t-1}) \cdot P_a(x_t|x_{t-1})$$

Let us first consider the decomposition of the context dynamic model $P_c(x_t|x_{t-1})$:

$$P_c(x_t|x_{t-1}) = \prod_{i \in C} P(i|x_t|m, i|x_{t-1}(m), i|x_{t-1}(e)) \cdot P(i|x_t|e)$$

$$= \prod_{i \in C} P(i|x_t|m, i|x_{t-1}(m), i|x_{t-1}(e)) P(i|x_t|e|x_{t-1}(m))$$

(4)
where the second equality is due to the independence between context factors. The third equality is due to the conditional independence relationships discussed in the previous section (see Figure 5).

Similarly, the activity dynamic model, $P_a(x_t|x_{t-1})$, is decomposed as follows:

$$P_a(x_t|x_{t-1}) = \prod_{i \in A} P(i \cdot x_t|x_{t-1}) = \prod_{i \in A} P(i \cdot x_{t(m)}|i \cdot x_{t-1(m)}, i \cdot x_{t-1(e)}) = \prod_{i \in A} P(i \cdot x_{t(m)}|i \cdot x_{t-1(m)}, i \cdot x_{t-1(e)}) P(i \cdot x_{t(e)}|i \cdot x_{t(m)})$$

(5)

Note that in the latter decomposition we used the term $P(i \cdot x_{t(e)}|i \cdot x_{t(m)})$ instead of $P(i \cdot x_{t(e)}|i \cdot x_{t(m)}, C_i)$. This is a simplifying assumption, which indicates that although the events in the activity factor $i$ at time $t$, depend on the context, our simplified model will not consider this dependence for the time being (we will correct for this during inference). Using the above models it suffices to derive $P(i \cdot x_{t(e)}|i \cdot x_{t(m)})$ and $P(i \cdot x_{t(m)}|i \cdot x_{t-1(m)}, i \cdot x_{t-1(e)})$ for each factor $i$ from the Petri Net structure. As no information is available on the first component of the dynamic model, we define an uninformative $P(i \cdot x_{t(e)}|i \cdot x_{t(m)})$, for some factor $i$ giving equal probability to all enabled transitions from the structure of the Petri Net factor as follows:

$$\hat{P}(i \cdot x_{t(e)}|i \cdot x_{t(m)}) = \begin{cases} 1 & \text{if all transitions in } i \cdot x_{t(e)} \text{ are enabled in marking } i \cdot x_{t(m)} \\ 0 & \text{otherwise} \end{cases}$$

(6)

where $i \cdot x_{t(e)} = \emptyset$ (the empty set) is considered to be enabled in all markings.

We then normalize

$$P(i \cdot x_{t(e)}|i \cdot x_{t(m)}) = \frac{\hat{P}(i \cdot x_{t(e)}|i \cdot x_{t(m)})}{\sum_{x' \in M_i} \hat{P}(i \cdot x'|i \cdot x_{t(m)})}$$

(7)

Similarly, $P(i \cdot x_{t(m)}|i \cdot x_{t-1(m)}, i \cdot x_{t-1(e)})$ is derived from the Petri Net fragment $i$ as follows:
\[
P^\prime(i_{x(t(m)}|i_{x(t-1(m))}, i_{x(t-1(c))}) = \begin{cases} 1 & \text{if } i_{x(t(m))} \leftarrow i_{x(t-1(m))}|i_{x(t-1(c))} \\ 0 & \text{otherwise} \end{cases}
\]  

where \( i_{x(t(m))} \leftarrow i_{x(t-1(m))}|i_{x(t-1(c))} \) indicates there is a path from state \( i_{x(t-1(m))} \) to state \( i_{x(t(m))} \) via transitions labeled with the event(s) \( i_{x(t-1(c))} \)

We then normalize:

\[
P(i_{x(t(m)}|i_{x(t-1(m))}, i_{x(t-1(c))}) = \frac{\hat{P}(i_{x(t(m)}|i_{x(t-1(m))}, i_{x(t-1(c))})}{\sum_{x' \in S} \hat{P}(x'|i_{x(t-1(m))}, i_{x(t-1(c))})}
\]

5.2.7 The Measurement Model

Like the dynamic model the measurement model, \( P(y_t|x_t) \), can also be decomposed. First we use our assumption that only the context state factors determine the observation, and each context state factor has its own disjoint set of observations.

\[
P(y_t|x_t) = \prod_{i \in C} P(i_{y(t)|i_{x_t}})
\]

We also make use of the assumption that each observation is independent of all other observations. Thus, for each factor \( i \):

\[
P(i_{y(t)}|i_{x_t}) = \prod_{j=1}^{\mid O_i \mid} P(i_{y(t(j))}|i_{x_t})
\]

where \( i_{y(t(j))} \) is the \( j \)-th component of the vector \( i_{y_t} \). For some factor \( i \) and vector component \( j \) we can further simplify our formula:

\[
P(i_{y(t(j))}|i_{x_t}) = P(i_{y(t(j)}|i_{x(t(m))}, i_{x(t(e))}) = P(i_{y(t(j)}|i_{x(t(c))})
\]

where the second equality is due to the conditional independence relationships described in Figure 5.

Thus, for each (context) Petri Net factor \( i \) it suffices to derive \( P(i_{y(t(j)}|i_{x(t(c))}) \) from the Petri Net structure,
to obtain the measurement model. The construction of the observation likelihood, $P(iy_{t(j)} | ix_{t(e)})$, models the notion that if a particular event occurs, the observed certainty in this event should be high. That is, if the event set occurring at time $t$, denoted $ix_{t(e)}$, contains the $j$-th event $Oi(j)$, the certainty in the observation $iy_{t(j)}$ should be high. The sigmoid function (centered on 0.5) is a natural fit for modeling this intuition. Conversely if an event does not occur, the observed certainty in this event should be low. The complement to the sigmoid function models this situation. Each vector component in $iy_t$ represents certainty in one of the events in set $Oi$. Let us use $Oi(j)$ to denote the event whose certainty is represented by the $j$-th component of $iy_t$. Since, $ix_{t(e)} \in M_i \subseteq 2^{Oi}$ (the powerset of $Oi$) we can check the membership of $Oi(j)$ in the set $ix_{t(e)}$ thus:

$$P(iy_{t(j)} | ix_{t(e)}) = \begin{cases} 
\varphi(iy_{t(j)}) & \text{if } O_i(j) \in ix_{t(e)} \\
1 - \varphi(iy_{t(j)}) & \text{otherwise}
\end{cases} \quad (13)$$

where $\varphi(z)$ is the sigmoid function centered on 0.5 and $k$ is a parameter:

$$\varphi(z) = \frac{1}{1 + e^{-k(z-0.5)}} \quad (14)$$

5.2.8 The Prior Probability

The prior probability over the marking variable of the state, $P(ix_{0(m)})$, is dictated by the initial marking of the Petri Net Fragment $i$

$$P(ix_{0(m)}) = \begin{cases} 
1 & \text{if } ix_{0(m)} = \text{initial marking} \\
0 & \text{otherwise}
\end{cases} \quad (15)$$

5.3 Performing Inference

Recall that our objective in this work is to translate a list of events with associated certainty values into an activity label (with its own associated certainty value). In this section we describe the mechanics of
this “propagation” of certainty from the event level to the activity level. Our approach to the propagation of certainty is based on the particle filter. This approach attempts to estimate the system state based on all available information. Recall that this state is decomposed into the context state, which estimates the properties of all objects in the scene, and the activity state which estimates how many activities (and their components) have been recognized. Throughout our analysis of the video we will maintain a set of particles for each factor of the activity model. Each particle contains a hypothesis of the current state of the factor (see section 5.2.3) and corresponding weight. Together these particles will approximate the posterior distribution over the factor, \( P(i \mathbf{x}_t | \mathbf{y}_{1:t}) \). This distribution is the estimation of the factor state given all information we have seen so far.

Since all factors are independent we can compute the total posterior distribution using the formula:

\[
P(\mathbf{x}_t | \mathbf{y}_{1:t}) = \prod_{i \in F} P(i \mathbf{x}_t | \mathbf{y}_{1:t})
\]

Formally, we denote by the set of particles for factor \( i \) at time \( t \) as, \( i \mathbf{X}_t = \{i \mathbf{x}_t^{(1)}, i \mathbf{x}_t^{(2)}, i \mathbf{x}_t^{(3)}, \ldots, i \mathbf{x}_t^{(N)}\} \), where \( N \) is the number of particles, and each \( i \mathbf{x}_t^{(j)} \in S_i \times M_i \), for \( j = 1..N \). The set of weights corresponding to the particles for factor \( i \) at time \( t \) is denoted as \( i \mathbf{w}_t = \{i w_t^{(1)}, i w_t^{(2)}, i w_t^{(3)}, \ldots, i w_t^{(N)}\} \), such that, \( \sum_{j=1}^{N} i w_t^{(j)} = 1 \).

The posterior distribution is then approximated by the particles as follows:

\[
P(i \mathbf{x}_t | \mathbf{y}_{1:t}) = \sum_{j=1}^{N} i w_t^{(j)} \delta(i \mathbf{x}_t, i \mathbf{x}_t^{(j)})
\]

where \( \delta \) indicates the Dirac delta function.

The dynamic model is denoted \( P(\mathbf{x}_t | \mathbf{x}_{t-1}) \), and is derived from the various Petri Net fragments as described in section 5.2.6. The proposal distribution, from which samples are drawn, is set to be equal to this dynamic model. This is a simplifying assumption that is adopted in many works utilizing the particle filter framework.
5.3.1 Initialization

The particle set $iX_0$, is initialized in two stages. First we sample the marking component of the state variable from the prior probability distribution, $P(i\cdot x_0(m))$ (see section 5.2.8) defined above. Second, we sample the event combination component of the state variable from $P(i\cdot x_0(e)|i\cdot x_0(m))$, using the dynamic model (this is a special case of the prediction step). The corresponding weights are initialized to $1/N$, where $N$ is a parameter indicating the number of particles.

For $j = 1 \ldots N$

$$i\cdot x_0^{(j)}(m) \sim P(i\cdot x_0(m))$$

$$i\cdot x_0^{(j)}(e) \sim P(i\cdot x_0(e)|i\cdot x_0^{(j)}(m))$$

$$i\cdot w_0^{(j)} = \frac{1}{N}$$

5.3.2 Update

At each frame $t$ from 1 to $T$, we denote the observation vector as $y_t$. Recall that this vector resides in $[0, 1]^{\mid O\mid}$, where each entry represents the observation certainty in one of the events in set $O$. This vector is given as input from the event recognition layer. Each factor has its own subset of relevant observation components, denoted $y_t$.

For each frame $t$ we do the following:

1. for each context factor $i \in C$

   1.1. Update the weights of each particle using the observation as follows (the correction step of the Bayes Recursive Filter):
For $j = 1 \ldots N$

$$i \hat{w}_t^{(j)} = i w_{t-1}^{(j)} \cdot P(y_t^{i} | x_t^{(j)})$$ (18)

1.2. Normalize the weights:

For $j = 1 \ldots N$

$$i w_t^{(j)} = \frac{i \hat{w}_t^{(j)}}{\sum_t i \hat{w}_t^{(j)}}$$ (19)

1.3. Sample the next set of particles from the proposal distribution (the prediction step of the Bayes Recursive Filter):

For $j = 1 \ldots N$

$$i x_t^{(j)} \sim P_t^{i}(x_t | x_{t-1}^{(j)})$$ (20)

2. for each activity relation factor $k \in A$

2.1. Update the weights of each particle using the observation as follows (the correction step of the Bayes Recursive Filter):

For $j = 1 \ldots N$

$$k \hat{w}_t^{(j)} = k w_{t-1}^{(j)} \cdot \frac{P(C^{(k x_{t(e)}^{(j)}, k x_{t(m)}^{(j)})})}{P(k x_{t(e)}^{(j)}, k x_{t(m)}^{(j)})}$$ (21)

Here $P(C^{(k x_{t(e)}^{(j)}, k x_{t(m)}^{(j)})})$ denotes the marginal probability over the context fragment for the condition required to enable event(s) $k x_{t(e)}^{(j)}$ while in marking $k x_{t(m)}^{(j)}$.

2.2. Normalize the weights:

For $j = 1 \ldots N$

$$k w_t^{(j)} = \frac{k \hat{w}_t^{(j)}}{\sum_t k \hat{w}_t^{(j)}}$$ (22)

2.3. Sample the next set of particles from the proposal distribution (the prediction step of the Bayes Recursive Filter):
For $j = 1 \ldots N$

$$k^{(j)}_x \sim P_a(t_x | k^{(j)}_x)$$  \hspace{1cm} (23)$$

As previously mentioned, the particle weights provide an approximation of the posterior probability distribution, $P(x_t | y_{1:t})$. Calculating the marginal over all recognized places in each of the activity fragments and computing the product of these marginals, gives us the certainty in the occurrence of each of the activities.

5.3.3 Calculating the Marginals

During the inference process we are often interested in calculating the probability of the context state configuration being consistent with some logical clause of literals (e.g. $E_1 \land \neg D_1$), representing the conditions under which a transition in one of the activity fragments should be allowed to fire. Each literal in this clause represents a place node in one of the context fragments. For example, in the clause above, literal $E_1$ represents place node $E$ in context fragment $c_1$ (depicted in Figure 7).

Let us define a positive literal to be consistent with some marking $m$ of the corresponding context fragment, if the place node referred to by the literal contains one or more tokens in marking $m$. For example, the positive literal $E_1$ is consistent with any marking of context fragment $c_1$ in which place node $E$ contains a token.

Similarly, a negative literal is defined to be consistent with some marking $m$ of the corresponding context fragment, if the place node referred to by the literal contains no tokens in marking $m$. For example, the negative literal $\neg D_1$ is consistent with any marking of context fragment $c_1$ in which place node $D$ contains no token.

Using our approximation of the posterior it is possible to compute the marginal probability of some positive literal $A$ with corresponding context fragment/factor $c$ as follows:

25
\[ P(A) = \sum_{j=1}^{N} I_{A}(c_{x^{(j)}_{t(m)}}) \cdot c_{w^{(j)}_{t}} \]  

(24)

where

\[ I_{A}(x') = \begin{cases} 
1 & \text{if marking } x' \text{ is consistent with literal } A \\
0 & \text{otherwise} 
\end{cases} \]  

(25)

Similarly the marginal probability of some negative literal \( \neg B \) with corresponding context fragment \( c \) is given by:

\[ P(\neg B) = \sum_{j=1}^{N} (1 - I_{B}(c_{x^{(j)}_{t(m)}})) \cdot c_{w^{(j)}_{t}} \]  

(26)

using this definition for the marginal probability of literals, we can now compute the marginal probability of a conjunction of literals (e.g. \( C = A \land B \)):

\[ P(C) = \prod_{c \in C} P(c) \]  

(27)

where \( c \) is a literal in conjunction \( C \).

Similarly we compute the marginal probability of a disjunction (e.g. \( D = A \lor B \)):

\[ P(D) = 1 - \prod_{d \in D} (1 - P(d)) \]  

(28)

where \( d \) is a literal in disjunction \( D \).

Now if we are given a formula in conjunctive normal form (e.g. \( C = D_1 \land D_2 \land D_3 \), where each \( D_i \) is a disjunction) we can compute its marginal probability as follows:

\[ P(C) = \prod_{D_i \in C} \left( 1 - \prod_{d \in D_i} (1 - P(d)) \right) \]  

(29)

where \( d \) is a literal in disjunction \( D_i \).
5.3.4 The Context Update step

In this section we derive the update equations used for inference in section 5.3.2. The update step of the particle filter is analogous to the correction step of the Bayesian recursive filter. That is, we aim to correct our prediction in light of new information. This correction is achieved by dividing the estimation of the posterior, evaluated at the sampled particle values, by the proposal distribution used to approximate the posterior during sampling.

In the context portion of the inference procedure (Equation 21), we use the context dynamic model, \( P_c(i_x_t|i_x_{t-1}) \), to perform the prediction (i.e. sample particles). We should correct using the posterior evaluated at \( i_y_t \), given by \( P(i_y_t|i_x_t)P_c(i_x_t|i_x_{t-1}) \).

Hence for each particle \( j \) the correction step is as follows:

\[
i_w^{(j)}_t = i_w^{(j)}_{t-1} \cdot \frac{P(i_y_t|x^{(j)}_t)P_c(x^{(j)}_t|x^{(j)}_{t-1})}{P_c(x^{(j)}_t|x^{(j)}_{t-1})} = i_w^{(j)}_{t-1} \cdot P(i_y_t|x^{(j)}_t)
\]

5.3.5 Activity Update Step

Let us consider the “true” posterior over the activity state implied by the conditional independence relationships shown in Figure 6.

\[
P_a(i_x_t) = P(i_{x_t(m)}, i_{x_t(c)}) = \sum_{C_t} P(i_{x_t(m)}|i_{x_{t-1}(m)}), P(i_{x_t(c)}|i_{x_{t-1}(c)}, C_t) \cdot P(C_t)
\]

\[
= P(i_{x_t(m)}|i_{x_{t-1}(m)}, i_{x_{t-1}(c)}) \cdot \sum_{C_t} P(i_{x_t(c)}|i_{x_{t(m)}, C_t}) \cdot P(C_t)
\]

where the sum over \( C_t \) denotes summing over all possible conditions that can exist in the context fragment.

Although there are many possible such conditions we observe that, keeping \( i_x_t \) fixed:
\[ P(i_{x(t(e)}|i_{x(t(m)}), C_t) = \begin{cases} 1 & C_t = C(i_{x(t(m)}, i_{x(t(e)}) \\ 0 & \text{otherwise} \end{cases} \tag{32} \]

where \(C(i_{x(t(m)}, i_{x(t(e)})\) denotes the condition that enables event set \(i_{x(t(e)}\) in marking \(i_{x(t(m)}\).

Using the above we can simplify:

\[ \sum_{C_t} P(i_{x(t(e)}|x_{t1}, C_t) \cdot P(C_t) = P(C_t = C(i_{x(t(m)}, i_{x(t(e)})) \tag{33} \]

And thus the posterior (Equation 31) is reduced to:

\[ P_a(x_t) = P(i_{x(t(m)}|i_{x(t-1(m)}, i_{x(t-1(e)}), P(C_t = C(i_{x(t(m)}, i_{x(t(e)})) \tag{34} \]

In the activity fragment update (Equation 23) \(i_{x(t(m)}\) and \(i_{x(t(e)}\) are known from the prediction(sampling) step, and \(P(C_t = C(i_{x(t(m)}, i_{x(t(e)}))\) is known by calculating the marginal over the context fragment using the particle approximation of the context posterior.

Thus in the correction phase we can use the following formula:

\[ i_{w(j)}^{(t)} = i_{w(j)}^{(t-1)} \cdot \frac{P(i_{x(t(m)}|i_{x(t-1(m)}, i_{x(t-1(e)}), P(C_t = C(i_{x(t(m)}, i_{x(t(e)}))}{P(i_{x(t(e)}|i_{x(t(m)}))} \tag{35} \]

6 Example

In this section we will run through constructing an example activity model in the Bank surveillance domain.

We shall start by defining the roles that participate in our relevant activities. In our experiments we are interested in two roles: Visitor and Cashier. Now we can construct the context fragments. In the bank
setting, each object can occupy one of a number of disjoint zones. We will represent this knowledge of
the context by creating the context fragment shown in Figure 7. All transition nodes within the context
fragment are labeled with events that we may observe in our video sequence. Referring to the figure, *In
Entrance Zone, In FrontCounter Zone*, etc. are all events which can be recognized up to some certainty
by our event recognition module. One such fragment will be created for each of the context roles.

Now let us consider one of the activities we are interested in recognizing in the bank surveillance domain,
namely the *Bank Attack* activity. In natural language we can describe this activity as follows: A visitor
enters the bank as the cashier is behind the counter, the visitor goes behind the counter, the cashier and the
visitor walk into the safe area at the same time. Translating this description in a set of temporal constraints,
we have *During(visitor in entrance, cashier in backcounter), Before(cashier in backcounter,cashier in safe),
Before(visitor in backcounter,visitor in safe), GeneralOverlap(visitor in safe,cashier in safe)*. Clearly,
these relations correspond to the relations we have previously discussed, so it is straightforward to construct
the appropriate PN fragments for them. These fragments are shown in Figure 8.

Recall that we consider the right-most place node of each fragment in the figure to be the “source” place
node and the left-most place node to be the “sink” place node. The source place node will contain
a token when the system is initialized. The sink place will contain a token when the relation has been
observed. When all relation fragments participating in the activity contain tokens in their recognized place
node, the activity is recognized. The dynamics of the activity Petri Net fragments are straightforward, each transition node is labeled with a “context condition”. These conditions are conjunctions of literals. Each literal corresponds to a place node within the context fragment(s). A literal becomes true when the corresponding place node contains a token (or does not if it is a negative literal). If a transition is enabled while the appropriate context condition becomes true, that transition is fired, modifying the token distribution throughout the Petri Net fragment.

6.1 Constructing the Bayes Recursive Filter Components

Let us consider the marking space of each fragment presented in the above example. The context fragment (Figure 7) will allow a token to be contained in only one of five places at any given time. Thus, using an initial marking with a single token, there are five reachable markings for this fragment. The activity fragments similarly have a small number of reachable markings, summarized in Table 1. Recall that $S_i$ denotes the set of all possible markings in Petri Net fragment $i$. For example, the state space of context
Table 1: The reachable states of the PN fragments depicted in Figures 7, 8.

<table>
<thead>
<tr>
<th>Fragment / Marking</th>
<th>$\sigma_1^{(i)}$</th>
<th>$\sigma_2^{(i)}$</th>
<th>$\sigma_3^{(i)}$</th>
<th>$\sigma_4^{(i)}$</th>
<th>$\sigma_5^{(i)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context Fragment 1 ($c_1$)</td>
<td>$A_1$</td>
<td>$B_1$</td>
<td>$C_1$</td>
<td>$D_1$</td>
<td>$E_1$</td>
</tr>
<tr>
<td>Context Fragment 2 ($c_2$)</td>
<td>$A_2$</td>
<td>$B_2$</td>
<td>$C_2$</td>
<td>$D_2$</td>
<td>$E_2$</td>
</tr>
<tr>
<td>Relation Fragment 1 ($a_1$)</td>
<td>$P_1$</td>
<td>$P_2$, $P_3$</td>
<td>$P_4$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>Relation Fragment 2 ($a_2$)</td>
<td>$P_1$</td>
<td>$P_2$, $P_3$</td>
<td>$P_4$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
<tr>
<td>Relation Fragment 3 ($a_3$)</td>
<td>$P_1$</td>
<td>$P_2$, $P_3$</td>
<td>$P_4$, $P_5$</td>
<td>$P_6$</td>
<td>$-$</td>
</tr>
<tr>
<td>Relation Fragment 4 ($a_4$)</td>
<td>$P_1$</td>
<td>$P_2$, $P_3$</td>
<td>$-$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

fragment 1, $S_{c_1} = \{(c_1)\sigma_1, (c_1)\sigma_2, (c_1)\sigma_3, (c_1)\sigma_4, (c_1)\sigma_5\}$. $(c_1)\sigma_1$ denotes the marking in which place node $A_1$ in the context fragment contains a token.

$O_{c_1}$ denotes the set of relevant events to context fragment $c_1$, and is determined from the unique transition labels in the PN fragment definition. The fragment depicted in Figure 7 implies the following:

$$O_{c_1} = \{\text{Cashier In Entrance Zone, Cashier In FrontCounter Zone, Cashier In Safe Zone, Cashier In BackCounter Zone, Cashier Disappear}\}$$

Recall, that we create an instance of the context fragment for each activity role. Thus in our example, the set $O_{c_1}$ refers to events involving the Cashier object. A second set $O_{c_2}$ refers to events involving the Visitor object.

Set $M_i$, represents all relevant combinations of events in set $O_i$ that may occur. Clearly, $M_i \subset 2^{O_i}$. This set contains combinations of events that may occur simultaneously according to the definition of fragment $i$. In examining the context fragment in Figure 7, we note that all transitions are in conflict with one another. That is no two transitions may fire at the same time (the firing of one would disable the firing of
Let us now discuss the construction of the elements of the BRF for the context fragment depicted in Figure 7, \( c_1 \), whose initial state is \( (c_1)\sigma_1 \). The prior distribution is straightforward to construct given our initial state. We can define the following simple prior probability distribution over the marking space \( S_{c_1} \):

\[
P((c_1)x_{0(m)}) = \begin{cases} 
1 & \text{if } (c_1)x_{0(m)} = (c_1)\sigma_1 \\
0 & \text{otherwise}
\end{cases}
\]

Let us now illustrate the construction of the context dynamic model corresponding to fragment \( c_1 \). Recall that, \( i_{x_{t-1}}(m) \) denotes the marking of PN fragment \( i \) in the previous frame, \( i_{x_{t-1}}(e) \) denotes the event set that occurred in the previous frame, and \( i_{x_{t}}(m) \) denotes the resulting marking of fragment \( i \).

The first component of the context dynamic model is \( P((c_1)x_{t(m)}|(c_1)x_{t-1(m)},(c_1)x_{t-1(e)}) \), which describes the Petri Net fragment dynamics.

Let us examine the case of \( (c_1)x_{t-1(m)} = (c_1)\sigma_2 \), that is, in the previous time frame place node \( B_1 \) contains a token. By examining Figure 7 we can see that if event Cashier In FrontCounter Zone occurs (i.e. \( (c_1)x_{t-1(e)} = \{ \text{Cashier In FrontCounter Zone} \} \)) the resulting marking would be one where place \( C_1 \) contains a token \( (c_1)x_{t(m)} = (c_1)\sigma_3 \). These dynamics are captured as a discrete probability distribution as follows:

\[
P((c_1)x_{t(m)}|(c_1)x_{t-1(m)} = (c_1)\sigma_2, (c_1)x_{t-1(e)} = \{ \text{Cashier In FrontCounter Zone} \}) = \begin{cases} 
1 & \text{if } (c_1)x_{t(m)} = (c_1)\sigma_3 \\
0 & \text{otherwise}
\end{cases}
\]

32
We use the same process to define the remainder of

\[ P^{(c_1)x_t(m)}|^{(c_1)x_{t-1}(m)},^{(c_1)x_{t-1}(e)}; \]

<table>
<thead>
<tr>
<th>( (c_1)x_t(m) )</th>
<th>( (c_1)x_{t-1}(m) )</th>
<th>( (c_1)x_{t-1}(e) )</th>
<th>( (c_1)\sigma_1 #{{\text{In Entrance Zone}}} )</th>
<th>( (c_1)\sigma_2 #{{\text{In Entrance Zone}}} )</th>
<th>( (c_1)\sigma_1 #{{\text{In FrontCounter Zone}}} )</th>
<th>( (c_1)\sigma_2 #{{\text{In FrontCounter Zone}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (c_1)\sigma_1 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( (c_1)\sigma_2 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( (c_1)\sigma_3 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( (c_1)\sigma_4 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( (c_1)\sigma_5 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The second component of the context dynamic model, \( P^{(c_1)x_t(m)}|^{(c_1)x_{t}(m)}; \) defines a distribution over the relevant event sets, \( M_{c_1} \), that can occur at frame \( t \). Again consider the case of \( (c_1)x_t(m) = (c_1)\sigma_2 \).

Examining Figure 7 we see that four transitions, respectively labeled with events \textit{Cashier Disappear}, \textit{Cashier In BackCounter Zone}, \textit{Cashier In FrontCounter Zone}, \textit{Cashier In Safe Zone} are enabled in this marking. These correspond to the possible event sets when fragment \( c_1 \) takes on marking \( (c_1)\sigma_2 \). Recall also that the empty event set, \( \emptyset \), is enabled in all markings.

\[
\hat{P}^{(c_1)x_t(e)}|^{(c_1)x_t(m)} = (c_1)\sigma_2 = \begin{cases} 
1 & \text{if } (c_1)x_t(e) = \emptyset \text{ or } \{\text{Cashier Disappear}\} \text{ or } \{\text{Cashier In BackCounter Zone}\} \\
& \text{ or } \{\text{Cashier In FrontCounter Zone}\} \text{ or } \{\text{Cashier In Safe Zone}\} \\
0 & \text{otherwise} 
\end{cases}
\]

After the normalization step we get:

\[
P^{(c_1)x_t(e)}|^{(c_1)x_t(m)} = (c_1)\sigma_2 = \begin{cases} 
1/5 & \text{if } (c_1)x_t(e) = \emptyset \text{ or } \{\text{Cashier Disappear}\} \text{ or } \{\text{Cashier In BackCounter Zone}\} \\
& \text{ or } \{\text{Cashier In FrontCounter Zone}\} \text{ or } \{\text{Cashier In Safe Zone}\} \\
0 & \text{otherwise} 
\end{cases}
\]

The remainder of \( P^{(c_1)x_t(e)}|^{(c_1)x_t(m)} \) is calculated similarly:
Once the context dynamic model is constructed, it remains to construct the context measurement model.

Recall that for each event, \( O_i(j) \), \( j = 1 \ldots |O_i| \), we construct a separate model, \( P(y_t\mid x_t(e)) \).

Again let us consider the context fragment \( c_1 \), depicted in Figure 7. Suppose the first event in an arbitrary ordering of the elements of \( O_{c_1} \) is \( \text{Cashier In Entrance Zone} \), so that \( O_{c_1}(1) = \text{Cashier In Entrance Zone} \) and \( (c_1)y_{t(1)} \) is the certainty with which \( \text{Cashier In Entrance Zone} \) is observed at frame \( t \). Since, of all the sets in \( M_{c_1} \), \( O_{c_1}(1) = \text{Cashier In Entrance Zone} \) is only a member of one of the sets (i.e \{Cashier In Entrance Zone\}) we set

\[
P((c_1)y_{t(1)}\mid (c_1)x_t(e)) = \begin{cases} 
\varphi((c_1)y_{t(1)}) & \text{if } (c_1)x_t(e) = \{\text{Cashier In Entrance Zone}\} \\
1 - \varphi((c_1)y_{t(1)}) & \text{otherwise}
\end{cases}
\]

where \( \varphi(\cdot) \) is the sigmoid function defined in Equation 14.

Similarly, the remainder of the measurement model is constructed:

\[
P((c_1)y_{t(j)}\mid (c_1)x_t(e)):
\]

<table>
<thead>
<tr>
<th>( O_{c_1} )</th>
<th>( (c_1)x_t(e) )</th>
<th>( (c_1)y_{t(1)} )</th>
<th>( (c_1)y_{t(2)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Cashier In Entrance Zone}</td>
<td>1 - \varphi((c_1)y_{t(1)})</td>
<td>\varphi((c_1)y_{t(1)})</td>
<td>1 - \varphi((c_1)y_{t(2)})</td>
</tr>
<tr>
<td>{Cashier In FrontCounter Zone}</td>
<td>1 - \varphi((c_1)y_{t(1)})</td>
<td>1 - \varphi((c_1)y_{t(2)})</td>
<td>\varphi((c_1)y_{t(2)})</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

### 6.1.1 Inference

Now let us use the above to make the particle filter mechanism more concrete. We shall use an example with \( N = 3 \) particles. These particles are factored over the number of PN fragments in our example.

Before beginning our inference process we have to initialize our particle set for each factor of our model. Recall from section 5.3 that this is achieved by first sampling the marking, \( x_0(m) \), from \( P(x_0(m)) \) and then using the sampled value, sample the event combination, \( x_{0(e)} \), from \( P(x_{0(e)}\mid x_0(m)) \).

\[\vartheta \]

Technion - Computer Science Department - Technical Report  CIS-2012-07 - 2012
Let us again consider the context fragment $c_1$ shown in Figure 7. The marking component of the three particles in our example is initialized by sampling from $P^{(c_1)x_0(m)}$. Since by the construction of this distribution all probability mass is concentrated on the initial marking $(c_1)\sigma_1$, this marking will be the outcome of these samplings. Now we sample the event combination of each particle using $P^{(c_1)x_0(e)}$. Examining the construction of this probability distribution above, we can see that each of the event combinations: $\emptyset$, \{Cashier In Entrance Zone\}, \{Cashier In FrontCounter Zone\}, \{Cashier In Safe Zone\}, \{Cashier In BackCounter Zone\} is given equal probability (1/5). Thus, one likely result of our initialization would be the following particle assignments:

$$(c_1)x_0^{(1)} = (c_1)\sigma_1, \{\text{Cashier In BackCounter Zone}\},$$

$$(c_1)x_0^{(2)} = (c_1)\sigma_1, \emptyset,$$

$$(c_1)x_0^{(3)} = (c_1)\sigma_1, \{\text{Cashier In FrontCounter Zone}\}.$$  

where $x_0^{(1)} = (c_1)\sigma_1, \{\text{Cashier In BackCounter Zone}\}$ is shorthand for $x_0^{(1)} = (c_1)\sigma_1, x_0^{(1)} = \{\text{Cashier In BackCounter Zone}\}$. 

We also initialize the weights to $1/N$ (recall $N = 3$ in our example)

$$(c_1)w_0^{(1)} = 1/3, (c_1)w_0^{(2)} = 1/3, (c_1)w_0^{(3)} = 1/3$$

We follow the same procedure to initialize all other context and activity factors. For example one possible initialization for the particles pertaining to activity factor $a_1$, which we shall refer to later in this example is given by:

$$(a_1)x_0^{(1)} = (a_1)\sigma_1, \emptyset, (a_1)x_0^{(2)} = (a_1)\sigma_1, \{\text{trans}_1\}, (a_1)x_0^{(3)} = (a_1)\sigma_1, \{\text{trans}_1\}$$
After the initialization is complete we can evaluate new observations in an online fashion using our particle filter framework. For the sake of this example let us focus on how this is achieved in fragments \( c_1 \) and \( a_1 \).

Let us suppose our event recognition component for the first frame of the video considers the event \{Cashier in BackCounter Zone\} to have occurred with 0.8 certainty, and no other events to have occurred. Thus the observation vector \( (c_1)y_1 \) will have 5 components (\( O_{(c_1)} \) has 5 members), all of which will have a 0 certainty, save for component \( k \) which represents the event Cashier In BackCounter Zone (i.e. \( (c_1)y_{1(k)} = 0.8 \)).

Continuing our example, since \( (c_1)x_{1}^{(1)} = \{(c_1)\sigma_1, \{\text{Cashier In BackCounter Zone}\}\} \):

\[
P((c_1)y_{1(j)}|(c_1)x_{1}^{(1)}) = \begin{cases} 
\varphi((c_1)y_{1(j)}) & \text{if } j = k \\
1 - \varphi((c_1)y_{1(j)}) & \text{otherwise}
\end{cases}
\]

where \( k \) represents the index of event \{Cashier In BackCounter Zone\}.

Thus since \((c_1)y_{1(j)} = 0\) for all \( j \neq k \) and \((c_1)y_{1(k)} = 0.8\)

\[
P((c_1)y_{1}^{(1)}|(c_1)x_{1}^{(1)}) = \prod_{j=1}^{\left|O_{(c_1)}\right|=5} P((c_1)y_{1(j)}|(c_1)x_{1}^{(1)}) = \varphi(0.8) \cdot (1 - \varphi(0))^4
\]

Updating the weights is then done according to:

\[
(c_1)\hat{w}_{1}^{(1)} = (c_1)w_{0}^{(1)} \cdot P((c_1)y_{1}^{(1)}|(c_1)x_{1}^{(1)}) = (c_1)w_{0}^{(1)} \cdot \varphi(0.8) \cdot (1 - \varphi(0))^4 = .3091
\]

where we are using

\[
\varphi(z) = \frac{1}{1 + e^{-h(z-0.5)}}
\]

with \( h = 10 \).

We adjust the other particle weights similarly:

\[
(c_1)\hat{w}_{1}^{(2)} = (c_1)w_{0}^{(2)} \cdot P((c_1)y_{1}^{(2)}|(c_1)x_{1}^{(2)}) = (c_1)w_{0}^{(2)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^4 = .0154
\]
\[(c_1)\hat{w}_1^{(3)} = (c_1)w_0^{(3)} \cdot P((c_1)y_1|^{(c_1)}x_1^{(3)}) = (c_1)w_0^{(3)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^3 \cdot \varphi(0) = 1.03e - 4\]

Our final step is to normalize our weights:

\[
\sum_{i=1}^{N=3} (c_1)\hat{w}_1^{(i)} = .3246
\]

\[
(c_1)w_1^{(1)} = \frac{(c_1)\hat{w}_1^{(1)}}{.3246} = .9522
\]

\[
(c_1)w_1^{(2)} = \frac{(c_1)\hat{w}_1^{(2)}}{.3246} = .0474
\]

\[
(c_1)w_1^{(3)} = \frac{(c_1)\hat{w}_1^{(3)}}{.3246} = .0001
\]

After the correction step for the context fragment above is complete, we perform the prediction step. That is we use the current context state estimation to sample from the context proposal distribution, \(P_c\), to derive an updated estimation of the context state.

The prediction for fragment \(c_1\) is done in two phases: initially we sample from \(P((c_1)x_{t(e)}|^{(c_1)}x_{t-1(m)}^{(c_1)}x_{t-1(e)})\), and then, using the sampled result, we sample from \(P((c_1)x_{t(e)}|^{(c_1)}x_{t(m)})\).

Continuing our example, the marking component of particle 1, \(^{(c_1)}x_{1(m)}^{(1)} \sim P((c_1)x_{t(m)}|^{(c_1)}x_{t-1(e)} = (c_1)\sigma_1, (c_1)x_{t-1(e)} = \{\text{Cashier In Back Counter Zone}\})\) would likely result in \((c_1)x_{1(m)}^{(1)} = (c_1)\sigma_5\) (the only outcome with non-zero probability). Then, \((c_1)x_{1(e)}^{(1)} \sim P((c_1)x_{t(e)}|^{(c_1)}x_{t(m)} = (c_1)\sigma_1)\) could result in any one of five outcomes, each with equal probability \(\{\emptyset, \{\text{Cashier In Front Counter Zone}\}, \{\text{Cashier In Back Counter Zone}\}, \{\text{Cashier In Safe Zone}\}, \{\text{Cashier Disappear}\}\)).

Similarly the other particle markings are given by: \((c_1)x_{1(m)}^{(2)} = (c_1)\sigma_1, \) and \((c_1)x_{1(m)}^{(3)} = (c_1)\sigma_4\) thus, let us assume one of several possible result of such a sampling:

\[
^{(c_1)}x_1^{(1)} = \{(c_1)\sigma_5, \emptyset, (c_1)\sigma_1, \{\text{In Safe Zone}\}\}, \; ^{(c_1)}x_1^{(2)} = \{(c_1)\sigma_1, \{\text{In Safe Zone}\}\}, \; ^{(c_1)}x_1^{(3)} = \{(c_1)\sigma_4, \{\text{In Back Counter Zone}\}\}
\]
After the update and prediction of the context fragments, we perform the update and prediction of the activity fragments. The update step of the activity fragments differs slightly from the update step of the context fragments, in that we rely on marginal probabilities computed from the context fragment to perform the update. More specifically, we correct the weight of each of the activity fragment particles by multiplying by the marginal probability of the context condition.

For example, consider particle \( (a_1)x_0^{(1)} \) (initialized earlier). The marking variable \( (a_1)x_0(m) = (a_1)\sigma_1 \) and the event set variable \( (a_1)x_0(e) = \{trans_1\} \), therefore we are interested in \( P(C((a_1)\sigma_1, \{trans_1\}) = P(\neg E_1 \land \neg D_1) \). Recall that \( C(i_{x_t(m)}, i_{x_t(e)}) \) is our notation for referring to the context condition given to a transition with label \( i_{x_t(m)} \) enabled in marking \( i_{x_t(m)} \).

This quantity is given by the following formula (see section 5.3.3):

\[
P(\neg E_1 \land \neg D_1) = P(E_1) \cdot (1 - P(D_1))
\]

where \( P(E_1) \) is the probability place node \( E_1 \) in context fragment \( c_1 \) contains a token, and \( P(D_1) \) is the probability that place node \( D_1 \) in context fragment \( c_1 \) contains a token.

Examining Table 1 we see that marking \( (a_1)\sigma_4 \) is the only context marking where place node \( D_1 \) contains a token. Furthermore, marking \( (a_1)\sigma_5 \) is the only marking where place node \( E_1 \) contains a token.

using the formula:

\[
P(E_1) = \sum_{j=1}^{N} (1_{E_1}(i_{x_t(m)}) \cdot (c_1)x_0^{(j)}) \cdot (c_1)w_0^{(j)} = 1 \cdot (c_1)w_0^{(1)} + 0 \cdot (c_1)w_0^{(2)} + 0 \cdot (c_1)w_0^{(3)} = (c_1)w_0^{(1)} = .9522
\]

This is due to the fact that \( 1_{E_1}(i_{x_t(m)}) = 1 \) for all \( j \neq 1 \) (i.e. place \( E_1 \) does not contain a token in these markings).

Similarly:

\[
P(D_1) = \sum_{j=1}^{N} (1_{D_1}(i_{x_t(m)}) \cdot (c_1)x_0^{(j)}) \cdot (c_1)w_0^{(j)} = 0 \cdot (c_1)w_0^{(1)} + 0 \cdot (c_1)w_0^{(2)} + 1 \cdot (c_1)w_0^{(3)} = (c_1)w_0^{(3)} = .0001
\]
Thus:

\[ P(\neg E_1 \land \neg D_1) = P(E_1 \cdot (1 - P(D_1))) = .9522 \cdot (1 - .0001) = .9521 \]

We use this value to update our activity weight:

\[ (a_1)\hat{w}_1^{(1)} = (a_1)\hat{w}_0^{(1)} \cdot \frac{P(\neg E_1 \land \neg D_1)}{P((a_1)x_0^{(1)}|[a_1]x_0^{(1)}]} = 1/3 \cdot \frac{.9521}{1/5} = 1.587 \]

The remaining particle weights are updated in the same manner

\[ (a_1)\hat{w}_1^{(2)} = (a_1)\hat{w}_0^{(2)} \cdot \frac{1 - P(\neg E_1 \land \neg D_1)}{P((a_1)x_0^{(2)}|[a_1]x_0^{(2)}]} = 1/3 \cdot \frac{.0479}{1/5} = .08 \]

\[ (a_1)\hat{w}_1^{(3)} = (a_1)\hat{w}_0^{(3)} \cdot \frac{1 - P(\neg E_1 \land \neg D_1)}{P((a_1)x_0^{(3)}|[a_1]x_0^{(3)}]} = 1/3 \cdot \frac{.0479}{1/5} = .08 \]

We then normalize in the same manner as above

\[ \sum_i (a_1)w_i^{(i)} = 1.683 \]

\[ (a_1)w_1^{(1)} = \frac{(a_1)\hat{w}_1^{(1)}}{1.683} = .9430 \]

\[ (a_1)w_1^{(2)} = \frac{(a_1)\hat{w}_1^{(2)}}{1.683} = .0284 \]

\[ (a_1)w_1^{(3)} = \frac{(a_1)\hat{w}_1^{(3)}}{1.683} = .0284 \]

After the update step the prediction step of the activity fragment proceeds similar to the prediction step of the context fragment. For completion we will carry it this step in our example.

Continuing our example, particle \( (a_1)x_1(m) \sim P((a_1)x_1(m)) ) \) would likely result in \( (a_1)x_0^{(1)}(m) = (a_1)\sigma_1 \) (the only outcome with non-zero probability). Note that in marking \( (a_1)\sigma_1 \),
$trans_1$ is the only enabled transition. Thus, $x_1^{(a_1)(c)(1)} \sim P(x_1^{(a_1)(c)}|x_1^{(m)}) = (a_1)\sigma_1$ could result in one of two outcomes, each with equal probability ($\emptyset$, $\{trans_1\}$).

Similarly the other particle markings are given by: $x_1^{(a_1)(2)} = (a_1)\sigma_2$, and $x_1^{(a_1)(3)} = (a_1)\sigma_2$

Thus, a plausible result of the prediction step yields the following particle values:

$\begin{align*}
(a_1)x_1^{(1)} &= \{(a_1)\sigma_1, \{trans_1\}\}, \\
(a_1)x_1^{(2)} &= \{(a_1)\sigma_2, \{trans_2\}\}, \\
(a_1)x_1^{(3)} &= \{(a_1)\sigma_2, \emptyset\}
\end{align*}$

7 Experiments

In this section we illustrate the benefits of our approach over several datasets. We are able to show both an increase in recognition performance and a reduction in the complexity of our models.

7.1 System Description

For the empirical validation of our method we constructed an experiment system with several components (shown in Figure 9). The raw image data (input #1) is fed to Module #1 (Object Detection and Tracking) to generate a list of objects and their corresponding locations at each frame. This information is then input into Module #2 (Event Recognition). Specifications of scene information (Input #2), such as zone definitions and camera calibration information, are also input into Module #2. Module #2 outputs a list of events and their corresponding certainty values that occur at each frame. The model specifications (input #3), along with the output of Module #2, are input into Module #3 (Activity Recognition). This module outputs a list of recognized activities in the video sequence. We devote the following few paragraphs to elaborate on the implementation details of each of these modules in our experiments.

Module #1 (Object Detection and Tracking) takes as input raw video (a single image per frame) and outputs a list of objects and their per frame location. In our experiments, we used the ground truth locations on the synthetic bank dataset, a particle filter tracker [16] on the ETISEO dataset, and a tracker based on boosting [6] for the Technion dataset. We used a semi-automatic approach to tracking, where if an object

40
Figure 9: Flowchart of System
track had been lost by the tracker, the tracker was again reinitialized using the ground truth. The result of this process is a mostly accurate but at times noisy tracker output. The Technion dataset was also evaluated using “ideal” tracking conditions, to allow evaluation of the activity recognition without the influence of observation uncertainty.

Module #2 (Event Recognition) takes as input the tracked locations of Module #1 as well as scene specifications and outputs a list of events and corresponding certainties per frame. We chose to implement a simple rule based event recognition similar to the one adopted in [35]. In our experiments we recognize several kinds of events including: Appearance, Disappearance, In Zone. We make use of the scene specification to recognize these events and associate the appropriate certainty value. For instance an In Zone event recognition certainty is calculated as a function of the zone bounding box specification and the object location. The Euclidean distance, $d$, of the object from the zone center is calculated and the certainty is then calculated by the formula:

$$
\psi = 1 - \frac{1}{1 + e^{-v_{zone} \cdot d}}
$$

where $v_{zone}$ is a parameter provided in the scene specification.

Module #3 (Activity Recognition) takes as input the activity model specification along with the output of Module #2, a list of events and their certainties. In our experiments we implemented four different types of activity recognition approaches. The Store Totally Recognized Scenarios (STRS) approach is implemented based on [35] and represents deterministic approaches to activity recognition. In this approach, event inputs were thresholded and only those whose certainties were above a threshold were passed into the activity recognition module. The Propagation Net (P-Net) approach is implemented based on [33] and represents competing probabilistic approaches to activity recognition. Note that the P-Net approach relies on an available duration model for each of the events in the activity. The third approach considered is the Particle Filter Petri Net (PFPN)[22]. This approach represents a competing Petri Net construction in which each activity is modeled recognized independently of all other. The Factored Particle Filter Petri
Net (FPFPN) is our approach and is implemented using the methods discussed above. The models used by all approaches were specified to represent the various activities of interest in the event domain, in particular to capture the temporal ordering of their composing events. While the modeling strength of the various approaches is not equivalent (i.e. some modeling approaches can capture the types of relationships we are interested in with a simpler model), we ensured that the models describing the same activity in different modeling formalisms were equivalent (not adding or removing constraints because they are inconvenient to model).

### 7.2 Metrics For Performance

In order to give performance results that allow for comparison of the various activity recognition approaches, we must determine when an activity has been recognized. For those approaches that output a certainty associated with each recognition (e.g. P-Net, PFPN and FPFPN) we simply threshold this value. In all of our experiments an activity is considered to be recognized if its associated certainty is above some threshold $\theta$. Since at each frame we may have a different activity recognition certainty we simply threshold the maximum certainty for this activity over the length of the video sequence.

Comparing the activity recognition output to the available ground truth allows us to compute the number of true positives, true negatives, false positives, and false negatives. Since each clip has only a few activities occurring in it, the number of positive examples is significantly smaller than the number of negative examples. This is also the case in real surveillance systems. For this reason an approach may achieve a fairly high accuracy score by classifying all examples as negative. Thus when evaluating each approach it is important to take into account the tradeoff between true and false positive recognition rates. To this end, we have chosen to present the results as an ROC curve, which plots the true positive rate (also known as recall) against the false positive rate. These metrics are given by the formulas:

$$\text{true positive rate} = \frac{tp}{(tp + fn)}$$
false positive rate = \frac{fp}{(fp + tn)}

Clearly, as we decrease our recognition threshold, \( \theta \), from 1.0 to 0.0 we will achieve a higher number of both true and false positives, causing both above rates to rise. An ideal threshold selection would achieve a high true positive rate and a low false positive rate. Generally, a tradeoff exists between these rates and we must choose a threshold to favor one or the other. The ROC curve presentation illustrates this tradeoff for a number of values of \( \theta \), for each of the algorithms we have evaluated in this paper.

It should be clarified that the STRS algorithm is not a stochastic algorithm, thus its decision is binary and cannot be thresholded. To compile the curve for this algorithm we instead varied the threshold for event recognition.

### 7.3 Metrics for Complexity

In particle filter approaches the temporal complexity is a function of the number of particles used. The number of particles must be increased as the state space becomes more complex. Thus a decrease in the complexity of the state space can be translated into an improvement in efficiency. We have chosen an additional metric to quantify this improvement. We plot the recognition rate (expressed as an f-score) of the various approaches against the number of particles used. A recognition rate which remains steady as the number of particles decreases implies that using less particles (i.e. computational resources) would yield similar performance results. Hence such a method can be said to be able to scale better as activities and contexts become more complex. Specifically, the f-score is computed from the precision and recall as follows:

\[
\text{f-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]
Figure 10: Example Sequence from synthetic Bank Dataset. A Bank Attack activity is taking place. Sequence should be interpreted left to right, top to bottom.

7.4 Datasets

Unfortunately, a surveillance video dataset with annotated activities is time consuming to compile. Recent public datasets pertaining to surveillance video, such as VIRAT [28], emphasize event recognition and do not include repeated non-trivial activities such as the ones we consider in this work. Existing activity datasets have only a few examples of each activity. Thus, in addition to using a publicly available dataset, we captured and annotated our own dataset. In order to allow evaluation of our approach on a large number of examples we created additional synthetic animated video clips. These clips can be created at considerably less cost than real data and serve to further illustrate the effectiveness of our approach. Example activity clips from the datasets used in our experiments can be viewed at [1].

7.4.1 Synthetic Bank Dataset

In the first set of experiments we considered 500 short synthetic video clips lasting from 274-526 frames (9-17 seconds) each. Based on [3] we considered the activities (1) Bank Attack (2) Attempted Bank Attack (3) Normal Customer Interaction (4) Cashier accesses safe (5) Outsider enters safe. Most of the clips contain one or more of these activities. An example sequence from this dataset is shown in Figure 10.
Figure 11: Example Sequence from ETISEO Building Sequence Dataset. A *Depart By Car* activity is taking place. Frames are taken from four different camera angles. Sequence should be interpreted left to right, top to bottom.

Figure 12: Example Sequence from Technion Parking Lot Dataset. An *Arrive By Car* activity is occurring simultaneously with a *Car Theft* activity taking place. Sequence should be interpreted left to right, top to bottom.

Figure 13 shows the quantitative results obtained on the synthetic Bank Dataset. Note that for these experiments we applied the P-Net approach twice. In the first run a duration model for each of the events in the activity was used. In the second run an uninformative duration model was used.

### 7.4.2 ETISEO Building Entrance Dataset

In order to evaluate our approaches on an existing, publicly available set of video sequences, we chose the ETISEO Building Entrance dataset [27]. This dataset is a publicly available dataset of real videos which includes multiple camera views of the scene and includes several non-trivial activities. The ground truth of object locations across the different camera locations is available for download. The ground truth activity labeling was manually annotated by the authors. This dataset contains 6 sequences from up to 4 camera angles (though not all sequences contain data for all cameras angles). We defined 6 activities that can take place in this domain: (1) *Arrive By Car*, (2) *Arrive On Foot*, (3) *Depart by Car*, (4) *Depart on Foot*, (5) *Meet and Walk Together*, and (6) *Meet and Walk Apart*. Note that different scene objects may be involved in one or more activities. The various camera angles provide an additional challenge to the event
recognition module: how to integrate event recognitions from different camera angles. In our experiments we chose a simple approach of merging events from all camera angles into a single list before inputting this list into the activity recognition module. In the case where the same event is detected in multiple camera angles, we choose the event with the highest certainty value. The sequences in this dataset ranged from 924 to 1649 frames in (30-54 seconds) in length. Each sequence contained one or more of the activities. An example sequence from this dataset is shown in Figure 11. Figure 14 shows the quantitative results obtained on the ETISEO Building Entrance Dataset.

7.4.3 Technion Parking Lot Dataset

The Technion Parking Lot dataset was captured and manually annotated by the authors. This dataset is intended to expand the breadth of our experiments on real data. We captured over two hours of surveillance footage with multiple objects and several activities. Several staged activities were acted out in order to test our system’s ability to recognize rarely occurring activities. These include: (1) Car Theft - a person approaches several vehicles before finally entering a final vehicle and driving off, (2) Break In- A person enters a car and then exits without driving away, (3) Arrive by car , (4) Depart by Car , and (5) Drop Off events. For the latter four activities our dataset contains both staged and real occurrences of the activities. In this dataset there were a total of 24 activities ranging from 43 to 207 seconds in length, with a median length of 100 seconds. Each activity includes at least two and up to four objects. An example sequence from this dataset is shown in Figure 12 (clips may be viewed at [1] ). This dataset proved to be the most challenging and all approaches yielded lower recognition results when applied to it. Figure 15 shows the ROC curve obtained on the Technion Parking Lot Dataset. Figure 16 shows the results on this same dataset under “ideal” tracking conditions (i.e. the tracking information was taken directly from ground truth annotation of object locations.)
7.5 Results and Discussion

The FPFPN approach outperforms the baseline methods in our experiments (see AUC results in Table 2). We attribute this performance advantage to the core intuition that distinguishes FPFPN from the other methods. Namely, the modeling of what is possible in the scene (i.e. context) is separate from the modeling of the activity structure. Thus, the observation is used to estimate the context, which in turn is used to estimate the activity state. This is in contrast to the competing approaches which generally seek to estimate the activity state directly from the observation, resulting in a more complex model which makes successful state estimation more difficult.

The results shown in Figures 13 – 16 were all conducted using \( N = 2000 \) particles. The number of particles is the critical factor in determining the running time of the algorithm (this can be understood by inspecting the inference procedure in section 5.3.2). For this reason, it is interesting to examine the performance of the various approaches as the number of particles is reduced. Figures 17 – 19 show a plot of the f-score of each method as the number of particles is varied. The f-score represents the harmonic average of the precision and recall scores for the best threshold value. It is apparent from the figure that the results achieved by FPFPN can be also be achieved with significantly less particles (5% of the amount used in our experiments). This observation implies that a FPFPN activity recognition approach can be tuned to achieve significantly smaller running times than the competing formalisms without sacrificing the recognition performance.

The results in the figure are due to the factorization of the state space into simple fragments which have a small number of markings and events (recall that the state of the Bayesian Recursive Filter is composed of all combinations of the underlying Petri Net’s markings and transitions). Because each factor has a relatively small state space (e.g. Figures 7, 8) a small number of particles is needed to cover this space. Since we estimate the state of each factor independently, we require a small number of (factored) particles to estimate the joint state (context and activity) of the entire system. Thus a small number of particles will provide as good recognition performance as a larger number. This is in contrast to other approaches,
such as PFPN, which model an activity as a single fragment, which takes into account all possible events including those which are not part of the activity. In addition to being complex and difficult to construct, these models also imply a significantly larger state space than that of the small fragments created by FPFN. Such a space requires more particles to span, and is the reason that recognition performance deteriorates as the number of particles used decreases.
Figure 15: ROC Curve Comparing Recognition Performance for the Technion Dataset (Real Tracking)

Figure 16: ROC Curve Comparing Recognition Performance for the Technion Dataset (Perfect Tracking)

<table>
<thead>
<tr>
<th>Method</th>
<th>Bank</th>
<th>ETISEO</th>
<th>Technion (Real)</th>
<th>Technion (Perfect)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factored Particle Filter PN</td>
<td>0.9991</td>
<td>0.9166</td>
<td>0.8701</td>
<td>0.9355</td>
</tr>
<tr>
<td>Particle Filter PN</td>
<td>0.92285</td>
<td>0.82672</td>
<td>0.80766</td>
<td>0.91036</td>
</tr>
<tr>
<td>Propagation Net</td>
<td>0.71401</td>
<td>0.7672</td>
<td>0.76389</td>
<td>0.90007</td>
</tr>
<tr>
<td>Constraint Propagation</td>
<td>0.93858</td>
<td>0.75995</td>
<td>0.72278</td>
<td>0.84328</td>
</tr>
</tbody>
</table>

Table 2: Area Under The Curve
Figure 17: F-Score vs. Number of Particles for the Bank Data Set

Figure 18: F-Score vs. Number of Particles for the Technion Data Set (real tracking)
8 Conclusion

This work considers the problem of recognizing activities in surveillance video. Activities are high-level non-atomic semantic concepts which may have complex temporal structure. Activities are not easily identifiable using image features, but rather by the recognition of their composing events. Unfortunately, these composing events may only be observed up to a particular certainty.

The Factored Particle Filter Petri Net (FPFPN) approach described in this paper is based on the intuition that the activity model should be decomposed into two components: the context and activity. The advantage of a factored state space, such as the one proposed in our approach, is that we can estimate the state of each factor independently. Each factor in our system is relatively simple (only a few states and events) the number of particles needed to perform effective state estimation in each factor is small. Thus, the number of particles required overall decreases. In our experiments we are able to show that the effectiveness of our approach improves on competing approaches (PFPN and P-Net). We also show that the recognition rate for FPFPN remains steady as the number of particles decreases, while the effectiveness of the competing approaches deteriorates. This result implies that our proposed approach can be considered
more efficient than baseline approaches.

References


