STAIRS 2014
U. Endriss and J. Leite (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-421-3-280

Towards Learning and Classifying Spatio-Temporal Activities in a Stream Processing Framework

Mattias TIGER [a] and Fredrik HEINTZ [a]
[a] *Department of Computer and Information Science, Linköping University, Sweden* [1]

Abstract. We propose an unsupervised stream processing framework that learns a Bayesian representation of observed spatio-temporal activities and their causal relations. The dynamics of the activities are modeled using sparse Gaussian processes and their causal relations using a causal Bayesian graph. This allows the model to be efficient through compactness and sparsity in the causal graph, and to provide probabilities at any level of abstraction for activities or chains of activities. Methods and ideas from a wide range of previous work are combined and interact to provide a uniform way to tackle a variety of common problems related to learning, classifying and predicting activities. We discuss how to use this framework to perform prediction of future activities and to generate events.

Keywords. activity recognition, machine learning, knowledge representation, situation awareness, knowledge acquisition, unsupervised learning

Introduction

Learning and recognizing spatio-temporal activities from streams of observations is a central problem in artificial intelligence. Activity recognition is important for many applications including detecting human activities such that a person has forgotten to take her medicine in an assisted home [1], monitoring telecommunication networks [2], and recognizing illegal or dangerous traffic behavior [3,6]. In each of the applications the input is a potentially large number of streams of observations coming from sensors and other information sources. Based on these observations common patterns should be learned and later recognized, in real-time as new information becomes available.

The main contribution of this paper is an unsupervised framework for learning activities and causal relations between activities from observed state trajectories. Given a set of state space trajectories, the proposed framework segments the continuous trajectories into discrete activities and learns the statistical relations between activities as well as the continuous dynamics of each activity. The dynamics of the activities are modeled using sparse Gaussian Processes and their contextual relations using a causal Bayesian graph. The learned model is sparse and compact and supports both estimating the most likely activity of an observed object, and predicting the most likely future activities.

[1]Corresponding Authors: {Mattias Tiger, Fredrik Heintz}, Department of Computer and Information Science, Linköping University, Sweden; E-mail: {matti166@student.liu.se, fredrik.heintz@liu.se}.

As an example, consider a traffic monitoring application where a UAV is given the task of monitoring the traffic in a T-crossing. The UAV is equipped with a color and a thermal camera and is capable of detecting and tracking vehicles driving through the intersection [6]. The output of the vision-based tracking functionality consists of streams of states where a state represents the information about a single car at a single time-point and a stream represents the trajectory of a particular car over time (Figure 1, left). Based on this information the UAV needs to learn traffic activities such as a car driving straight through the intersection or making a turn (Figure 1, right).

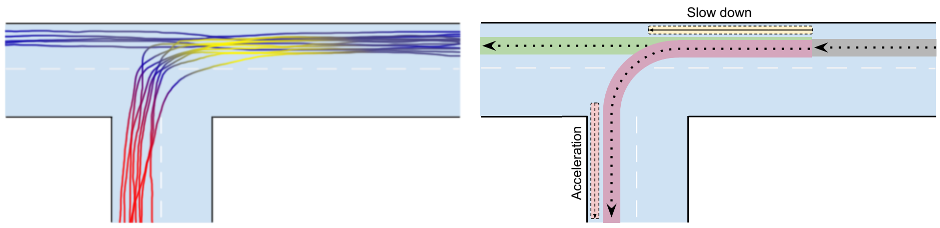


Figure 1. An illustrative example of the expected input and output of the proposed framework. The left figure shows observed trajectories. Blue is constant velocity, yellow is negative acceleration, and red is positive acceleration. The right figure shows a conceptual illustration of what we want to learn.

1. Related Work

There are many different approaches to activity recognition, both qualitative and quantitative. Here we focus on quantitative approaches since they are most related to our work. As far as we know there is no other unsupervised framework that can learn the atomic activities, the dynamics of each activity and the causal relations between the activities.

A common approach is to use images as the raw input. One approach uses a Bayesian Hierarchical Model with moving pixels from video surveillance as input [17]. They divide the image into square regions which the pixels belong to and use together with four different motion directions as a code book. They treat 10 second video clips as documents and the moving pixels as words and learn a Hierarchical Dirichlet Process. By performing word document analysis, the resulting topics are normative atomic activities in the form of pixel motion trajectories in the 2D image.

Our unsupervised framework in comparison requires trajectories, but can learn primitive activities bottom up from these trajectories. This allow our framework to rapidly learn new activities, while being able to compare their support by each activity's accumulated evidence. We also learn the causal statistical connections between activities allowing for activity prediction. Our framework could use moving pixels as input, but also other types of information such as 3D-positions. It also incorporates the uncertainties of those measurements.

Another approach is to use splines with rectangular envelopes to model trajectories [10,5], compared to our approach and the approach of Kim [8] which use Gaussian Processes (GPs) in a continuous Bayesian setting. Similar to our approach [5] divides trajectories into shorter segments and split trajectories at intersections, while [8] only considers trajectories beginning and ending out of sight and performs no segmentation of activities. Neither of the frameworks mentioned are suitable in a stream processing context with incrementally available information in the learning phase. Our framework can continue to learn in the field on a robotic platform and adapt directly to new situations in regards to learning, classification and prediction. Our framework supports not only detecting and keeping count of abnormal behaviors, but also learns those behaviors and collect statistics on their detailed activities.

2. The Activity Learning and Classification Framework

The proposed framework (Figure 2) learns activities and their relations based on streams of temporally ordered time-stamped probabilistic states for individual objects in an unsupervised manner. Using the learned activities, the framework classifies the current trajectory of an object to belong to the most likely chain of activities. The framework can also predict how abnormal the current trajectory is and how likely future activities are.

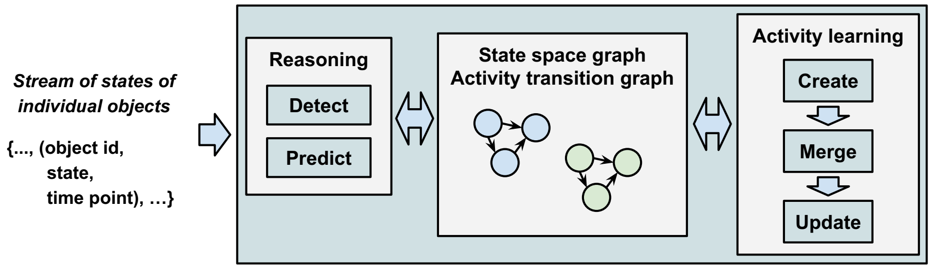


Figure 2. An overview of the framework. It is built around the three modules *Reasoning*, *Activity Learning* and the knowledge base consisting of two graphs, of which the two former modules make extensive use of. The *Activity learning* takes entire observed trajectories and updates the graphs. It is divided into three parts which are performed sequentially. The *Reasoning* module detects and predicts activities based on observations.

An activity can informally be defined as something a particular object does. In our case, an instance of an activity is represented by a state trajectory where some state variables are expected to change in a certain way while others may change freely. For example, an activity could be *slow down* where the velocity state variable is expected to decrease. An activity model should capture such continuous developments.

The framework models activities as Gaussian Processes, which are sufficiently expressive to model the development of the state variables including uncertainties in observations. Activities are learned as edges in a graph, where the nodes are intersection points, in the state-space, between activities. On a higher abstraction level activities are seen as nodes, with edges representing transitions which are weighted according to the observed empirical probability of the transitions. The latter construct enables prediction of activities through probabilistic inference. The graphs are called the *State Space Graph* and the *Activity Transition Graph*. An example is shown in Figure 3. The Activity Transition Graph is a causally ordered discrete Markov chain. We currently only consider 1-order Markovian transitions in the formulation of this graph. However, a natural extension is to use a higher order Markov chain.

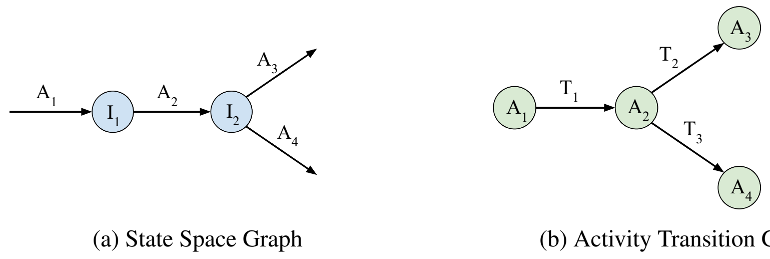


Figure 3. A denotes an activity in both the State Space Graph and the Activity Transition Graph, I denotes an intersection point in the state space, T denotes a transition in the Activity Transition Graph. All transitions between A_2, A_3 and A_4 can be seen as contained within the intersection point I_2.

To handle activities on different scales the framework uses two parameters $s_{threshold}$ and $l_{threshold}$. The parameter $s_{threshold}$ is a threshold on the similarity between activities in the state space, based on the likelihood of one explaining the other. If an observed trajectory has a similarity below the threshold when compared to any other activity, the observation is considered novel and unexplained. The parameter $l_{threshold}$ is a threshold on the minimal length of an activity in the state space, which is used to limit the amount of detail in the learned model.

The framework captures and models activities as statistically significant trajectories through the state space. With enough observations all activities captured are discriminative and invariant of the observed variations which are not statistically significant.

In the work of [8] they detect abnormal trajectories by using the lack of dominant trajectory models in explaining observed trajectories. In our case a similar lack of dominance of activity models, or that several are almost equally likely, can be due to two things. Either due to abnormality (novelty for learning) or due to a transition between activities. By using the similarity threshold $s_{threshold}$ we can distinguish between intervals that are either abnormal or transitional.

3. Modeling Activities

An activity is seen as a continuous trajectory in the state space, i.e. a mapping from time t to state y. A state can for example be a 2D position and a 2D velocity. To get invariance in time, t is normalized to be between 0.0 and 1.0 and acts as a parametrization variable of $y = f(t)$. The mapping is modeled using Gaussian Processes. The assumption is that an activity is a curve in a N-dimensional state space. The activity model is the result of the continued accumulation of explained observed trajectories. The amount of these supporting trajectories is seen as the amount of evidence E of the activity model, and is used in comparison with other activity models.

Given an observed state, this parametrization variable must be known to calculate how well the activity explains the observation. In [8] they assume this parametrization to be known by assuming that it is already normalized to the interval $[0, 1]$ which can only occur after the whole trajectory has been observed. Further, they only consider entire trajectories without temporal segmentation, and with a state space consisting of positions and velocities. When performing stream reasoning the entire trajectory cannot be assumed to be present at once such that it can be normalized, because states are made available incrementally. We estimate the parametrization for any observation to allow the system to operate with only incrementally available pieces of the observed trajectories.

To estimate the temporal parameter for any observed state, an inverse mapping $t = g(y)$ is also learned. This adds a bijective restriction on an activity, meaning that an activity cannot intersect itself. Concretely, it cannot map to the same state at different points in time. This is a limitation arising from the non-causality of Gaussian Processes used to model the activities. The limitation can be circumvented by segmenting observed state trajectories into non-intersecting intervals and model each separately.

3.1. Gaussian Processes

Gaussian Processes have been shown to be useful for prediction, modeling and detecting spatio-temporal trajectories such as motor vehicles in crossings [8], marine vessel paths [12] and human body dynamics [16]. GPs have also been used to model spatial regions and structures such as occupancy maps [9] and smooth motion paths [7]. They are also good for handling noisy or missing data [8,16], where large chunks of trajectories can reliably be reconstructed.

One major obstacle restricting their use has in the past been the expensive matrix inversion operation necessary to update the model. Various methods have been employed to reduce this weakness, by efficient iterative updates [13], segmentation into local patches of GPs [11,15], and by techniques to make the GP sparse [14,15]. In this work we employ sparse GPs to model activities, which are systematically segmented into local models.

The non-parametric model of the Gaussian Process is associated with a vector of inputs $\mathbf{x}$ and two corresponding vectors of outputs $\mathbf{y}$ and output uncertainties $\sigma_{\mathbf{y}}$. The posterior density given an observed point x^* is a Gaussian distribution. We use a sparse representation of support points $\mathbf{x}$ and $\mathbf{y}$, kept to a limited number, to make the calculations possible with any number of accumulated observations. The intention is to make it possible for the system to continue to learn from as many observations as possible to improve the confidence and get as much evidential support as possible.

3.2. Modeling Spatio-Temporal Dynamics

The activity model consist of the parametrized state trajectory $y = f_{GP}(t)$ and its inverse mapping $t = g_{GP}(y) = f_{GP}^{-1}(y)$. The inverse mapping is constructed by switching the input and the output, $\mathbf{x}$ and $\mathbf{y}$, of f_{GP}. Since the inverse mapping is with respect to the mean function of f_{GP}, it is exact at these points and the observation error is therefore zero. It is however chosen to be very small as to minimize numerical problems.

The main purpose of the inverse mapping is to project a state space point $x^* \sim \mathcal{N}(\mu_{x^*}, \sigma_{x^*}^2)$ onto the mean function of f_{GP}, thereby becoming the state space point $y^* \sim \mathcal{N}(\mu_{y^*}, \sigma_{y^*}^2)$. This is used when calculating the likelihood of a point on one trajectory being explained by a point on another. The calculation is performed by approximating x^* as μ_{x^*} and t^* as μ_{t^*},

$$y^* = f_{GP}(\mu_{t^*}), \quad t^* = g_{GP}(\mu_{x^*}). \tag{1}$$

The latter approximation is valid if the inverse mapping accurately models the inverse mean function. The former approximation can be improved by techniques for uncertain inputs to GPs described in [4]. The mappings $y = f_{GP}(t)$ and $t = g_{GP}(y)$ currently use the squared exponential kernel, which is commonly used in Gaussian Process regression,

$$k(x_1,x_2) = \theta_0^2 e^{-\frac{||x_1-x_2||_2^2}{2\theta_1^2}}, \tag{2}$$

where θ_0^2 denotes the global variance of the mapping and θ_1^2 denotes the global smoothness parameter of the mapping. The two mappings have separate parameters. The kernel hyper parameters θ are optimized for each activity by maximizing the marginal likelihood.

3.3. Local Likelihood

Let A be an activity with $\{f_{GP}, g_{GP}\}$ and let $x^* \sim \mathcal{N}(\mu_{x^*}, \sigma_{x^*}^2)$ be an observed point or a point on another activity. If the different dimensions of the state space are assumed to be independent, the local likelihood[8] of x^* given A is

$$P(x^*|A) = P(x^*|y^*) = \prod_{n=0}^{N} P(x_n^*|y_n^*), \qquad y^* = f_{GP}(g_{GP}(x^*)), \tag{3}$$

where N is the dimension of the state space and y^* is a Gaussian distribution. When considering multiple activities the calculation of the local likelihood is weighted by the relative evidence E in support of respective activity. If T_n is point n on trajectory T then

$$P^*(T_n|A_k) = \frac{E_k}{\sum_i(E_i)} P(T_n|A_k), \tag{4}$$

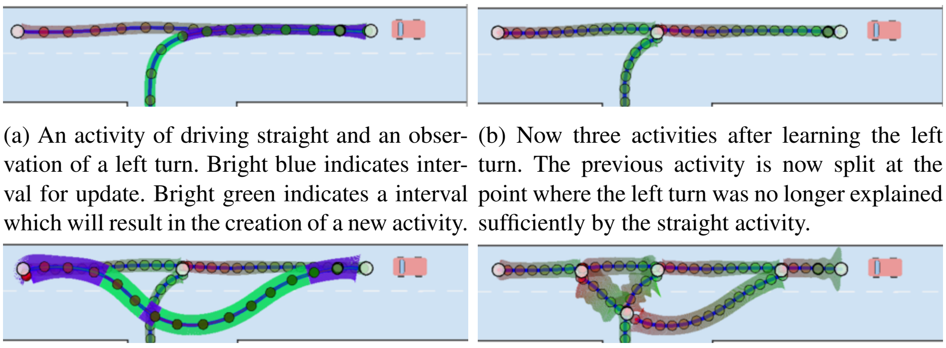
where i ranges over all activities in consideration of which A_k is one. This allows a well supported activity with high variance to keep significant importance.

4. Learning Activities

To learn activities from observed trajectories, the framework segments trajectories into sequences of activities, where each activity differs from all other activities currently known. Activities are segmented such that no two activities can have overlapping intervals in the state space, where the margin for overlap is controlled by $s_{threshold}$.

The learning process updates the State Space Graph and the Activity Transition Graph based on an observed trajectory. The first step is to generate a GP from the observed trajectory, and then to generate a new trajectory by uniformly sample the GP to get a smaller number of support points for computational efficiency. The next step is to classify the trajectory given the current set of activity models. Based on this classification the graphs are updated. If a long enough interval of an observed trajectory lacks sufficient explanation given the known activities, then this interval is considered novel and is added as a new activity. Intervals that are explained sufficiently by a single activity more than by any other is used to update that activity with the new observed information. Sometimes long enough parts of activities become too similar, i.e. too close together in the state space, to be considered different activities. In those cases such parts of these activities are merged. This removes cases where intervals of the observation is sufficiently and on equal terms explained by more than one activity.

By letting the transition nodes store statistics of the transitions between the activities, the Activity Transition Graph becomes another perspective of the same information contained in the State Space Graph. It is updated simultaneously with the State Space Graph in the different learning steps.



(a) An activity of driving straight and an observation of a left turn. Bright blue indicates interval for update. Bright green indicates a interval which will result in the creation of a new activity.

(b) Now three activities after learning the left turn. The previous activity is now split at the point where the left turn was no longer explained sufficiently by the straight activity.

(c) An illegal overtake is observed. The rounded trajectory of the overtake is indicated to be created as two activities since it is too similar to the left turn when spatially crossing it. This do not happen if velocities are part of the state space.

(d) After the overtake has been learned. The highly varying variance seen on the shorter activities is due to the parameters of the kernels not being updated properly.

Figure 4. Example of the learning of the State Space Graph using only 2D positions as state variables for the activities, in order (a)-(d). Activity coloring is from green (t = 0.0) to red (t = 1.0). The white circles are intersection points, the other circles are support points of the activities' f_{GP}.

4.1. Learning New Activities

A new activity is created for each maximal interval $\tau := [t_0 : t_1]$ where $P(T_\tau|A_k) < s_{threshold}$ for all activities A_k in consideration and where $\Delta l_\tau > l_{threshold}$. That is, any long enough interval of the observed trajectory, which is insufficiently explained by every concerned activity, is added to the framework as a new activity.

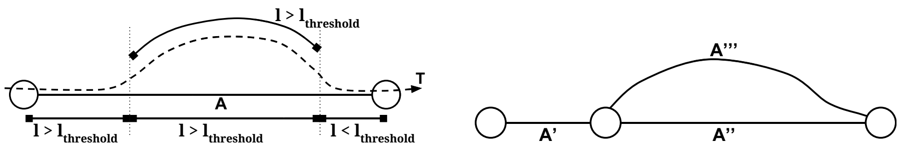


Figure 5. Illustrative example of the *create procedure*, where the left figure is before and the right figure is after. The dotted lines indicate where the similarity of A and T is crossing the $s_{threshold}$.

Intersection points are used to connect activities with other activities. Since activities are not allowed to be shorter than $l_{threshold}$, it is also the case that two intersection points cannot be created on the same activity with a distance between them shorter than $l_{threshold}$.

A new intersection point is created at each end if the length to the nearest other intersection point is larger or equal to $l_{threshold}$. If not, the new activity is connected to the nearest intersection point and the intersection point is updated to be placed at the weighted mean position of the end-points of the connected activities. The weighting is done in accordance to the amount of evidence of each connected activity.

If a new activity is branching into or out of a previous activity, the previous activity is split in two at the branching point. The intersection point is placed at the weighted mean position.

If an intersection point after its placement update is closer than $l_{threshold}$ to another intersection point, they are merged into a single intersection point. An illustrative example is shown in Figure 5.

4.2. Merging Activities

To remove redundant activities, activities are merged at intervals that are too similar to each other. They are also merged in order to allow for spatially wide activities, which might be initially observed as several parallel activities growing together over time as more observations are acquired.

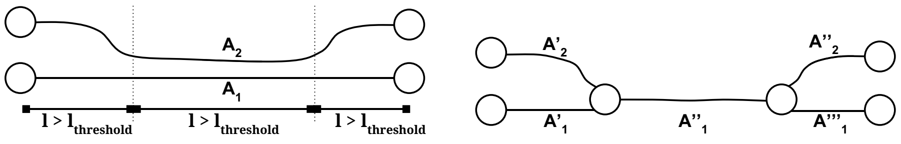


Figure 6. An illustrative example of the *merge procedure*, where the left figure is before and the right figure is after. The dotted lines indicate where the similarity of both A_2 and A_1 are exceeding the $s_{threshold}$

All activities A_k are merged with A^* on the interval $\tau := [t_0 : t_1]$ for which $P(T_\tau|A_k) \geq s_{threshold}$, $\Delta l_\tau > l_{threshold}$ and A^* is the activity with the maximum similarity $P(T_\tau|A^*)$. That is, all activities which sufficiently explain long enough intervals of the observed trajectory are merged with the activity best explaining that interval.

Activities are currently merged and updated by updating the most supported activity's set of GP support points within the given interval with the corresponding state space points sampled from the other activities. The new state space location of the support point is the weighted mean of its previous location and the sampled location. The weighting is based on the number of previous observations in support of respectively activity, used as a measure of evidence for respective activity model. The final activity has the combined amount of evidence from all activities merged. Intersection points are created and updated in the same way as before. An illustrative example is shown in Figure 6.

4.3. Updating Activities

Any interval τ_i on T not used for creating or merging in the two previous steps carry information that are used to update activities. The activity best explaining such an interval cannot have been created or been part of a merge in the previous two steps.

The intervals remaining with $length > l_{threshold}$, $I_n, n = 1...N$, always have at every point an activity A^* with maximum similarity where $P(T_t|A^*) \geq s_{threshold}$ holds except for intervals with $length \leq l_{threshold}$. These intervals are segmented into shorter intervals such that there is only a single activity with maximum similarity for the entire interval, $I_{n,i}$.

If $length(I_{n,i}) \leq l_{threshold}$ it is combined with the neighbor of shortest length until they are composite intervals satisfying $length(Ic_{n,i}) > l_{threshold}$. The activity assigned as the most likely to the composite interval is the activity A_k which has the highest $P(Ic_{n,i}|A_k)$ for all k in consideration.

The intervals and composite intervals are merged with the most similar activity as described in the previous section, with the weight of the intervals equal to one since they only represent a single observation.

5. Detecting and Predicting Activities

The State Space Graph and the Activity Transition Graph can be used in several ways to detect and predict activities. There is statistical information such as the evidence for each activity, similarity between any two activities and the probability of transition between activities. Assuming the presence of one or several learned activities, it is straightforward to calculate the activity chain from n steps in the past to m steps into the future that is most likely, least likely or anything in between. It is also possible to calculate the k most likely activity chains, with or without sharing activities at any step.

Another possibility is to generate relevant events as part of the learning procedure. Events can for example be created when the learning procedure creates, updates and merges activities, or when an object has completed an activity due to the object crossing an intersection point.

6. Preliminary Results

To verify that our approach works as expected, we have done a case study with a T-crossing scenario. The data set contains two types of trajectories, straight trajectories and left turning trajectories, 50 of each. These trajectories are provided to the framework in a random order. The converged trajectories can be seen in Figure 7 compared to the generated data. With position and constant velocity the turning activity breaks up a bit earlier than with only position, and even more so with the slow down taking place.

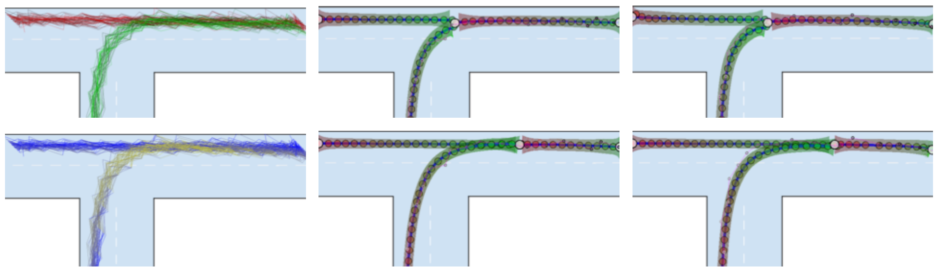


Figure 7. Case study. Top row: Scenario with positions only. Bottom row: Scenario with a slow down. The figures in the middle show the three learned activities after two observed trajectories, while the figures to the right show the result after all observations. Blue color indicate a higher speed than yellow.

7. Conclusions and Future Work

Learning and recognizing spatio-temporal activities from streams of observations is a central problem in artificial intelligence. In this paper, we have proposed an initial approach towards an unsupervised stream processing framework that learns a Bayesian representation of observed spatio-temporal activities and their causal relations from observed trajectories of objects. An activity is represented by a Gaussian Process and the set of activities is represented by a State-Space Graph where the edges are activities and the nodes are intersection points between activities. To reason about chains of related activities an Activity Transition Graph is also learned which represents the statistical information about transitions between activities. To show that the framework works as intended, it has been applied to a small traffic monitoring example.

The framework cannot yet learn complex activities as discrete entities other than indirectly by parts, i.e. by learning chains of primitive activities. Neither are contexts such as *a T-crossing* learned, instead configurations of different activities are learned for each T-crossing instance. The current framework is designed to provide the building blocks for these extensions to be possible in future work.

References

[1] J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43(3), 2011.

[2] C. Dousson and P. Le Maigat. Chronicle recognition improvement using temporal focusing and hierarchization. In *Proc. IJCAI*, 2007.

[3] R. Gerber and H.-H. Nagel. Representation of occurrences for road vehicle traffic. *Artificial Intelligence*, 172(4–5):351–391, 2008.

[4] A. Girard, C. Rasmussen, J. Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Proc. NIPS*, 2002.

[5] N. Guillarme and X. Lerouvreur. Unsupervised extraction of knowledge from S-AIS data for maritime situational awareness. In *Proc. FUSION*, 2013.

[6] F. Heintz, P. Rudol, and P. Doherty. From images to traffic behavior - a uav tracking and monitoring application. In *Proc. FUSION*, 2007.

[7] C. Tay Meng Keat and C. Laugier. Modelling smooth paths using gaussian processes. In *Proc. FSR*, 2007.

[8] K. Kim, D. Lee, and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In *Proc. ICCV*, 2011.

[9] S. Kim and J. Kim. Continuous occupancy maps using overlapping local gaussian processes. In *Proc. IROS*, 2013.

[10] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3):397–408, 2005.

[11] M. Schneider and W. Ertel. Robot learning by demonstration with local gaussian process regression. In *Proc. IROS*, 2010.

[12] M. Smith, S. Reece, I. Rezek, I. Psorakis, and S. Roberts. Maritime abnormality detection using gaussian processes. *Knowledge and Information Systems*, pages 1–26, 2013.

[13] M. Smith, S. Reece, S. Roberts, and I. Rezek. Online maritime abnormality detection using gaussian processes and extreme value theory. In *Proc. ICDM*, 2012.

[14] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Proc. NIPS*, 2005.

[15] E. Snelson and Z. Ghahramani. Local and global sparse gaussian process approximations. In *Proc. AISTATS*, 2007.

[16] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):283–298, 2008.

[17] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception by hierarchical bayesian models. In *Proc. CVPR*, 2007.