Context Aware Observation of Human Activities

James L. Crowley
Professeur I.N.P. Grenoble
INRIA Rhône Alpes, Montbonnot, France

Abstract

Interactive environments combine perception, action and communication to extend human-computer interaction. We believe that a fundamental challenge for interactive environments is developing models and methods for "context awareness". In this paper we present an ontology for context awareness for interactive environments. We show how the elements of this ontology correspond to the elements of a software architecture for observing situation and context. Within this framework, context predicts the evolution of situation, and provides "meaning" for objects and events. Context also provides a specification for assembling federations of processes to measure properties, determine relations and detect events.

1. INTRODUCTION

In this paper, we propose an ontology and a software architecture for modeling context and situation. A key aspect of our approach is that we recognize that a context aware system must be able to sense users and their activities. Unlike much of the previous work on context aware systems, we are especially concerned with the perceptual components for context awareness. We propose a data-flow architecture based on dynamically assembled federations [1], [2]. Our model builds on previous work on process-based architectures for machine perception and computer vision [3], [4], as well as on data flow models for software architecture [5].

We propose a model in which a users context is described by a set of roles and relations. A context is translated into a federation of observational processes. Different configurations of roles and relations correspond to situations within the context. This model leads to an architecture in which reflexive elements are dynamically composed to form federations of processes for observing and predicting the situations that make up a context. As context changes, the federation is restructured. Within a context, the federation can adapt so as to provide services that are appropriate and invariant over a range of situations.

2. CONTEXT AWARE OBSERVATION

In order to provide an operational theory of context awareness, we develop an ontology for context and situation. As we develop each term of the ontology, we give the term computational meaning by describing the corresponding architectural components. As in other domains, an ontology for context awareness requires both top-down and bottom up components. Bottom up components are tied to whatever the system can sense and interpret. The top down elements are derived from users and their tasks.

2.1 The user's context

The context of which the system should be aware is that of one or more humans. Let us refer to these human agents using the common computer science term of user. We assume that in most cases users are driven by one of more goals, although often not in the purely rational single-minded manner that is assumed by most AI planning systems. The user may have many possible goals, sometimes in parallel, and he may switch among these goals in a dynamic manner that may be difficult to predict. In most cases, interacting directly with the system is NOT the goal of the user. Thus, as the system designer, we must endeavor to make the system disappear into the environment in order to assist users without drawing their attention away from their current tasks. To design such systems we need to have a clear notion of goal, task and activity.

A rational system chooses its actions to accomplish its goals [6]. The fundamental concept for a formal definition of task is that of state [7]. A state is defined using a predicate expression. The logical functions that make up this expression are functions of properties observed in the world. Each possible combination of predicates (or their negation) defines a state. A universe is a graph in which states are connected by arcs that represent actions. At any instant in time the universe is in a state called the current state. The user may desire to bring the universe to another state called a goal state. To attain a goal state, the user must perform some sequence of actions. To determine possible sequences of actions he must search the graph of states for a path to the desired state [8]. The association of a current state and a goal state is a task. Unlike some work in HCI, we insist that a task does not explicitly determine the sequence of user's actions. The set of action sequences that a user may choose is an open set that may be determined "on the fly".

Real humans are rarely obsessed with a single task. In most situations, humans react opportunistically, switching among a set of possible goals, abandoning and adding new goals in response to events and opportunities. One of the most difficult challenges in designing context aware systems is to recognize and allow for such unpredictable behavior. We call a composition of states and actions for the user a domain. The current set of tasks is the user's activity. We assume that at any instant, the user is pursuing a task from this set. The other tasks may be referred to as background tasks. Together, the current task, and the background tasks define the set of things that the user may attend to, and the set of actions that he may undertake.

2.2 The system's context

The system's context is composed of a model of the user's context plus a model of its own internal context. The system's model of the user's context provides the means to determine what to observe and how to interpret the observations. The system's model of its own context provides a means to compose the federation of components that observe the user's context.

At the lowest level, the system's view of the world is provided by a collection of sensors. These sensors generate values for observable variables. Observable variables may be numeric or symbolic entities. They may be produced as a synchronous stream of data or as asynchronous events. In order to determine meaning from observable variables the system must perform some series of transformations. The fundamental component for our software architecture is an observational process, as shown in figure 1.

An observational process has two functional facets: A transformation component and a supervisory controller. The supervisory controller enables reflexive control of observational processes and thus provides a number of important functions. The control component receives commands and parameters, supervises the execution of the transformation component, and responds to queries with a description of the current state and capabilities. The characteristics of the control component are developed below.

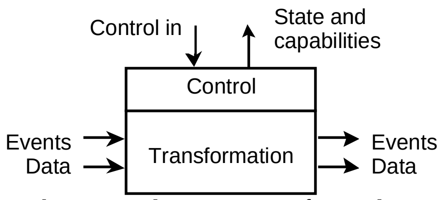

Fig. 1. An observational process transforms data and events into data and events.

The input data to the transformational component is generally composed of some raw numerical values, generally arriving in a synchronous stream, accompanied by meta-data. Meta data includes information such as a time-stamp, a confidence factor, a priority or a description of precision. An input event is a symbolic message that can arrive asynchronously and that may be used as a signal to begin or terminate the transformation of the input data. Output data and the associated meta-data is a synchronous stream produced from the transformation of the input data. We also allow the possibility of generating asynchronous output messages that may serve as events for other processes. This model is similar to that of a *contextor* [9], which is a conceptual extension of the context widget implemented in the Context Toolkit [10].

2.3 Examples

A very simple example of a observational process is provided by a transformation that uses table look-up to convert a color pixel represented as an RGB vector into a probability of skin. Such a table can easily be defined using the ratio of a histograms of skin colored pixels in a training image, divided by the histogram of all pixels in the same image [11]. A fundamental aspect of interpreting sensory observations is grouping observations to form entities. While entities may generally be understood as corresponding to physical objects, from the perspective of the system, an entity is an association of correlated observable variables. This association is commonly provided by an observational process that groups variables based on spatial co-location. Correlation may be based on temporal location or other, more abstract relations. Thus, an entity is a predicate function of one or more observable variables. Entities may be composed by a entity grouping processes. The input data is typically a set of streams of numerical or symbolic data. The output of the transformation is a stream including a symbolic token to identify the kind of the entity, accompanied by a set of numerical or symbolic properties. These properties allow the system to define relations between entities. The detection or disappearance of an entity may, in some cases, also generate asynchronous symbolic signals that are used as events by other processes.

A simple example of an entity detection process is provided by a process that groups adjacent skin colored pixels into regions (commonly called blobs). The zeroth moment is the sum of the probabilities in the ROI. Let us suppose that the ROI is composed of R rows and C columns to provide N pixels. The ratio of the sum of probability pixels M over the number or pixels in the ROI, N provides a measure of the confidence that a skin colored region has been observed. The first moment of w(i, j) is the center of gravity in the row and column directions (x, y). This is a robust indicator of the position of the skin colored blob. The second moment of w(i, j) is a covariance matrix. The square root of the principle components are the length and breadth of the region. The principal vector indicates the dominant direction of the region. Principal components analysis of the covariance matrix formed from σ_{ii}^2, σ_{jj}^2,and σ_{jj}^2 yield the length and breadth of the blob (s_x, s_y) as well as its orientation θ.

A fundamental aspect of interpreting sensory observations is determining relations between entities. Relations can be formally defined as a predicate function of the properties of entities. Relations that are important for describing context include 2D and 3D spatial relations, as well as temporal relations [12]. Other sorts of relations, such as acoustic relations (e.g. louder, sharper), photometric relations (e.g. brighter, greener), or even abstract geometric relations may also be defined. As with observable variables and with entities, we propose to observe relations between entities using observational processes. Observational processes transform entities into relations based on their properties. As before, this transformation may be triggered by and may generate asynchronous symbolic messages that can serve as asynchronous events.

An example of relation detector is provided by a process that associates the output from two eye detectors and a skin blob detector to detect the left and right eyes of a face. Eyes may be detected using a process based on receptive field vectors [13] that goes beyond the scope of this paper. Each eye-entity is labeled with a position and size. The eye pair detector uses the relative positions and sizes to determine if two possible eye entities can be eyes, and to determine which entity is the left eye, and which is the right eye.

Tracking processes provide a number of important properties for observing context. A tracking system conserves information about entities over time. Thus, for example, it is only necessary to recognize an entity once. Tracking also makes it possible to compose a history of the positions of an entity. Changes in position can be important indicators of changes in the user's situation or context. Finally, tracking is very useful for optimizing processing by focusing attention. The ROI's used in skin color detection, and skin blob detection may be provided by the position of the blob from a previous observation by a tracking process.
Tracking is a process of recursive estimation. A well-known framework for such estimation is the Kalman filter. A complete description of the Kalman filter [14] is beyond this paper. A general discussion of the use of the Kalman filter for sensor fusion is given in [15]. The use of the Kalman filter for tracking faces is described in [16]. For face tracking we commonly use a simple zeroth order Kalman filter, in which

the observation and estimation state vectors are each composed of (x, y, s_x, s_y, θ).

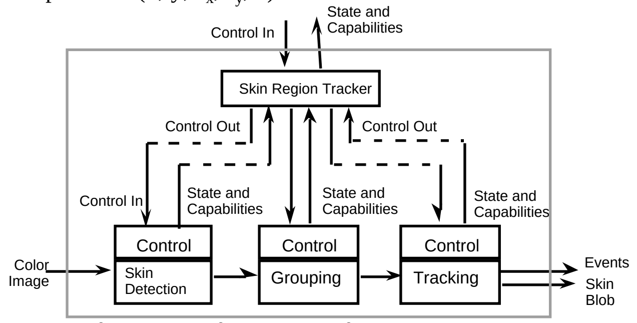

Fig 2 A federation of processes for observing skin colored blobs. A second level supervisory controller invokes the first level observational processes, and supervises their execution.

2.4 A supervisory controller for observational processes

A federation of observational processes may be composed using a hierarchy of reflexive supervisor controllers. Each supervisory controller invokes and controls lower level controllers that perform the required transformation. At the lowest level are observational processes that observe variables, group observational variables into entities, track entities and observe the relations between entities.

The skin blob tracker provides an example of such a controller. The supervisory controller, labeled as "skin region tracker" in figure 2 invokes and coordinates observational processes for skin detection, pixel moment grouping and tracking. This federation provides the transformation component for a composite observation process. The skin region tracker provides the supervisory control for this federation.

3. CONTEXT AND SITUATION.

From the user's perspective we have definitions for task and activity. From the system's perspective, we have definitions for observable variables, entities and relations. These definitions meet to provide a model of situation and context.

3.1 Formal Definition of Context and Situation

The context for a user U and task T is a composition of situations. These situations all share the same set of roles and relations. Thus a context determines the collection of roles and relations to observe. These are the roles and relations that are relevant to the task.

A role is a function relative to a task. A role may be satisfied by one or more entities in the user's environment. An entity is judged to be capable of providing a role if it passes an acceptance test on its properties. For example, a horizontal surface may serve as a seat if it is sufficiently large and solid to support the user, and is located at a suitable height above the floor. An object may serve as a pointer if it is of a graspable size and appropriately elongated. In the user's environment, pens, remote controls, and even a wooden stick may all meet this test and be potentially used by the user to serve the role of a pointer.

The set of entities that can provide a role may be open ended. A user determines if an entity can satisfy a role for a task by applying the acceptance test. This test is a predicate function defined over entities and their properties. When the test is applied to multiple entities, the most suitable entity may be selected based on a confidence factor, CF.

The set of entities is not bijective with the set of roles. One or more entities may play a role. A role may be played by one or several entities. What's more the assignment of entities to roles may (often will) change dynamically. Such changes provide the basis for an important class of events.

The user's situation is a particular assignment of entities to roles completed by a set of relations between the entities. Situation may be seen as the "state" of the user with respect to his task. The predicates that make up this state space are the roles and relations determined by the context. If the relations between entities changes, or if the binding of entities to roles changes, then the situation within the context has changed. The context and the state space remains the same.

Thus a context can be seen as a network of situations defined in a common state space. A change in the relation between entities, or a change in the assignment of entities to roles is represented as a change in situation. Such changes in situation constitute an important class of events that we call Situation-Events. Situation-Events are data driven. The system is able to interpret and respond to them using the context model. They do not require a change in the federation of observational processes.

4. PROCESS FEDERATIONS.

In this section we describe how to construct a hierarchical federations of observational processes. We develop a set of software properties that permit processes to be dynamically composed into federations to robustly observe and predict user actions. We then describe a meta-process that dynamically composes process federations based on the system context and the user context.

The system context provides a method to compose a federation of observation processes for observing the roles and relations relevant to the user's context. In order to compose these processes, we define a reflexive supervisory controller that recruits lower observational processes to form local federations. The roles and relations specified by a system context are used by supervisory controllers to construct a "federation" of observational processes. This federation determines and tracks the entities that may play roles in the users context, determines the assignment of roles to entities to roles, and determines the relations between theses entities.

Just as the user may select entities to perform a role, so the system may also select observational processes to satisfy observational roles. The systems task is to observe the roles and relations of the user's context. This defines a system context in which observational processes perform functions, and thus may be said to assume roles. A supervisory controller observes the state and capabilities of observational processes to determine if they are most appropriate at the current time to provide the required function.

Similarly the system's situation is the current federation of processes that have been assembled to observe the user's context. Observational processes serve roles in the systems context. If the observational processes for serving a system role changes, the systems situation changes, but the system context remains the same. Whenever the set of relevant roles or relations changes, the system must reorganize the federation in order to accommodate the required observations. Thus a change in context is a separate class of event, a Context-Event. Recognizing context events

constitutes a special challenge in designing a context aware system.
In order to dynamically assemble and control observational processes, the system must have information about the capabilities and the current state of component processes. Such information can be provided by assuring that supervisory controllers have the reflexive capabilities of auto-regulation, auto-description and auto-criticism.

A process is auto-regulated when processing is monitored and controlled so as to maintain a certain state. For example, processing time and precision are two important state variables for a tracking process. These two may be traded off against each other. The choice of priority is dictated by a more abstract supervisory controller.

A second level supervisory controller may be coordinating several skin-region trackers. The time available for each tracker will depend, in part on the number of regions to be tracked. Thus the second level controller must dynamically inform each observation process of the required T_{max}. Furthermore, the relative priorities of time and precision may vary according to the role that has been assigned to each blob. Thus a hierarchy of more abstract controllers may be involved in providing the reference commands for an observational process. Such coordination of such a hierarchy requires that the processes be capable of describing both their current state and their capabilities.

An auto-descriptive controller can provide a symbolic description of its capabilities and state. The description of the capabilities includes both the basic command set of the controller and a set of services that the controller may provide to a more abstract controller. Thus when applied to the systems context, our model provides a means for the dynamic composition of federations of controllers. In this view, the observational processes may be seen as entities in the system context. The current state of a process provides its observational variable. Supervisory controllers are formed into hierarchical federations according to the system context. A controller may be informed of the possible roles that it may play using a meta-language, such as XML.

An auto-critical process maintains an estimate of the confidence for its outputs. For example, the skin-blob detection process maintains a confidence factor based on the ratio of the sum of probabilities to the number of pixels in the ROI. Such a confidence factor is an important feature for the control of processing. Associating a confidence factor to all observations allows a higher-level controller to detect and adapt to changing observational circumstances. When supervisor controllers are programmed to offer “services” to higher-level controllers, it can be very useful to include an estimate of the confidence for the role. A higher-level controller can compare these responses from several processes and determine the assignment of roles to processes.

5. CONCLUSIONS

A context is a network of situations concerning a set of roles and relations. Roles are services or functions relative to a task. Roles may be “satisfied” with one or more “entities”. A relation is a predicate defined over the properties of entities. A situation is a particular assignment of entities to roles completed by the values of the relations between the entities. Entities and relations are predicates defined over observable variables.

This ontology provides the basis for a software architecture for the observational components of context aware systems. Observable variables are provided by reflexive observational processes whose functional core is a transformation. Observational processes are invoked and organized into hierarchical federations by reflexive supervisory controllers. A model of the user’s context makes it possible for a system to provide services with little or no intervention from the user. Applying the same ontology to the system’s context provides a method to dynamically compose federations of observational processes to observe the user and his context.

6. REFERENCES

[1] Software Process Modeling and Technology, edited by A. Finkelstein, J. Kramer and B. Nuseibeh, Research Studies Press, John Wiley and Sons Inc, 1994.

[2] J. Estublier, P.Y.Cunin, N. Belkhatir, "Architectures for Process Support Ineroperability", ICSP5,Chicago, 15-17 juin, 1997.

[3] J. L. Crowley, "Integration and Control of Reactive Visual Processes", Robotics and Autonomous Systems, Vol 15, No. 1, décembre 1995.

[4] J. Rasure et S. Kubica, “The Khoros application development environment “, in Experimental Environments for computer vision and image processing, H. Christensen et J. L. Crowley, Eds, World Scientific Press, pp 1-32, 1994.

[5] M. Shaw and D. Garlan, Software Architecture: Perspectives on an Emerging Disciplines, Prentice Hall, 1996.

[6] Newell, A. "The Knowledge Level", Artificial Intelligence 28(2), 1982.

[7] Nilsson, N. J. Principles of Artificial Intelligence, Tioga Press, 1980.

[8] R. Korf, "Planning as Search", Artificial Intelligence, Vol 83, Sept. 1987.

[9] J. Coutaz and G. Rey, “Foundations for a Theory of Contextors”, in Computer Aided Design of User Interfaces, Springer Verlag , June 2002.

[10] D. Salber, A.K. Dey, G. Abowd. The Context Toolkit: Aiding the development of context-enabled Applications. *In Proc. CHI99*, ACM Publ., 1999, pp. 434-441.

[11] K. Schwerdt and J. L. Crowley, "Robust Face Tracking using Color", 4th IEEE International Conference on Automatic Face and Gesture Recognition", Grenoble, France, March 2000.

[12] J. Allen, "Maintaining Knowledge about Temporal Intervals", Journal of the ACM, 26 (11) 1983.

[13] D. Hall, V. Colin de Verdiere and J. L. Crowley, "Object Recognition using Coloured Receptive Field", 6th European Conference on Computer Vision, Springer Verlag, Dublin, June 2000.

[14] R. Kalman, "A new approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Series D. J. Basic Eng., Vol 82, 1960.

[15] J. L. Crowley and Y. Demazeau, “Principles and Techniques for Sensor Data Fusion“, Signal Processing, Vol 32 Nos 1-2, p5-27, May 1993.

[16] J. L. Crowley and F. Berard, "Multi-Modal Tracking of Faces for Video Communications", IEEE Conference on Computer Vision and Pattern Recognition, CVPR '97, St. Juan, Puerto Rico, June 1997.