Context Driven Observation of Human Activity

James L. Crowley

Laboratoire GRAVIR, INRIA Rhône Alpes,
655 Ave de l'Europe, F-38330 Montbonnot, France
Crowley@inrialpes.fr
http://www-prima.imag.fr

Abstract. Human activity is extremely complex. Current technology allows us to handcraft real-time perception systems for a specific perceptual task. However, such an approach is inadequate for building systems that accommodate the variety that is typical of human environments. In this paper we define a framework for context aware observation of human activity. A context in this framework is defined as a network of situations. A situation network is interpreted as a specification for a federation of processes to observe humans and their actions. We present a process-based software architecture for building systems for observing activity. We discuss methods for building systems using this framework. The framework and methods are illustrated with examples from observation of human activity in an "Augmented Meeting Environment".

1 Introduction

This paper presents a framework for context aware observation of human activity. Within this framework, contexts are modeled as a network of situations. Situation models are interpreted to dynamically configure a federation of processes for observing the entities and relations that define a situation. We propose a software architecture based on dynamically assembled process 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].

Within this framework, a situation is described by a configuration of relations for observed entities. Changes in relations correspond to events that signal a change in situation. Such events can be used to trigger actions by the system. Situation models are used to specify an architecture in which reflexive processes are dynamically composed to form federations for observing and predicting situations. We believe that this architecture provides a foundation for the design of systems that act as a silent partner to assist humans in their activities in order to provide appropriate services without explicit commands and configuration. In the following section we review the use of the term "context aware" in different domains. This leads us to a situation-based approach to modeling context.

2. A Brief History of Context

The word "context" has come to have many uses. Winograd [6] points out that the word “Context” has been adapted from linguistics. Composed of “con” (with) and “text”, context refers to the meaning that must be inferred from the adjacent text. Such meaning ranges from the references intended for indefinite articles such as “it” and “that” to the shared reference frame of ideas and objects that are suggested by a text. Context goes beyond immediate binding of articles to the establishment of a framework for communication based on shared experience. Such a shared framework provides a collection of roles and relations with which to organize meaning for a phrase.

Early researchers in both artificial intelligence and computer vision recognized the importance of a symbolic structure for understanding. The “Scripts” representation [7] sought to provide just such information for understanding stories. Minsky’s Frames [8] sought to provide the default information for transforming an image of a scene into a linguistic description. Semantic Networks [9] sought to provide a similar foundation for natural language understanding. All of these were examples of what might be called “schema” [10]. Schema provided context for understanding, whether from images, sound, speech, or written text. Recognizing such context was referred to as the “Frame Problem” and became known as one of the hard unsolved problems in AI.

In computer vision, the tradition of using context to provide a framework for meaning paralleled and drew from theories in artificial intelligence. The “Visions System” [11] expressed and synthesized the ideas that were common among leading researchers in computer vision in the early 70’s. A central component of the “Visions System” was the notion of a hierarchical pyramid structure for providing context. Such pyramids successively transformed highly abstract symbols for global context into successively finer and more local context terminating in local image neighborhood descriptions that labeled uniform regions. Reasoning in this system worked by integrating top-down hypotheses with bottom-up recognition. Building a general computing structure for such a system became a grand challenge for computer vision. Successive generations of such systems, such as the “Schema System”[12] and “Condor” [13] floundered on problems of unreliable image description and computational complexity. Interest in the 1990’s turned to achieving real time systems using “active vision” [14], [15]. Many of these ideas were developed and integrated into a context driven interpretation within a process architecture using the approach “Vision as Process” [16]. The methods for sensing and perceiving context for interaction described below draws from this approach.

Context awareness has become very important to mobile computing where the term was first introduced by Schilit and Theimer [17]. In their definition, context is defined as “the location and identities of nearby people and objects and changes to those objects”. While this definition is useful for mobile computing, it defines context by example, and thus is difficult to generalize and apply to other domains. Other authors, such as [18] [19] and [20] have defined context in terms of the environment or situation. Such definitions are essentially synonyms for context, and are also difficult to apply operationally. Cheverest [21] describes context in anecdotal form using scenarios from a context aware tourist guide. His system is considered one of the early models for a context aware application.

Pascoe [22] defines context to be a subset of physical and conceptual states of interest to a particular entity. This definition has sufficient generality to apply to a recognition system. Dey [23] reviews definitions of context, and provides a definition of context as "any information that can be used to characterize situation". This is the sense in which we use the term context. Situation refers to the current state of the environment. Context specifies the elements that must be observed to model situation. However, to apply context in the composition of perceptual processes, we need to complete a clear definition with an operational theory. Such a foundation is provided by a process-based software architecture.

3. Perceptual Components for Context Awareness

In this section we describe a process-based software architecture for real time observation of human activity. The basic component of this architecture is a perceptual process. Perceptual processes are composed from a set of modules controlled by a supervisory controller. We describe several common classes of modules, and describe the operation of the controller. We also present several classes of perceptual processes and discuss how they can be combined into process federations according to a network of expected situations.

3.1 Perceptual processes

A system's view of the external world is driven by a collection of sensors. These sensors generate observations that may have the form of numeric or symbolic values. Observations may be produced in a synchronous stream or as asynchronous events. In order to determine meaning from observations, a system must transform observations into some form of action. Such transformations may be provided by perceptual processes.

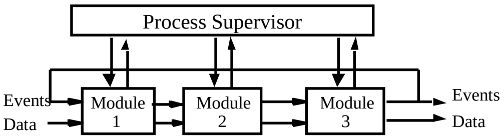

Fig. 1. A perceptual process integrates a set of modules to transform data streams or events into data streams or events.

Perceptual processes are composed from a collection of modules controlled by a process supervisor, as shown in figure 1. Processes operate in a synchronous manner within a shared address space. In our experimental system, the process supervisor is implemented as a multi-language interpreter [24] equipped with a dynamic loader for precompiled libraries. This interpreter allows a processes to receive and interpret messages containing scripts, to add new functions to a process during execution.

The modules that compose a process are formally defined as transformations applied to a certain class of data or event. Modules are executed in cyclic manner by the supervisor according to a process schedule. We impose that transformations return an auto-critical report that describes the results of their execution. Examples of information contained in an auto-critical report include elapsed execution time, confidence in the result, and any exceptions that were encountered. The auto-critical report enables a supervisory controller to adapt parameters for the next call in order to maintain a execution cycle time, or other quality of service.

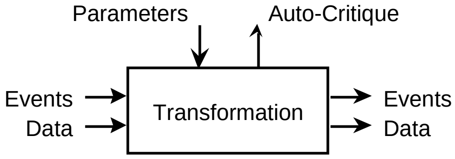

Fig. 2. Modules apply a transformation to an input data stream or events and return an auto-critical report.

3.2 Examples: modules for observing, grouping and tracking

A typical example of a module is a transformation that uses table look-up to convert a color pixel into a probability of skin, as illustrated in figure 3. 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 [25]. Skin pixels for an individual in a scene will all exhibit the same chrominance vector independent of surface orientation and thus can be used to detect the hands or face of that individual [26]. This technique has provided the basis for a very fast (video rate) process that converts an RGB color image into image of the probability of detection based on color using a look-up table.

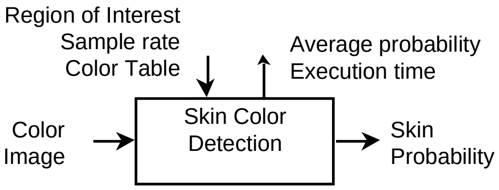

Fig 3. A module for detecting skin colored pixels with a region of interest

A color observation module applies a specified look-up table to a rectangular "Region of Interest" or ROI using a specified sample rate. The sample rate, S, can be adapted to trade computation time for precision. The output from the module is an image in which pixels inside the ROI have been marked with the probability of detection. The auto-critical report returns the average value of the probabilities (for use as a confidence factor) as well as the number of microseconds required for execution. The average probability can be used to determine whether a target was detected within the ROI. The execution time can be used by the process supervisor to assure that the overall execution time meets a constraint. This module can be used either for initial detection or for tracking, according to the configuration specified by the supervisor.

The color observation module is one example of a pixel level observation module. Pixel level observation modules provide the basis for an inexpensive and controllable perceptual processes. In our systems, we use pixel level observation modules based on color, motion, background subtraction [27], and receptive field histograms [28]. Each of these modules applies a specified transformation to a specified ROI at a specified sample rate and returns an average detection probability and an execution time.

Interpretation requires that detected regions be grouped into "blobs". Grouping is provided by a grouping module, defined using on moments, as shown in figure 4.

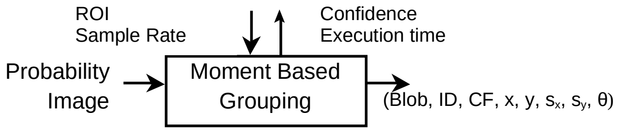

Fig 4. A module for grouping detected pixels using moments

Let w(i,j) represent an image of detection probabilities provided by a pixel level observation process. The detection mass, M, is the sum of the probabilities within the ROI. The ratio of the sum of probability pixels to the number of pixels, N, in the ROI provides a measure of the confidence that a skin colored region has been observed.

$$M = \sum_{i,j \;\; ROI} w(i,j) \qquad\qquad CF = \frac{M}{N}$$

The first moment of the detected probabilities 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.

$$x = \frac{1}{M} \sum_{i,j \;\; ROI} w(i,j)\ \ i \qquad\qquad y = \frac{1}{M} \sum_{i,j \;\; ROI} w(i,j)\ \ j$$

The second moment of w(i, j) is a covariance matrix. Principal components analysis of the covariance matrix formed from $\sigma_{ii}{}^2$, $\sigma_{jj}{}^2$,and $\sigma_{jj}{}^2$ yield the length and breadth of (s_x, s_y), as well as its orientation θ,of the blob of detected pixels.

$$\sigma_{ii}^2 = \frac{1}{M} \sum_{i,j \;\; ROI} w(i,j)\ \ (i-x)^2 \qquad\qquad \sigma_{jj}^2 = \frac{1}{M} \sum_{i,j \;\; ROI} w(i,j)\ \ (j-y)^2$$

$$\sigma_{ij}^2 = \frac{1}{M} \sum_{i,j \;\; ROI} w(i,j)\ \ (i-x)\ \ (j-y)$$

Tracking is a cyclic process of recursive estimation applied to a data stream. The Kalman filter provides a framework for designing tracking processes [29]. A general discussion of the use of the Kalman filter for sensor fusion is given in [30]. The use of the Kalman filter for tracking faces is described in [31].

Tracking provides a number of fundamentally important functions for a perception system. Tracking aids interpretation by integrating information over time. Tracking makes it possible to conserve information, assuring that a label applied to an entity at time T_1 remains associated with the entity at time T_2. Tracking provides a means to focus attention, by predicting the region or interest and the observation module that should be applied to a specific region of an image. Tracking processes can be designed to provide information about position speed and acceleration that can be useful in describing situations.

In perception systems, a tracking process is generally composed of three phases: predict, observe and estimate, as illustrated in figure 4. Tracking maintains a list of entities, known as "targets". Each target is described by a unique ID, a target type, a confidence (or probability of existence), a vector of properties and a matrix of uncertainties (or precisions) for the properties.

The prediction phase uses a temporal model (called a "process model" in the tracking literature) to predict the properties that should be observed at a specified time for each target. For many applications of tracking, a simply linear model is adequate for such prediction. A linear model maintains estimates of the temporal derivatives for each target property and uses these to predict the observed property values. For example, a first order temporal model estimates the value for a property, X_{T1}, at time T_1 from the value X_{T0} at a time T_0 plus the temporal rate of change multiplied by the time step, $\Delta T = T_1 - T_0$.

$$X_{T1} = X_{T0} + \Delta T(dX/dt)_{T0}$$

Higher order linear models may also be used provided that the observation sample rate is sufficiently fast compared to the derivatives to be estimated. Non-linear process models are also possible. For example, articulated models for human motion can provide important constraints on the temporal evolution of targets.

The prediction phase also updates the uncertainty (or precision model) of properties. Uncertainty is generally represented as a covariance matrix for errors between estimated and observed properties. These uncertainties are assumed to arise from imperfections in the process model as well as errors in the observation process.

Restricting processing to a region of interest (ROI) can greatly reduce the computational load for image analysis. The predicted position of a target determines the position of the ROI at which the target should be found. The predicted size of the target, combined with the uncertainties of the size and position, can be used to estimate the appropriate size for the ROI. In the tracking literature, this ROI is part of the "validation gate", and is used to determine the acceptable values for properties.

Observation is provided by the observation and grouping modules described above. Processing is specific for each target. A call to a module applies a specified observation procedure for a target at a specified ROI in order to verify the presence of the target and to update its properties. When the detection confidence is large, grouping the resulting pixels provides the information to update the target properties.

The estimation process combines (or fuses) the observed properties with the previously estimated properties for each target. If the average detection confidence is low, the confidence in the existence of a target is reduced, and the predicted values are taken as the estimates for the next cycle. If the confidence of existence falls below a threshold, the target is removed from the target list.

The detection phase is used to trigger creation of new targets. In this phase, specified observation modules are executed within a specified list of "trigger" regions. Trigger regions can be specified dynamically, or recalled from a specified list. Target detection is inhibited whenever a target has been predicted to be present within a trigger region.

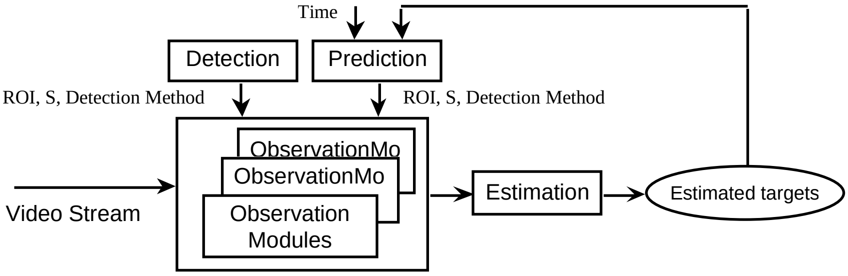

Fig 5. Tracking is a cyclic process of four phases: Predict, Observe, Detect and Estimate. Observation is provided by the observation and grouping modules described above.

A simple zeroth order Kalman filter may be used to track bodies, faces and hands in video sequences. In this model, targets properties are represented by a "state vector" composed of position, spatial extent and orientation (x, y, s_x, s_y, θ). A 5x5 covariance matrix is associated with this vector to represent correlations in errors between parameters. Although prediction does not change the estimated position, it does enlarge the uncertainties of the position and size of the expected target. The expected size provides bounds on the sample rate, as we limit the sample rate so that there are at least 8 pixels across an expected target.

3.3 A Supervisory controller for perceptual processes

The supervisory component of a process provides four fundamental functions: command interpretation, execution scheduling, parameter regulation, and reflexive description. The supervisor acts as a programmable interpreter, receiving snippets of code script that determine the composition and nature of the process execution cycle and the manner in which the process reacts to events. The supervisor acts as a scheduler, invoking execution of modules in a synchronous manner. The supervisor regulates module parameters based on the execution results. Auto-critical reports from modules permit the supervisor to dynamically adapt processing. Finally, the supervisor responds to external queries with a description of the current state and capabilities. We formalize these abilities as the autonomic properties 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 quality of service. 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 process controllers may be instructed to give priority to either the processing rate or precision. The choice of priority is dictated by a more abstract supervisory controller.

An auto-critical process maintains an estimate of the confidence for its outputs. Such a confidence factor is an important feature for the control of processing. Associating a confidence factor to every observation allows a higher-level controller to detect and adapt to changing 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 of their ability to "play the role" required for the service. A higher-level controller can compare responses from several processes and determine the assignment of roles to processes.

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 supervisor. Such descriptions are useful for the dynamic composition of federations of controllers.

3.4 Classes of perceptual processes.

We have identified several classes of perceptual processes. The most basic class is composed of processes that detect and track entities. Entities may generally be understood as spatially correlated sets of properties, corresponding to parts of physical objects. However, correlation may also be based on temporal location or other, more abstract, relations. From the perspective of the system, an entity is any association of correlated observable variables.

Formally, an entity is a predicate function of one or more observable variables.

$$\text{Entity-process}(v_1, v_2, \ldots, v_m) \qquad \text{Entity}(\text{Entity-Class}, \text{ID}, \text{CF}, p_1, p_2, \ldots, p_n)$$

Entities may be composed by an entity detection an tracking processes, as shown in figure 6.

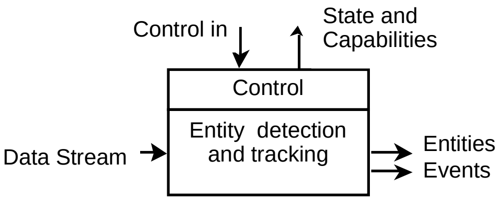

Fig 6. Entities and their properties are detected and described by a special class of perceptual processes.

The input to an entity detection process is typically a stream of numeric or symbolic data. The output of the transformation is a stream including a symbolic token to identify the class 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 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 may be unary, binary, or N-ary. For example, Visible(Entity1), On(Entity1, Entity2), and Aligned(Entity1, Entity2, Entity3) are examples of relations.

Relations that are important for describing situations include 2D and 3D spatial relations, as well as temporal relations [32]. 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 entity detection tracking, we propose to observe relations using Perceptual processes.

Relation-observation processes are defined to transform a list of entities into a list of relations based on their properties, as shown in figure 7. Relation observation processes read in a list of entities tracked by an entity detection and tracking process and produce a list of relations that are true along with the entity or entities that render them true. Relation observation uses tracking to predict and verify relations. Thus they can generate an asynchronous event when a new relation is detected or when a previously detected relations becomes false.

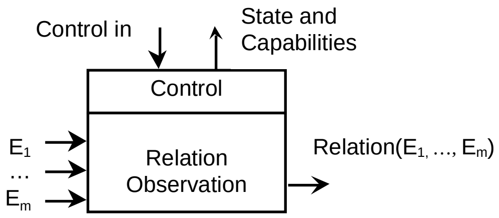

Fig 7. Relations are predicates defined over one or more entities. Relation observation processes generate events when relations become true or false.

Composition processes assemble sets of entities into composite entities. Composition processes are similar to relation observation entities, in that they operate on a list of entities provided by an entity observation process. However, entity observation processes produce a list of composite objects satisfying a set of relations. They can also measure properties of the composite object. As with relation observation processes, composition processes can generate asynchronous events when a composite object is detected or lost.

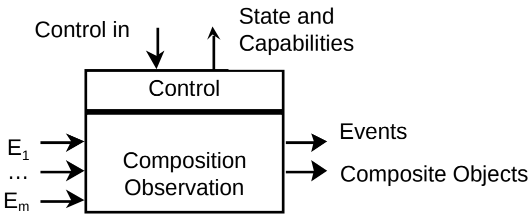

Fig 8. Composition processes observe and track compositions of entities.

3.5 Process federations

Perceptual processes may be organized into software federations [2]. A federation is a collection of independent processes that cooperate to perform a task. We have designed a middle ware environment that allows us to dynamically launch and connect process on different machines. In our system, processes are launched and configured by a "meta-supervisor". The meta-supervisor configures a process by sending snippets of control script to be interpreted by the controller. Each control script defines a command that can be executed by a message from the meta-supervisor. Processes may be interrogated by the meta-supervisor to determine their current state and the current set of commands.

Meta-supervisors can also launch and configure other meta-supervisors so that federations can be built up hierarchically. Each meta-supervisor invokes and controls lower level supervisors that perform the required transformation. At the lowest level are Perceptual processes that observe and track entities and observe the relations between entities. These are grouped into federations as required for to observe the situations in a context.

As a simple example of a federation of perceptual processes, consider a system that detects when a human is in the field of view of a camera and tracks his hands and face. We say that observed regions can be selected to "play the role" of torso, hands and faces. We call this a FaceAndHand observer. The system uses an entity and detection tracking process that can use background difference subtraction and color modeling to detect and track blobs in an image stream. The set of tracked entities are sent to a composition process that labels likely blobs as a torso, face or a left or right hand.

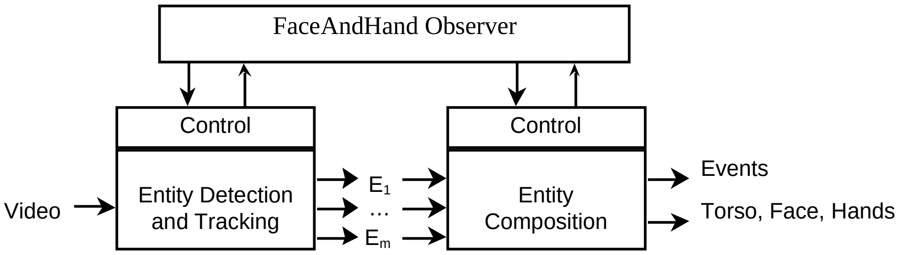

Fig 9. A simple process federation composed of an entity detection process, a composition process and a meta-supervisor.

The control for this process federation works as follows. The meta-supervisor begins by configuring the entity detection and tracking processes to detect a candidate for torso by looking for a blob of a certain size using background subtraction in a pre-configured "detection region". The acceptance test for torso requires a blob detected by background subtraction in the center region of the image, with a size within a certain range. Thus the system requires an entity detection process that includes an adaptive background subtraction detection.

When a region passes the torso test, the composition process notifies the meta-supervisor. The meta-supervisor then configures new trigger regions using color detection modules in the likely positions of the hands and face relative to the torso. The acceptance test for face requires a skin colored region of a certain range of sizes in the upper region of the torso. Hands are also detected by skin color blob detection over a regions relative to the torso. Sets of skin colored regions are passed to the composition process so that the most likely regions can be assigned to each role. We say that the selected skin-colored regions are assigned the "roles" of face, left hand and right hand. The assignments are tracked so that a change in the entity playing the role of hand or face signals an event. Such role assignment is a simple example of a more general principle developed in the next section.

4. Context and Situation

Perceptual processes provide a means to detect and track compositions of entities and to verify relations between entities. The design problem is to determine which entities to detect and track and which relations to verify. For most human environments, there is a potentially infinite number of entities that could be detected and an infinite number of possible relations for any set of entities. The appropriate entities and relations must be determined with respect to a task or service to be provided.

In this section we discuss the methods for specifying context models for human activity, We define the concept of a "role" and explain how roles can help simplify context models. We define three classes of events in such systems, and describe the system reaction to each class. We then present two examples of simple context models. An early version of the concepts presented in this section was presented in [33] . This paper refines and clarifies many aspects of this framework in the light of experience with implementing systems based on this model.

4.1 Specifying a context model

A system exists in order to provide some set of services. Providing services requires the system to perform actions. The results of actions are formalized by defining the output "state" of the system. Simple examples of actions for interactive environments include adapting the ambient illumination and temperature in a room, or displaying a users "availability for interruption". More sophisticated examples of tasks include configuring an information display at a specific location and orientation, or providing information or communications services to a group of people working on a common task.

The "state" of an environment is defined as a conjunction of predicates. The environment must act so as to render and maintain each of these predicates to be true. Environmental predicates may be functions of information observed in the environment, including the position, orientation and activity of people in the environment, as well as position, information and state of other equipment. The information required to maintain the environment state determines the requirements of the perception system.

The first step in building a context model is to specify the desired system behavior. For an interactive environment, this corresponds to the environmental states, defined in terms of the variables to be controlled by the environment, and predicates that should be maintained as true. For each state, the designer then lists a set of possible situations, where each situation is a configuration of entities and relations to be observed. Although a system state may correspond to many situations, each situation must uniquely belong to one state. Situations form a network, where the arcs correspond to changes in the relations between the entities that define the situation. Arcs define events that must be detected to observe the environment.

In real examples, we have noticed that there is a natural tendency for designers to include entities and relations that are not really relevant to the system task. Thus it is important to define the situations in terms of a minimal set of relations to prevent an explosion in the complexity of the system. This is best obtained by first specifying the system output state, then for each state specifying the situations, and for each situation specifying the entities and relations. Finally for each entity and relation, we determine the configuration of perceptual processes that may be used.

4.2 Simplifying context models with roles

The concept of role is an important (but subtle) tool for simplifying the network of situations. It is common to discover a collection of situations for an output state that have the same configuration of relations, but where the identity of one or more entities is varied. A role serves as a "variable" for the entities to which the relations are applied, thus allowing an equivalent set of situations to have the same representation. A role is played by an entity that can pass an acceptance test for the role. In that case, it is said that the entity can play or adopt the role for that situation. In our framework, the relations that define a situation are defined with respect to roles, and applied to entities that pass the test for the relevant roles.

For example, in a group discussion, at any instant, one person plays the "role" of the speaker while the other persons play the role of "listeners". Dynamically assigning a person to the role of "speaker" allows a video communication system to transmit the image of the current speaker at each instant. Detecting a change in roles allows the system to reconfigure the transmitted image.

Entities and roles are not bijective sets. One or more entities may play a role. A role may be played by one or several entities. The assignment of entities to roles may (and often will) change dynamically. Such changes provide the basis for an important class of events : role-events. Role events signal a change in assignment of an entity to a role, rather than a change in situation.

Roles and relations allow us to specify a context model as a kind of "script" for activity in an environment. However, unlike theater, the script for a context is not necessarily linear. Context scripts are networks of situations where a change in situations is determined based on relations between roles.

4.3 Context and situation

To summarize, a context is a composition of situations that concerns a set of roles and relations. A context determines the configuration of processes necessary to detect and observe the entities that can play the roles and the relations between roles that must be observed. The roles and relations should be limited to the minimal set necessary for recognizing the situations necessary for the environmental task. All of the situations in a context are observed by the same federation.

Context $\{\text{Role}_1, \text{Role}_2, \ldots, \text{Role}_n; \text{Relation}_1, \ldots, \text{Relation}_m\}$

A situation is a kind of state, defined by a conjunction of relations. Relations are predicate functions evaluated over the properties of the entities that have been assigned to roles. A change in the assignment of an entity to a role does not change the situation, unless a relation changes in value.

Entities are assigned to roles by role assignment processes. The context model specifies which roles are to be assigned and launches the necessary role assignment processes. A meta-supervisor determines what kind of entities can play each role, and launches processes to detect and observe these entities. A description of each detected entity is returned to the role assignment process where it is subjected to the acceptance test to determine its suitability for the role based on type, properties and confidence factor. The most suitable entity (or entities) is (are) assigned to the roles. Relations are then evaluated, and the set of relations determines the situation.

The situation is a set of relations computed on the entities assigned to roles. Situation changes when the relations between entities change. If the assignment of entities to situations changes, the situation remains the same. However, the system may need to act in response to a change in role assignment. For example, if the person playing the role of speaker changes, then a video communication system may need to change the camera view to center on the new speaker.

4.4 Classes of events.

From the above, we can distinguish three classes of events: Role Events, Relation events and Context events.

Role events signal a change in the assignment of entities to roles. Such a may result in a change in the system output state and thus require that the system act so as to bring the state back to the desired state. For example, the change in speaker (above) renders a predicate Camera-Aimed -At(Speaker) false, requiring the system to selected the appropriate camera and orient it to the new speaker. Situation events or (relation events) signal changes in relations that cause a change in situation. If the person playing the role of speaker stops talking and begins writing on a blackboard, then the situation has changed. Context events signal changes in context, and usually require a reconfiguration of the perceptual processes.

Role events and 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 Perceptual processes. Context events may be driven by data, or by some external system command .

4.5 Simple example: An interuptibility meter

As first simple example, consider a system whose task is to display the level of "interruptibility" of a user in his office environment. Such a system may be used to automatically illuminate a colored light at the door of the office, or it may be used in the context of a collaborative tool such as a media-space [34]. The set of output actions are very simple. The environment should display one of a set of interruptibility states. For example, states could be "Not in Office", "Ok for interruptions", "urgent interrupts only" and "Do not Disturb".

Suppose that the user has decided that his four interruptibility states depend on the following eight situations, labeled s1 to s8: (S1) The user is not in office when the user is not present. He is interruptible when (S2) alone in his office or (S3) not working on the computer or (S4) talking on the phone. He may receive urgent interruptions when (S5) working at his computer, or when (S6) visitors are standing in the office. The user should not be interrupted (S7) when on the phone, or (S8) when the visitors are sitting in his office.

The roles for these situations are <User>, <Visitor>, <Computer>, <Phone>. The <User> role may be played by a human who meets an acceptance test. This test could be based on an RFID badge, a face recognition system, a spoken password, or a password typed into a computer. A <Visitor> is any person in the office who has not met the test for <User>. A person is a class of entity that is detected and tracked by a person observation process. For example, this can be a simple visual tracker based on subtraction from an adaptive background.

The predicate "Present(User)" would be true whenever a person observed to be in the office has been identified as the <User>. The fact that entities are tracked means that the person need only be identified once. Evaluating the current situation requires applying a logical test for each person. These tests can be applied when persons enter the office, rather then at each cycle.

Situation S1 would be true if the no person being tracked in the office passes the test for user. Situation S2 also requires a predicate to know if a person is playing the role of visitor. States S4 and S7 require assigning an entity to the role <Phone>. This can be done in naïve manner by assigning regions of a certain color or texture at a certain location. However, if we wish to include cellular phones we would need more sophisticated vision processes for the role assignment.

For assigning an object to the role of <Computer> a simple method would be to consider a computer as a box of a certain color at a fixed location. The <User> could then be considered to be using the computer if a face belonging to his torso is in a certain position and orientation. Facing would normally require estimating the position and orientation of a person's face. The test would be true if the orientation of the position of the face was within a certain distance of the computer and the orientation of the face were opposite the compute screen. Again, such tests could be arbitrarily sophisticated, arbitrarily discriminant and arbitrarily costly to develop. Situations S6 and S8 require tests for persons to be sitting and standing. These can be simple and naïve or sophisticated and expensive depending on how the system is to be used.

4.6 Second example: A video based collaborative work environment

As a second example, consider a video based collaborative work environment. Two or more users are connected via high bandwidth video and audio channels. Each user is seated at a desk and equipped with a microphone, a video communications monitor and an augmented work surface. Each user's face and eyes are observed by a steerable pan-tilt-zoom camera. A second steerable camera is mounted on the video display and maintains a well-framed image of the user's face. The augmented workspace is a white surface, observed by a third video camera mounted overhead.

The system task is to transmit the most relevant image of the user. If the user is facing the display screen, then the system will transmit a centered image of the users face. If the user faces his drawing surface, the system will transmit an image of the drawing surface. If the user is facing neither the screen nor the drawing surface then the system will transmit a wide-angle image of the user within the office. This can be formalized as controlling two functions: transmit(video-stream) and center(camera, target). For each function, there is a predicate that is true when the actual value corresponds to the specified value.

The roles that compose the context are 1) the user 2) the display screen, 3) a writing surface. The user is a composite entity composed of a torso, face and hands. The system's task is to determine one of three possible views of the user: a well-centered image of the user's face, the user's workspace and an image of the user and his environment. Input data include the microphone signal strength, and a coarse resolution estimation of the user's face orientation. The system context includes the roles “speaker” and “listener”. At each instant, all users are evaluated by a meta-supervisor to determine assignment to one of the roles "speaker" and “listener”. The meta-supervisor assigns one of the users to the role speaker based on recent energy level of his microphone. Other users are assigned the role of listener. All listeners receive the output image of the speaker. The speaker receives the mosaic of output images of the listeners.

The user may place his attention on the video display, or the drawing surface or “off into space”. This attention is manifested by the orientation of his face, as measured by positions of his eyes relative to the center of gravity of his face (eye-gaze direction is not required). When the user focuses attention on the video display, his output image is the well-framed image of his face. When a user focuses attention on the work surface, his output image is his work-surface. When the user looks off “into space”, the output image is a wide-angle view of the user’s environment. This system uses a simple model of the user’s context completed by the system’s context to provide the users with the appropriate video display. Because the system adapts its display based on the situation of the group of users, the system, itself, fades from the user’s awareness.

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 “played” by one or more entities. A relation is a predicate defined over the properties of entities. A situation is a configuration of relations between the entities.

This ontology provides the basis for a software architecture for the perceptual components of context aware systems. Observations are provided by perceptual processes defined by a tracking process or transformation controlled by reflexive supervisor. Perceptual processes are invoked and organized into hierarchical federations by reflexive meta-supervisors. A model of the user’s context makes it possible for a system to provide services with little or no intervention from the user.

Acknowledgment

This work has been partly supported by the EC project IST FAME project (IST-2000-28323) and IST CAVIAR (IST 2001-37540) as well as French national project RNTL/ProAct CONTACT. This work has been performed in active collaboration with Joelle Coutaz, Gaetan Rey, Patrick Reignier, Dave Snowdon, Jean-Luc Meunier and Alban Caporossi.

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 and S. Kubica, “The Khoros application development environment “, in Experimental Environments for computer vision and image processing, H. Christensen and 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] T. Winograd, “Architecture for Context”, Human Computer Interaction, Vol. 16, pp401-419.

[7] R. C. Schank and R. P. Abelson, Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[8] M. Minsky, "A Framework for Representing Knowledge", in: The Psychology of Computer Vision, P. Winston, Ed., McGraw Hill, New York, 1975.

[9] M. R. Quillian, "Semantic Memory", in Semantic Information Processing, Ed: M. Minsky, MIT Press, Cambridge, May, 1968.

[10] D. Bobrow: "An Overview of KRL", Cognitive Science 1(1), 1977.

[11] A. R. Hanson, and E. M. Riseman, , VISIONS: A Computer Vision System for Interpreting Scenes, in Computer Vision Systems, A.R. Hanson & E.M. Riseman, Academic Press, New York, N.Y., pp. 303-334, 1978.

[12] B. A.Draper, R. T. Collins, J. Brolio, A. R. Hansen, and E. M. Riseman, "The Schema System", International Journal of Computer Vision, Kluwer, 2(3), Jan 1989.

[13] M.A. Fischler & T.A. Strat. Recognising objects in a Natural Environment; A Contextual Vision System (CVS). DARPA Image Understanding Workshop, Morgan Kauffman, Los Angeles, CA. pp. 774-797, 1989.

[14] R. Bajcsy, Active perception, Proceedings of the IEEE, Vol. 76, No 8, pp. 996-1006, August 1988.

[15] J. Y. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active Vision", International Journal of Computer Vision, Vol. 1, No. 4, Jan. 1988.

[16] J. L. Crowley and H. I Christensen, Vision as Process, Springer Verlag, Heidelberg, 1993.

[17] B. Schilit, and M. Theimer, “Disseminating active map information to mobile hosts”, IEEE Network, Vol 8 pp 22-32, 1994.

[18] P. J. Brown, “The Stick-e document: a framework for creating context aware applications”, in Proceedings of Electronic Publishing, ’96, pp 259-272.

[19] T. Rodden, K.Cheverest, K. Davies and A. Dix, “Exploiting context in HCI design for mobile systems”, Workshop on Human Computer Interaction with Mobile Devices 1998.

[20] A. Ward, A. Jones and A. Hopper, “A new location technique for the active office”, IEEE Personal Comunications 1997. Vol 4. pp 42-47.

[21] K. Cheverest, N. Davies and K. Mitchel, “Developing a context aware electronic tourist guide: Some issues and experiences”, in Proceedings of ACM CHI ’00, pp 17-24, ACM Press, New York, 2000.

[22] J. Pascoe “Adding generic contextual capabilities to wearable computers”, in Proceedings of the 2nd International Symposium on Wearable Computers, pp 92-99, 1998.

[23] Dey, A. K. “Understanding and using context”, Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.

[24] A. Lux, "The Imalab Method for Vision Systems", International Conference on Vision Systems, ICVS-03, Graz, april 2003.

[25] 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.

[26] M. Storring, H. J. Andersen and E. Granum, "Skin color detection under changing lighting conditions", Journal of Autonomous Systems, June 2000.

[27] J. Piater and J. Crowley, "Event-based Activity Analysis in Live Video using a Generic Object Tracker", Performance Evaluation for Tracking and Surveillance, PETS-2002, Copenhagen, June 2002.

[28] 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.

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

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

[31] 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.

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

[33] J. L. Crowley, J. Coutaz, G. Rey and P. Reignier, "Perceptual Components for Context Aware Computing", UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg, Sweden, September 2002.

[34] J. L. Crowley, J. Coutaz and F. Berard, "Things that See: Machine Perception for Human Computer Interaction", Communications of the A.C.M., Vol 43, No. 3, pp 54-64, March 2000.

[35] Schilit, B, N. Adams and R. Want, "Context aware computing applications", in First international workshop on mobile computing systems and applications, pp 85 - 90, 1994.

[36] Dey, A. K. "Understanding and using context", Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.