Perceptual Components for Context Aware Computing

James L. Crowley[1], Joëlle Coutaz[2], Gaeten Rey[2] and Patrick Reignier[1]

[1] Laboratoire GRAVIR, INRIA Rhône Alpes,
655 Ave de l'Europe, F-38330 Montbonnot, France
{Crowley, Reignier}@inrialpes.fr
http://www-prima.imag.fr
[2] CLIPS-IMAG, BP 53, Université Joseph Fourier
F-38051 St. Martin D'hères, France
{Coutaz, Rey}@imag.fr

Abstract. In this paper we propose an ontology and a software architecture for observing and modeling context and situation. We are especially concerned with the perceptual components for context awareness. We propose a model in which a user's context is described by a set of roles and relations. Different configurations of roles and relations correspond to situations within the context. We define an ontology for context awareness from both a bottom up system's perspective and a top-down users' perspective. As we define each element, we describe the corresponding components of a process-based software architecture. Using these components, a context is translated into a federation of observational processes. 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.

1 Introduction

Available technologies increasingly enable computing and communication to migrate out of the "gray box" and into ordinary objects. An inevitable result is a multiplication of digitally controlled devices with increasingly complex capabilities. In too many cases, the designers of such devices are forced to rely on the human capacity to learn. While standardization of interaction techniques may provide some help, ordinary humans are increasingly required to divert attention to "futzing with the device". There is an obvious need for methods for building systems that model the activities a human users and anticipate their needs. Much of the work on such systems focuses on defining and modeling "context" for interaction.

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 define an ontology for context awareness by proceeding top down, from the users' perspective, as well as bottom up, from the perspective of the system. As we define each element, we describe the corresponding components of a process-based software architecture. 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. Throughout the paper, we illustrate the components of the architecture with a system that observes and tracks faces. The result is both a clear, well-defined ontology for describing context and situation and a software architecture for building real systems. 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.

2. A Brief history of context

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 definition. 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. The inadequacy of the purely linguistic basis for meaning provided by schema was ultimately recognized as the problem of "grounding". Recognition of the "grounding-

problem" was responsible for turning a generation of AI researchers away from a purely linguistic theory of AI towards a theory of intelligence based on action and perception [11]. Purely symbolic context is now recognized as inadequate for intelligence. Intelligence requires the ability to perceive and to act.

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" [12] 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 sort of "Holy Grail" in computer vision. Successive generations of such systems, such as the "Schema System"[13] and "Condor" [14] floundered on problems of unreliable image description and computational complexity. Interest in the 1990's turned to achieving real time systems using "active vision" [15], [16]. Many of these ideas were developed and integrated into a context driven interpretation using a process architecture using the approach "Vision as Process" [17]. The methods for sensing and perceiving context for interaction described below draws from this approach.

The term "Context Aware" was first introduced to the mobile computing community by Schilit and Theimer [18]. 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 [19] [20] and [21] 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 [22] 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 [23] 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 [24] reviews definitions of context, and provides a definition of context as "any information that can be used to characterize situation". We are in agreement with Dey in his identification of "situation" as the current state of the environment and "context" as the elements by which situation is defined. However, to apply context in the composition of perceptual processes, we need to complete a clear semi-formal definition with an operational theory.

3. Fundamental Concepts

In order to provide an operational theory of context awareness, in this section, we develop an ontology for context and situation. As we develop each term of the ontol-

ogy, 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.

3.1 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 [25]. The HCI and mobile computing communities tend to criticize the concept of rationality used in planning in Artificial Intelligence as too single-minded to properly model users. However, such definitions can provide a formal basis for discussing tasks and activities.

The fundamental concept for a formal definition of task is that of state [26]. 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 [27]. 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.

3.2 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.

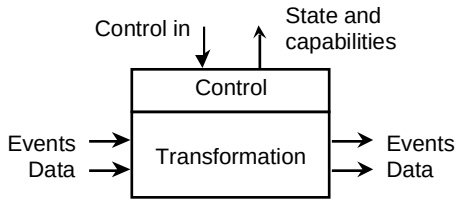

Fig. 1. An observational process transforms data and events into data and events.

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.

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* [28], which is a conceptual extension of the context widget implemented in the Context Toolkit [29].

3.3 Examples: processes for observing, grouping and tracking

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, as illustrated in figure 2. 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 [30].

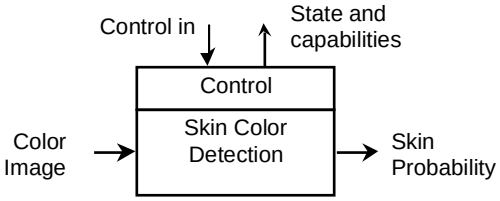

Fig 2. An observational process for detecting skin colored pixels

Let us define the chrominance vector (C_1,C_2) for each pixel as the red and green component normalized by luminance. Such a chrominance vector is a color signature that depends on both the skin pigment of an individual and the source color of the illumination [31]. Because of luminance normalization, the skin pixels for an individual in a scene will all exhibit the same chrominance vector and can be used to detect the hands or face of that individual.

$$C_1 = \text{Round}(Q\ \frac{R}{R+G+B}) \qquad C_2 = \text{Round}(Q\ \frac{G}{R+G+B})$$

The chrominance values can be converted to integers in the range of [0,Q], by multiplying by Q and rounding. If chrominance is suitably quantized, a probability density function for the chrominance of the entire image can be compiled by compiling a histogram of chrominance. A value of Q=32 will generally work well, but this can be a run time parameter. A probability density function for skin colored regions can be approximated by identifying the skin colored regions by some outside means in a set of calibration images. Histograms of the chrominance from all pixels $h_{tot}(C_1,C_2)$ and from all skin colored pixels $h_{skin}(C_1,C_2)$ may then be compiled.

(i,j) Image :

$h_{tot}(C_1(i,j),C_2(i,j)) := h_{tot}(C_1(i,j),C_2(i,j)) + 1;$

$M_{tot} := M_{tot} + 1;$

(i,j) SkinRegion :

$h_{skin}(C_1(i,j),C_2(i,j)) := h_{skin}(C_1(i,j),C_2(i,j)) + 1;$

$M_{skin} := M_{skin} + 1;$

Provided that the number of sample pixels, M_{skin} and M_{Tot} are larger than $10 \cdot Q^2$, the histogram $h_{tot}(C_1,C_2)$ provides an estimate for the probability density function for chrominance $p(C_1,C_2)$ while $h_{skin}(C_1,C_2)$ provides an estimate of the conditional density function for chrominance given skin, $p(C_1,C_2 \mid \text{skin})$.

$$p_{Tot}(C_1,C_2) \approx \frac{1}{M_{Tot}} h_{Tot}(C_1,C_2) \qquad p(C_1,C_2 \mid \text{skin}) \approx \frac{1}{M_{Skin}} h_{Skin}(C_1,C_2)$$

Baye's rule shows that the ratio of the two histograms provides a look up table that gives conditional probability for skin at each pixel given its chrominance. 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 skin using the look-up table $T_{ratio}(C_1,C_2)$.

$$p(skin|C_1,C_2) = \frac{p(C_1,C_2|skin)p(skin)}{p(C_1,C_2)} \approx \frac{h_{skin}(C_1,C_2)}{h_{Tot}(C_1,C_2)} = T_{ratio}(C_1,C_2)$$

To further reduce the computation time for this process, we provide the possibility that processing is restricted to a rectangular "Region of Interest" or ROI.

$$(i,j) \quad ROI : w(i,j) := T_{ratio}(C_1(i,j),C_2(i,j))$$

The ROI is an example of control information provided via a supervisory controller.

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.

Entity-process(v_1, v_2, …, v_m) Entity(Entity-Class, ID, CF, p_1, p_2,…, p_n)

Entities may be composed by a entity grouping processes, as shown in figure 3.

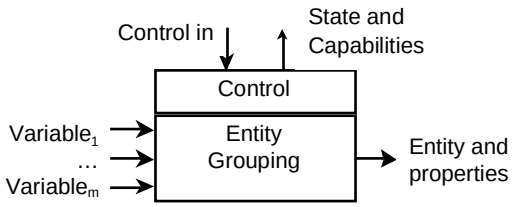

Fig 3. Entities and their properties are detected and described by a special class of observational processes.

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.

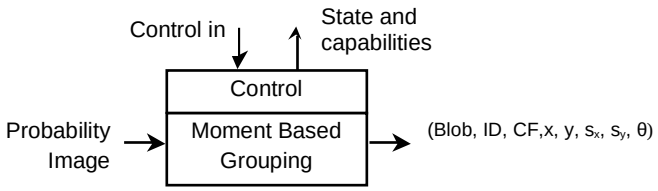

Fig 4. A observational process for grouping skin colored regions.

A simple example of an entity detection process is provided by a process that groups adjacent skin colored pixels into regions (commonly called blobs), as shown in figure 4. Such a process can be easily defined based on moments. 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.

$$N = R \quad C$$

$$M = \sum_{i,j \quad ROI} w(i,j)$$

$$CF = \frac{M}{N}$$

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.

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

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.

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

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

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 [Allen 83]. 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.

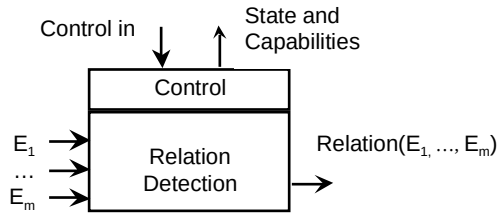

Fig 5. Relations between entities are detected by relation detection processes

As before, this transformation may be triggered by and may generate asynchronous symbolic messages that can serve as asynchronous events.

Relation-process(E_1, E_2, …, E_m) (Relation-Class, ID, E_1, E_2,…, E_n)

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 [33] 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.

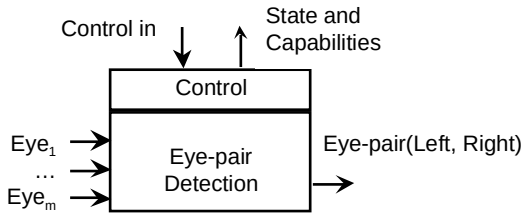

Fig 6. An Eye pair detector uses size and position to determine pairs of possible eye entities that satisfy the relation "eye-pair".

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 makes it possible to know the identity of a blob that is currently observed based on an earlier recognition. 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 [34] is

beyond this paper. A general discussion of the use of the Kalman filter for sensor fusion is given in [35]. The use of the Kalman filter for tracking faces is described in [36]. 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).

A simple example of a federation of observation processes shown in figure 7. This process takes in color images and produces the current position of a skin blob. Each process. However, as presented , this federation lacks a mechanism to initiate the tracking, to initialize the parameters , and to globally adapt parameters to maintain a desired quality of service. These functions can be provided by using a higher-level supervisory controller to initiate and supervise the federation, as described in the next section.

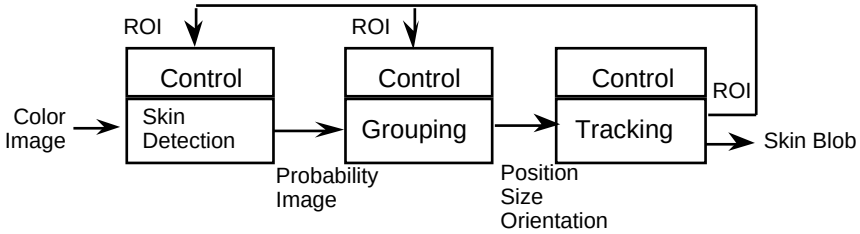

Fig. 7 A process federation for observing skin colored blobs

3.4 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.

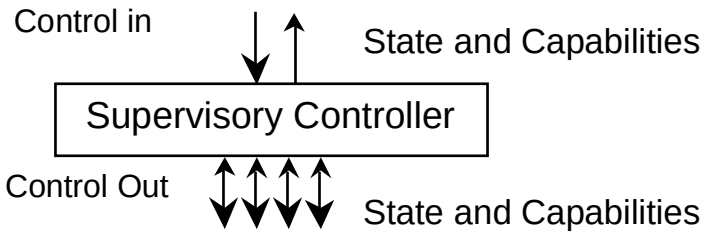

Figure 8. A supervisory controller uses observational process to associate entities with roles and to determine the relations between entities.

The skin blob tracker provides an example of such a controller. The supervisory controller, labeled as “skin region tracker” in figure 9 invokes and coordinates observa-

tional 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.

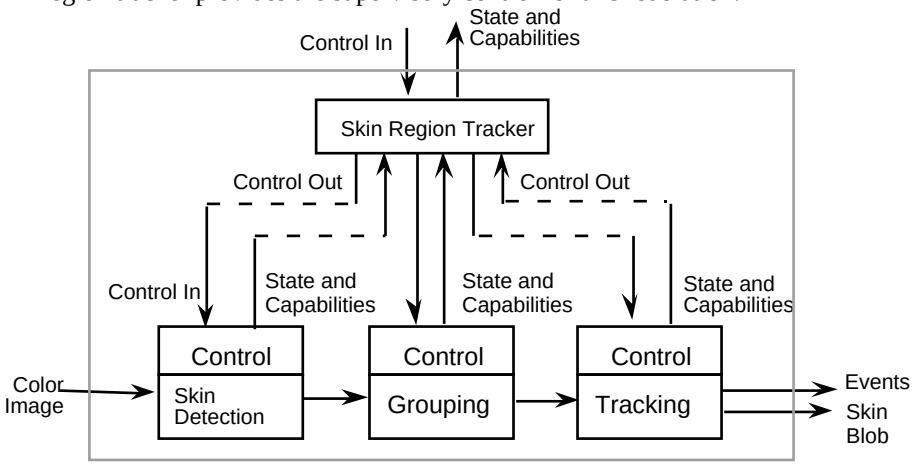

Fig 9 A federation of processes for observing skin colored blobs. A second level supervisory controller invokes the first level observational processes, and supervises their execution.

4. 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.

4.1 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.

$$\text{Context}(U,T) \quad \{\text{Role}_1, \text{Role}_2,\ldots,\text{Role}_n; \text{Relation}_1,\ldots,\text{Relation}_m\}$$

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.

$$\text{Role}(E_1, E_2, \ldots, E_m) \quad (\text{Role-Class, ID, CF}, E_1, E_2, \ldots, E_n)$$

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.

4.2 Example: A video based collaborative work environment

As a simple example of a user's context, consider a video based collaborative working 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 entities that compose the user's context are 1) the writing surface, 2) one or more pens, 3) the other users, and 4) the other users' writing surfaces. The roles of the user's context are 1) the current focus of attention, 2) the drawing tool, and 3) the pointer. The focus of attention may be "assigned" by the user to the drawing surface, to another user, or to another user's workspace. Relations for entities include "looking at", "pointing at", "talking to", and "drawing on". Situations include "user speaking",

"user listening", "user drawing", "user pointing while speaking", and "user drawing while speaking". If the system can properly evaluate and respond to the user's situation, then other objects, such as the video display, disappear from the users focus of attention.

The system's model of context includes the users and the entities that make up their contexts. It also includes three possible views of the user: a well-centered image of the user's face, user's workspace or an image of the user and his environment. Observable variables 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, one of the users is assigned the role of the "speaker". The other users are assigned the role of "listener". The system uses a test on the recent energy level of the microphones to determine the current speaker.

Each 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. All listeners receive the output image of the speaker. The speaker receives the mosaic of output images of the listeners.

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. Observing Context with 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 supervisor 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.

5.1 Properties for Observational Processes

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 skin region tracking process described above requires a certain number of operations per pixel. Let us call this number P. The time for processing a ROI, T, is thus directly proportional with the number of pixels in the ROI, N.

$$T = N \cdot P.$$

Video rate processing requires placing a bounds T_{max} on T. The size of the ROI is determined by the size of the target in the previous image. If the number of pixels in the ROI exceeds, T_{max}/P then some pixels must be skipped. A simple means to reduce the number of pixels is to treat only one out of S pixels. The resulting observation is still valid, but its precision will have been degraded by a factor of S. Let us refer to S as the "step size" used in computing the probability of skin and in grouping the skin colored pixels. The supervisory controller may be instructed to give priority to either the processing rate (increasing S so as to maintain a time less than T_{max}) or precision, increasing T_{max} so as to maintain S. 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.

5.2 A federation for observing faces orientation

A skin colored region of a certain size, aspect ratio and orientation, may potentially be a face. To determine if such a region is a face, a face detection controller may apply a test to the length, breadth and orientation of the skin colored region. A confidence that the region is a face can be used by a supervisory controller to detect and initiate tracking of the user's face.

If the regions properties pass the acceptance test, then a observational processes for detecting eyes may be applied within the ROI defined for the face. Detected eye entities are fed to a relation test for an "eye-pair" detector. The eye-pair detector and the skin blob are then fused to form a face entity. This face entity is tracked using a Kalman filter based entity tracker. The result is a face detection controller that recruits skin colored regions to play the role of face, and then applies a further test to validate the face hypothesis, as shown in figure 10.

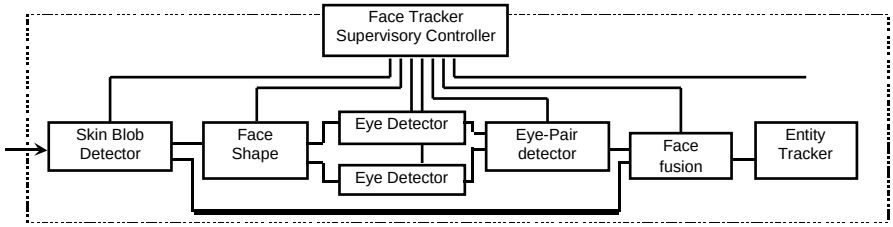

Fig 10 A second level federation that tracks faces. The skin blob detector is provided

by the federation described in figure 9. The dark line indicate data and meta-data flow between processes.

5.3 Creating process federations for observing context

A crucial problem with this model is how to provide a mechanism for dynamically composing federations of supervisory controllers that observe the entities and relations relative to the user's context. Our approach is to propose a "meta-controller". The meta-controller is designed for a specific domain. As described above, the domain is composed of a network of possible user contexts, and the associated systems contexts. The meta-controller maintains a model of the current user's context. This model includes information about adjacent contexts that may be attained from the current context, as well as the user and system context events that may signal such a change.

The meta-controller may be seen as a form of reactive expert system. For each user context, it invokes and revokes the corresponding highest-level supervisory controllers. These controllers, in turn, invoke and revoke lower level controllers, down to the level of the lowest level observational processes. Supervisory controllers may evoke competing lower-level processes, informing each process of the roles that it may play. The selection of process for a role can then be re-assigned dynamically according to the quality of service estimate that each process provides for its parent controller.

6. 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.

Acknowledgment

This work has been partly supported by the EC project TMR TACIT (ERB-FMRX-CT-97-0133) and by the IST-FET GLOSS project (IST-2000-26070) and IST FAME project (IST-2000-28323). It has been conducted with the participation of G. Calvary.

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] T. Winograd, “Architecture for Context”, Human Computer Interaction, Vol. 16, pp401-419.

[7] Shanks 77

[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] R. Brooks, , "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, RA-2, no. 1, 1986.

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

[19] P. J. Brown, “The Stick-e document: a framework for creating context aware applications”, in Proceedings of Electronic Publishing, ’96, pp 259-272.
[20] 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.
[21] A. Ward, A. Jones and A. Hopper, “A new location technique for the active office”, IEEE Personal Comunications 1997. Vol 4. pp 42-47.
[22] 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.
[23] J. Pascoe “Adding generic contextual capabilities to wearable computers”, in Proceedings of the 2nd International Symposium on Wearable Computers, pp 92-99, 1998.
[24] Dey, A. K. “Understanding and using context”, Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.
[25] Newell, A. "The Knowledge Level", Artificial Intelligence 28(2), 1982.
[26] Nilsson, N. J. Principles of Artificial Intelligence, Tioga Press, 1980.
[27] R. Korf, "Planning as Search", Artificial Intelligence, Vol 83, Sept. 1987.
[28] J. Coutaz and G. Rey, “Foundations for a Theory of Contextors”, in Computer Aided Design of User Interfaces, Springer Verlag , June 2002.
[29] 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.
[30] 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.
[31] M. Storring, H. J. Andersen and E. Granum, "Skin color detection under changing lighting conditions", Journal of Autonomous Systems, June 2000.
[32] J. Allen, "Maintaining Knowledge about Temporal Intervals", Journal of the ACM, 26 (11) 1983.
[33] 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.
[34] R. Kalman, "A new approach to Linear Filtering and Prediction Problems", Transactions of the ASME, Series D. J. Basic Eng., Vol 82, 1960.
[35] J. L. Crowley and Y. Demazeau, “Principles and Techniques for Sensor Data Fusion“, Signal Processing, Vol 32 Nos 1-2, p5-27, May 1993.
[36] 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.
[37] 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.
[38] 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.
[39] Dey, A. K. “Understanding and using context”, Personal and Ubiquitous Computing, Vol 5, No. 1, pp 4-7, 2001.