An Architecture for Context Aware Observation of Human Activity

James L. Crowley, Patrick Reignier
Project PRIMA, INRIA Rhône Alpes,
655 Ave de l'Europe, F-38334 Montbonnot, France

1. Introduction

We describe a process-based software architecture for context aware observation of human activity. This model builds on recent work on architectures for machine perception and computer vision [1], [2], as well as on data flow models for software architectures [3]. The basic components of this model are modules, defined as transformations of observations. Modules are assembled into reflexive processes under the direction of a supervisory controller. Federations of processes are dynamically assembled [4], [5] according to a model of the activities to be observed. This model is based on an ontology for context aware systems described in [6].

2. Modules

The basic component of our architecture is a software module. A module is defined by a transformation controlled by a set of parameters. A module returns a description of the results of processing in the form of a state vector.

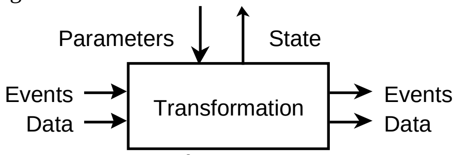

Fig. 1. A module transforms data and events according to control parameters.

The input is generally composed of some raw numerical values, generally arriving in a synchronous stream, accompanied by meta-data. Meta-data may be information such as a time-stamp, a confidence factor, a priority or a description of precision. However, modules may also process events. 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.

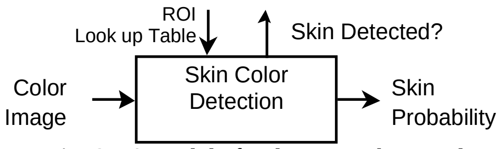

Fig 2. A module for detecting skin pixels

A simple example of a module is provided by a transformation that uses a look-up table to convert color pixels into a probabilities that the pixels are 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 [7]. Computation time for this process may be reduced by restricting processing to a rectangular "Region of Interest" or ROI.

A grouping module collects regions of detected pixels into "blobs" described by a vector of properties. An example is provided by a module that groups probabilities into regions (commonly called blobs) using moments, as shown in figure 3. In this module, the detection mass is the sum of the probabilities in the ROI. The first moment is the center of gravity in the row and column directions. This is a robust indicator of the position of the skin colored blob. The second moment is a covariance matrix. The square root of the principle components are the length and width of the region. The principal vector indicates the dominant direction of the region. Principal components analysis of the covariance matrix formed from σ_{ii}^2, σ_{ij}^2,and σ_{jj}^2 yield the length and breadth of the blob (s_x, s_y) as well as its orientation θ.

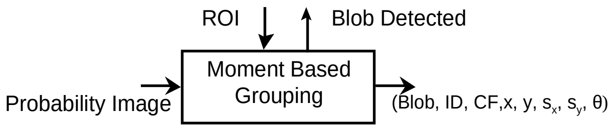

Fig 3. A module for grouping detected pixels into blobs.

Tracking blobs provides a number of interesting benefits. Tracking preserves information over time. For example, with tracking, it is only necessary to recognize a blob once. Tracking also makes it possible to compose a history of the positions of an blob. Finally, tracking can be used to reduce computation by focusing attention. The position predicted by tracking may be used to specify the ROI for detection and grouping.

Tracking is a process of recursive estimation, generally implemented as a Kalman filter. A general discussion of the use of the Kalman filter for sensor fusion is given in [8]. The use of the Kalman filter for tracking faces is described in [9]. 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).

These three modules for skin detection, blob grouping and blob tracking could easily be combined into a cyclic data-flow process. However, such a process would lack a mechanism to initiate tracking, to initialize the parameters, and to globally adapt parameters to maintain a desired quality of service. These functions can be provided by embedding the modules in a process controlled by a reflexive supervisory controller.

3 Observational Processes

An observational process is composed of an assembly of modules governed by a control component, as illustrated in figure 4. The control component interprets commands and parameters, supervises the execution of the transformation components, and responds to queries with a description of the current state and capabilities of the process. This model is similar to that of a *contextor* [10], which is a conceptual extension of the context widget implemented in the Context Toolkit [11].

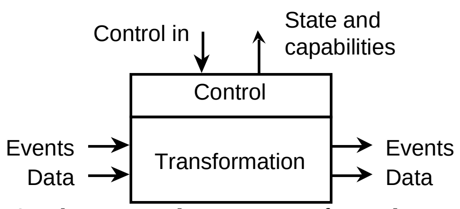

Fig. 4. An observational process transforms data and events under the direction of a reflexive controller.

Dynamic assembly and control of observational processes requires information about the capabilities and the current state of component processes. Such information can be provided by providing the controller with the 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 quality of service. For example, processing time and precision are two important state variables for a tracking process. These two parameters 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-descriptive process provides a symbolic description of its capabilities and state. The description of the capabilities may include either the basic command set of the controller or a set of services that the controller may provide to a more abstract meta-controller.

An auto-critical process maintains an estimate of the confidence for its outputs. Associating a confidence factor to observations allows a higher-level controller to detect and adapt to changing observational circumstances. When supervisory controllers are programmed to offer "services" to higher-level meta-controllers, it can be very useful to include an estimate of the confidence for the role. A higher-level meta-controller can compare these responses from several processes and determine the assignment of roles to processes.

The skin blob observer, shown in figure 5, provides an example of an observational process. A supervisory controller, labeled as "skin blob tracker" invokes and coordinates modules for skin detection, pixel moment grouping and tracking. The module state describes the number of blobs being tracked as well as information about the current processing cycle time and precision. In our system we have implemented the control component using an interpreter for the scheme (lisp) programming language, thus allowing code snippets to be downloaded and added to the module's capabilities

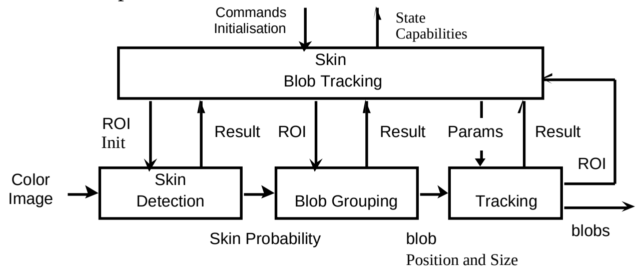

Fig. 5. An process for observing skin colored blobs.

4. Observing Entities and Relations

A fundamental aspect of interpreting sensory observations is grouping observations to form entities. Entities may generally be understood as corresponding to physical objects. However, 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 also be based on temporal location or other, more abstract relations. We define an entity as a predicate function of one or more observations, provided by an entity grouping process.

We define a situation as a configuration of entities and their relations. Relations are defined as a predicate of the properties of entities. Relations that are important for describing human context include 2D and 3D spatial relations, as well as temporal relations [12]. Other sorts of relations, such as acoustic relations (e.g. louder, sharper), photometric relations (e.g. brighter, greener), or even abstract geometric relations may also be defined. A relation-observation processes may be defined as transformation on a set of entities that outputs relations based on their properties. This transformation may generate asynchronous symbolic messages that can serve as asynchronous events.

5. Composing Process Federations.

We propose to dynamically compose federations of processes to observe the situations that make up a context using a hierarchy of supervisory meta-controllers. Each supervisory meta-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.

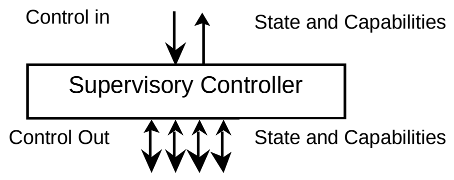

Fig. 6. A supervisory meta-controller uses observational process to associate entities with roles and to determine the relations between entities.

An example of a process federation is a federation for tracking faces, illustrated in figure 7. 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 meta-controller to detect and initiate tracking of the user's face.

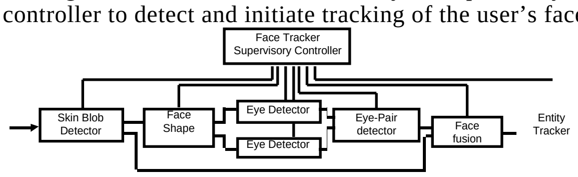

Fig. 7. A federation of processes to track faces.

If the region's properties pass the acceptance test, then observational processes for detecting eyes may be applied within the ROI defined for the face. Detected eye entities may be 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 7.

6. Context and Situation.

We propose to use a model of context to direct the assembly of process federations. Early researchers in both artificial intelligence and computer vision recognized the importance of context. The "Scripts" representation [13] sought to provide just such information for understanding stories. Minsky's Frames [14] sought to provide the default information for transforming an image of a scene into a linguistic description. Semantic Networks [15] sought to provide a similar foundation for natural language understanding. All of these were examples of what might be called "schema" [16].

In computer vision, the tradition of using context to provide a framework for meaning paralleled and drew from theories in artificial intelligence. A central component of the "Visions System" [17] 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" [18] and "Condor" [19] floundered on problems of unreliable image description and computational complexity.

Winograd [20] 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.

We define context as a composition of situations relative to a task. The situation within context 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.

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

A role is potential for action within a task. The actions of a role may be enabled by certain entities. When an entity enables the actions of a role, it is said to be able to "play" the role. An entity is judged to be capable of playing a role if it passes an acceptance test based 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.

The set of entities that can provide a role may be open ended. In the system's context model, entities are assigned to roles when they pass an acceptance test. Such assignment is provided by a process that applies a predicate function defined over entities and their properties.

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

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. The assignment of entities to roles may (often will) change dynamically. Such changes provide the basis for an important class of events.

A 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 scene with respect to the 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. Situation events may be contrasted with context events that do require a change to the federation.

In order to compose processes, the higher level supervisors recruit observational processes to form local federations. These federations determine and track the entities that may play roles in the users context, determines the assignment of entities to roles, and determines the relations between entities. The system's 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 meta-supervisor observes the state and capabilities of observational processes to determine if they are most appropriate at the current time to provide the required function.

A crucial problem with this model is how to provide a mechanism for dynamically composing federations of supervisory meta-controllers that observe the entities and relations relative to the user's context. Our approach is to construct the meta-supervisors using a forward chaining rule based system written in JESS (CLIPS in Java). Meta-supervisors are designed for specific contexts. Meta-supervisors 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-supervisor may be seen as a form of reactive expert system. For each user context, it invokes and revokes the corresponding highest-level meta-controllers. These meta-controllers, in turn, invoke and revoke lower level controllers, down to the level of the lowest level observational processes. Meta-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 meta-controller.

7. Conclusions

In this paper we have described our current approach to building context aware systems for observing human activity. A context in this system, 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 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.

Acknowledgment

This work has been partly supported by the EC projects IST FAME (IST-2000-28323) and CAVIAR (IST-2001-37540)

References

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

[2] 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.

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

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

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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] T. Winograd, "Architecture for Context", Human Computer Interaction, Vol. 16, pp401-419.