RC 21894 (98512) 30 November 2000

Computer Science

IBM Research Report

Event Perception in Agent Mediated Computer Supported Cooperative Work

Yiming Ye, Stephen Boies
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Event Perception in Agent Mediated Computer Suported Cooperative Work

Yiming Ye and Stephen Boies
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA

ABSTRACT

With ubiquitous connectivity on the horizon, collaborative computing will be one of the major applications in the evolution of computing and communication. In this paper we present an architecture for Agent Mediated CSCW system and describe its special characteristics. We point out that a challenging research issue in Agent Mediated CSCW concerns the task of event perception by devices within a user's working environment. We propose an eigenspace approach to perform event perception. We also propose a method which constructs an eigen-pyramid to perform this task when the number of devices is huge. We present experimental results to verify these methods.

1. INTRODUCTION

The worldwide natural of today's market has forced many companies to de-centralize their organizational structures. With ubiquitous connectivity on the horizon, collaborative computing promises to become one of this new century's core applications. People will be more and more involved in Computer Supported Cooperative Work (CSCW) because of the pressure from companies to improve their product-development and decision making process and because of the convenience brought by the information super-highway.

There are four modes conceptualized by CSCW researchers on how people work [20]. Synchronous mode refers to the situation that activities occur at the same time and in the same place; distributed synchronous mode refers to the situation that activities occur at the same time but at different places; asynchronous mode refers to the situation that activities occur at different times in the same place; and distributed asynchronous mode refers to the situation that activities occur at different times and places.

The task of event perception is very important with respect to the CSCW distributed synchronous mode. Many computer systems support simultaneous interaction by more than one user. However most of them support multiuser interaction in a way that prohibits cooperation - they give each user the illusion that he or she is the only one using the system. To support and encourage cooperation, cooperative applications must allow users to be aware of the activities of others. The purpose of a cooperative multiuser interface is to establish and maintain a common context, allowing the activities or events associated with one user to be reflected on other users' screens. For example, Lotus Sametime [12] is a family of real-time collaboration products which provides instant awareness, communication, and document sharing capabilities, bringing the flexibility and efficiency of real-time communication to the business world. The cornerstone of Sametime is awareness. With awareness of coworkers, partners, or customers online, users can communicate in a variety of ways. However, a direct reflection of all the activities on other users' screen is not approachable. The first reason is that it wastes communication bandwidth especially when users are far apart and the amount of data to be transmitted, such as video data, are huge. The second reason is that many users may not like the situation that his or her activities are broadcasted to all the other members of the team. The third reason is that each user is concentrated on his or her own work and does not have the energy and motivation to monitor every movement of other users. Thus, it is critical for CSCW interface to analyze activities of a given user, detect important events occurred, and only reflect necessary events to other users. Event perception will be even more important to CSCW in the pervasive computing world, where the dominance of the traditional PC as the primary computing resource is replaced by a large collection of devices with embedded computing. These intelligent, interconnected devices will be seamlessly embedded within our offices, constantly sensing and reacting to the environment. The information provided by these pervasive devices within an office environment will be very important in CSCW applications.

It is our belief that autonomous agents are of great value to a CSCW system and a certain amount of future research on CSCW will be centered on multi-agent aspect of groupware. Intelligent agents can undertake sophisticated processes on behalf of the user. A multiagent approach to CSCW can capture the dynamics of a team work and even re-shape its form and characteristics. The automation brought by CSCW agents will dramatically reduces certain types of frictional costs during team work. Furthermore, the intelligence of a multiagent CSCW system will be able to keep the pri-

vacy of its user and the security of each user's local work.

In this paper, we will present the system architecture of an agent mediated CSCW system called Agent Buddy and study in detail its event perception issues.

2. ARCHITECTURE OF AGENT BUDDY

Software agents are studied from two complementary perspectives. The first views software agents as entities with different skills and knowledge within a larger community of agents [17]. Each agent is independent or autonomous. It may accomplish its own task or cooperate with other agents to perform a personal or global task. The second approach concentrates on the necessity for agents to interact with users at the level of the interface [13]. The critical points here are how agents can understand the needs and goals of the user, how agents should behave, and how agents' behaviors can be perceived by the user via the interface.

The Agent Buddy approach is a combination of the above two approaches. Figure 1 shows the architecture of our agent mediated CSCW system - the Agent Buddy system. The goal of the system is to perceive events associated with one user and selectively provide the perceived information to other users of the team. The Agent Buddy system can be added to any CSCW system to enhance the sense of working "together" concurrently and at the same time keep the privacy of each user. It can also be used to detect abnormal visits of other agents and acted as a security keeper. When a user wants to contact another user about a certain issue such as making a phone call or scheduling a meeting, he usually contact his agent. His agent then negotiate with the agent of the other user and tell him about the negotiation results. The user then judge the negotiation results and decides what to do.

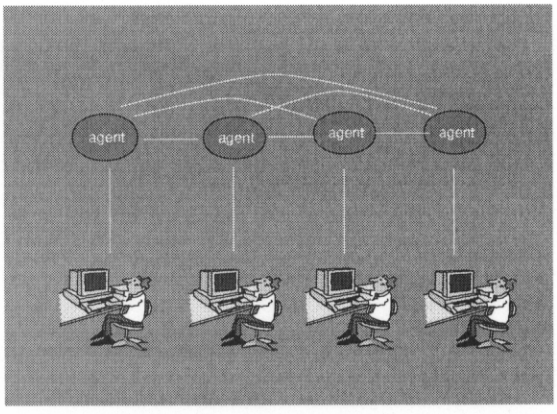


Figure 1: *The architecture of Agent Buddy*

An agent in Agent Buddy is a computational system that inhabits dynamic CSCW environments. It has knowledge about its responsible user and conventions of the working group. This knowledge can be used to guide its interactions with its responsible user and other agents of the group. The goal is to make CSCW easier and more efficient for members of the working group. Figure 2 shows modules within an agent. The User Interface Module is responsible for obtaining input from the user and input from various devices.

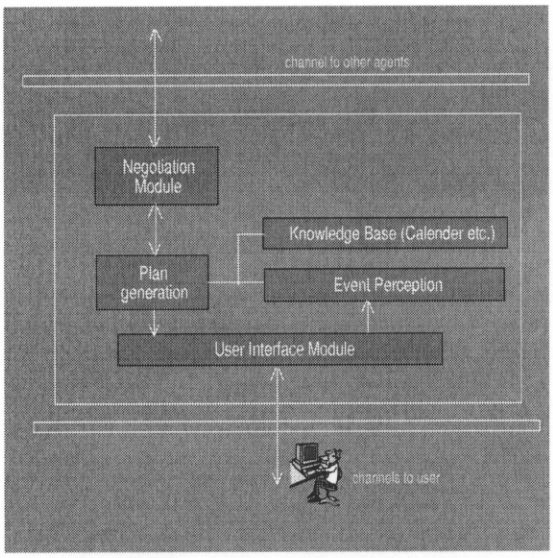


Figure 2: *The architecture of Agent Buddy*

It is also responsible for performing dialogue between the user and the agent, such as delivering the negotiation results to the user. The Event Perception Module analyzes input from various devices within a user's working environment and detects events. The Knowledge Base Module contains the knowledge of the agent on the user and the group. These knowledge include appointment schedules of the user, preferences of the user, and the relative importance of different group members, etc.. The Plan Generation module generates action plans for the user by combining the user's request, the knowledge from Knowledge Base, the events detected by Event Perception Module, and the requests of agents of other users delivered by the Negotiation Module. The Negotiation module is used to perform communications and negotiation dialogues with agents of other users. XML [4] is used to encode messages between system components, agents, and users.

In the following, we describe special features of our system.

- Each Agent Buddy can perceive events with respect to its user. This is the most important feature of our system. We will discuss this feature in detail in following sections.
- The User Interface Module allows multi-model user input. The input can come from different sources such as the keyboard, the video camera, the graphical user interface, and various devices of the office environment etc.. The input channels will be even more rich when pervasive computing becomes a reality.
- Multiagent negotiation is used to reduce the time and energy fraction between different users of the group. Our system supports the activity of negotiation among agents. The goal is to transform negotiation activities among users into negotiation activities among agents. This will greatly reduce the fraction of a group work and increase efficiency. Usually, group work involves

conflicts at various degrees. Users need to negotiate among each other to form a solution that is satisfiable by all parties. In a high conflict situation, rich interpersonal information is required in addition to information concerning the task [15]. When this kind of high conflict situation happens, agents for each user should automatically detect the situation and automatically setup full multimedia conferencing links among participant users. This functionality can greatly increase the efficiency of the group work. When a low conflict situation happens, agents can negotiate a solution automatically for users. This can dramatically reduce fractions in group work. Appointment scheduling is one of the examples. When one user is trying to schedule a meeting with another user. The user can simply ask his agent to contact the agent of the other user and negotiate a time. This back and forth negotiation process for agents can be complex because a lot of background information might be considered. Another example is phone call scheduling. For example, a user wants to make a phone call to his manager. It is not appropriate for him to simply pick up the phone and make the call. What he can do is to transfer his requests to his agent. His agent then contact the agent of his manager. The other agent then starts event perception process and detects the manager's state. If it concludes that it is ok to have a phone conversation based on the results of event perception and background knowledge, it will inform the user's agent about the decision. The user's agent then inform the user to make the call.

- Our system can create a sense of group work and at the same time keep the privacy and maintain the security of each user. To create a sense of group work, each agent has an interface to display events associated with other users. Events associated with a user can be whether the user is logged on, how frequently the user is typing on the keyboard, what program the user is running, whether the user is entertaining himself by browsing the Internet or is working on the project, whether the user is on the phone and who he is talking to, whether the user is happy, sad or simply normal, whether the user has a visitor, and even whether the user needs a break because he is not efficient at all, etc.. However, an agent is not able to display all events of other users. There are two reasons. The first and the most important reason is that the agent must communicate with agents of other users for the permission to access events detected by those agents. Agents of other users will decide whether the status of an event can be accessed by the asking agent. Only events that does not intrude privacy can be accessed. For different asking agents, the criteria will be difficult. Similarly, an user can not access and modify the documents of other users when there are potential security concerns. The second reason is that an agent should not display all the events of other users because it is usually not necessary and impossible to display everything within a single screen. An agent must intelligently select events to display for the most benefit of its user.

- XML is used to encode messages among agents, messages among different components of an agent, and messages transfered from devices to the Event Perception Module of an agent. XML is a descendant of the Standard Generalized Markup Language or SGML. It allows developers to create their own markup languages, the semantics of which enable specific applications. Using XML for messaging facilitates the development of common data abstractions leading to more modularity and sharing of data. XML simplifies transactions and negotiation processes in the agent mediated CSCW environment. If the message sent to an agent is structured with XML, it is much easier for the agent to understand exactly what the data means and how it relates to other piece of data it may already know. The messages in our system is interpreted using Document Object Model (DOM). In Agent Buddy, IBM DOM parser [11] is used.

  The following is a simple example on encoded messages sent from the keyboard device to the Event Perception Module. It tells the agent that the user typed character "a" at 1:05 PM. They keyboard belongs to machine orasis.watson.ibm.com. The machine is a ThinkPad 770X with TCPIP address "9.2.11.71".

```
⟨?xml version="1.0"?⟩
  ⟨Signal-From-Device-To-Agent⟩
    ⟨Signal-Source⟩
      ⟨Associated-Machine⟩
        TYPE="ThinkPad 770X"
        NAME="Orasis.watson.ibm.com"
      ⟨/Associated-Machine⟩
      ⟨Associated-TCPIP⟩
        9.2.11.71
      ⟨/Associated-TCPIP⟩
      ⟨Device⟩ Keyboard ⟨/Device⟩
    ⟨/Signal-Source⟩
    ⟨Content EventAction="Push-Down-Release"
        EventKey="a"
    ⟨/Content⟩
    ⟨Event-Time DAY="05" MONTH="11"
      TIME="01:05 PM"
    ⟨/Event-Time⟩
  ⟨/Signal-From-Device-To-Agent⟩
```

It is very important to evaluate the performance of different agent mediated CSCW systems. Here we propose the CSCW version of the Turing Test.

Definition: *Suppose that there are two CSCW group working environments available. One is Agent Mediated CSCW environment. The other is Human Mediated CSCW environment with human beings working as negotiators, event watchers, and secretary for a group of users. Each user is assigned one person for these purposes.*

If the group of users have been working under one of the environment for a time that is sufficiently long but could not tell whether the environment is Human Mediated CSCW or Agent Mediated CSCW, then we say that the given Agent Mediated CSCW have passed the CSCW version of Turing Test. Otherwise, we say that it failed the test.

We call the average number of transactions involved in the group work before the system failed the test the characteristic transaction number. This number can be used to compare two Agent Mediated CSCW systems. □

3. EVENT PERCEPTION

3.1 Previous Work

A challenging issue in Agent Mediated CSCW is event perception. The term "event" is widely used yet has no specific definition. It provides a useful categorization for describing everyday happenings, allowing the continuity of everyday experience to be cut up into discrete bounded temporal units. Schank and Abelson [24] compose events into scripts to describe typical behavior of a customer at a restaurant such as entering, going to a table, order, eating and paying. Nagel [19] provides an ontology for describing the behavior of agents. Badler [1] and Tsotsos [26] extract natural language description that captures the activity of the moving objects present in a sequence of images. Reynolds [23] recounts issues found when combining categorization schemes developed by different observers to describe the behavior of Rhesus Monkeys. He concludes that the flexibility of language may not be needed to describe perceptual events which are really pre-linguistic. The purpose of event representation is for event perception. Experimental psychology has been used to address the problem of how causal relationships are perceived by human. For example, Michotte [16] investigates the effect of varying the temporal interval between action and reaction. His experiments employ the movements of simulated blocks, which bring to light properties of perceptual control that are not that illustrative in terms of everyday events. There are also engineering approaches to study event perception. Mann, Jepson, and Siskind [14] present an implemented computational theory that derives qualitative scene dynamic descriptions directly from camera input. Their approach is based on an analysis of the Newtonian mechanics of a simplified scene model. Interpretations are expressed in terms of assertions about the kinematic and dynamic properties of the scene. A preference hierarchy is used to select plausible interpretations from multiple feasible solutions. solutions are compared using a preference hierarchy

3.2 Event Perception in Agent Buddy

Event perception in Agent Buddy is unique in the sense that events are perceived by a society of devices within a users working environment. Each device only sense the environment from a very specific angle. Attempting to perceive events using only one device is sometimes awkward and computationally intensive. However, the collective power of event perception is strong because of the varieties of aspects of dynamic environment that can be sensed by interconnected complementary devices. This observation can be easily illustrated by the famous parable "The Blind Men and the Elephant". Although each blind man can only feel a part of the elephant. Their collective observations can give true shape of an elephant if they communicate.

Suppose within the environment there are M devices. Suppose $\mathbf{r}_1(t)$, ..., $\mathbf{r}_M(t)$ are readings sensed by these devices at time instant t, where $t \in [0, T]$. Suppose E_1, ..., E_N are different events in consideration. Then the task of event perception is to analyze the sensing data $\mathbf{S}(t) = \langle \mathbf{r}_1(t), \ldots, \mathbf{r}_M(t) \rangle$ and to detect whether events E_i $(1 \leq i \leq N)$ have been occurred.

3.3 Single Device Event Perception

When the relationship between the event to be perceived and the readings from a specific device is close, event perception can be done by only using this device. We illustrate this by using an intelligent screen saver [31]. The idea of the intelligent screen saver is to use a camera to provide a non-intrusive way to automatically control the status of the screen saver of a computer. The event to be perceived is whether there is a human before a computer. The purpose is to use this event to trigger the status of the screen saver. Traditional screen saver uses a fixed time constant to turn on/off the screen. If there is no touches on the keyboard or no movement of the mouse within a certain amount of time, then the screen saver status will be on (screen is turned off). This is not convenient because the user has to touch the keyboard or mouse in order to bring the screen back even though he/she is still looking at the screen. Similarly when the user leaves his computer the screen saver mode is still off, it will turns on only after a certain amount of time. This is a waste of power because the screen is not been used. The intelligent screen saver that we have implemented can detect whether a user is before the computer. If the user is before the computer, the screen saver mode will be off, otherwise the screen saver mode will be on. The event that a user is before the computer is detected by analyzing the image sequences from the camera. The idea is to keep calculating the pixel difference between every consecutive images. If significant differences occurs between almost every consecutive images within a certain amount of time, then we believe that a user is before the computer. Otherwise, we believe that there is no user before the computer. We patented this technology [31] and demonstrated it at various industry conferences.

It is easy to use a camera to detect whether a user is before a computer. However, it may not be so easy to use a camera to detect other events, such as the identity of the user or what the user is doing. In general, a single device is not able a cover all the events. A good event perception requires a society of complementary devices.

4. EIGENSPACE EVENT PERCEPTION

We propose to use eigenspace method to perform the task of multi-device event perception in Agent Buddy. When we begin to consider this task, we must incorporate the underline probability distribution of events and the readings from the measuring devices. Eigenspace methods such as principal component analysis (PCA) are particularly well-suited to such a task since they provide a compact and parametric description of the readings of the devices and also automatically identify the essential components of the underlying statistical variability. An observed event can be modeled using device readings $\mathbf{S}(t)$ at discrete time instants. A reduced dimensionality model of events can be constructed using PCA of example device readings. Recognition of events is then posed as matching between the principal component representation of the observed activity to these learned models.

4.1 Event Modeling

To model events, we collect $N_{examplar}$ exemplars for each event. Each exemplar collects readings from devices within a time interval $[0,T]$. These readings are discretized into $N_{readings}+1$ readings at time instant 0, $\frac{1}{N_{readings}}T$, ..., $\frac{N_{readings}-1}{N_{readings}}T$, T. For device h $(0 \leq h \leq M)$, we denote its kth $(0 \leq k \leq N_{readings})$ readings for the jth $(1 \leq j \leq N_{examplar})$ exemplar from the ith $(1 \leq i \leq N)$ event as $r_h^{i,j}(k)$. Let $[r^{i,j}(k)] = (r_1^{i,j}(k), \ldots, r_M^{i,j}(k))^T$ be a column vector of kth readings of all the devices for the jth exemplar from the ith event. Let $[r^{i,j}]$ represent the column vector obtained by simply concatenating the $[r^{i,j}(k)]$ for all the k readings, $[r^{i,j}] = (r_1^{i,j}(0), \ldots, r_M^{i,j}(0), r_1^{i,j}(1), \ldots, r_M^{i,j}(1), \cdots, r_1^{i,j}(N_{readings}), \ldots, r_M^{i,j}(N_{readings})^T)$. Here, $[r^{i,j}]$ gives the readings for the jth exemplar of the ith event. This is the readings of all the devices with respect to an exemplar in the model training phase. The length of vector $[r^{i,j}]$ is $M \times (N_{readings}+1)$. Then the sampling readings matrix $\mathbf{A}$ for all the events and their associated exemplars can be created by the set of all j and i of $[r^{i,j}]$. $\mathbf{A} = ([r^{1,1}], \ldots, [r^{1,N_{examplar}}], \cdots, [r^{N,1}], \ldots, [r^{N,N_{examplar}}])$. The dimension of matrix $\mathbf{A}$ is $(N \times N_{examplar})$ columns and $(M \times (N_{readings}+1))$ rows. The $N \times N_{examplar}$ columns of $\mathbf{A}$ give readings for all the exemplars of all the events. This is the total number of training set of data. The $M \times (N_{readings}+1)$ elements of each column gives the readings from all the devices at all the discrete time instant for an exemplar. Thus, each column of $\mathbf{A}$ refers to a given training set, the elements of the column refers to the readings for this set. Usually $M \times (N_{readings}+1) \gg N \times N_{examplar}$.

Matrix $\mathbf{A}$ can be decomposed using singular value decomposition (SVD) [28] [22] as:

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T.$$

Where $\mathbf{U} = (U_1, \ldots, U_{N \times N_{examplar}})$ is an orthogonal matrix of the same size as $\mathbf{A}$ representing the principal component directions U_i $(1 \leq i \leq N \times N_{examplar})$ in the training set. These are best directions that can clearly distinguish the training data. $\mathbf{W}$ is a diagonal matrix with singular values $\lambda_1, \ldots \lambda_{N \times N_{examplar}}$, sorted in decreasing order along the diagonal. The virtual of these values is that they rank the dimensions of the space in terms of variations along the principal component directions, and that this ranking is very often related to their importance. $\mathbf{V}^T$ is a $(N \times N_{examplar}) \times (N \times N_{examplar})$ matrix that encodes the coefficients to be used in expanding each column of $\mathbf{A}$ in terms of principal component directions.

The readings from the jth exemplar of the ith event $[r^{i,j}]$ can be approximated according to the largest q $(q \leq N \times N_{examplar}$ singular values $\lambda_1, \ldots, \lambda_q$ as:

$$[r^{i,j}] \approx \sum_{l=1}^{q} c_l^{i,j} U_l,$$

where $c_l^{i,j}$ are scalar values that can be calculated by taking the dot product of $[r^{i,j}]$ and U_l, $c_l^{i,j} = [r^{i,j}]^T U_l$. This is the process of projecting the reading vector $[r^{i,j}]$ onto the subspace spanned by the q basis vectors $U_1, \ldots, U_q$. It can also be viewed as a parameterization of $[r^{i,j}]$ in terms of $U_1, \ldots, U_q$ with parameters $c_1^{i,j}, \ldots, c_q^{i,j}$. Thus, for a given i and j, we can obtain a vector $C^{i,j} = (c_1^{i,j}, \ldots, c_q^{i,j})^T$ that gives the coefficients of the corresponding readings. For all the possible i and j, we can get a coefficient matrix $\mathbf{C} = (C^{1,1}, \ldots, C^{1,N_{examplar}}, \cdots, C^{N,1}, \ldots, C^{N,examplar})$.

Now we transform $\mathbf{C}$ into a matrix that represents the average coefficient for each event. For any event i, matrix $\mathbf{C}$ contains the coefficient vectors of all its exemplars: $(C^{i,1}, \ldots, C^{i,N_{examplar}})$. The average coefficient vector for these exemplar vectors can be calculated by: $\overline{C}^i = \frac{\sum_{j=1}^{N_{examplar}} C^{i,j}}{N_{examplar}}$. The average coefficient matrix becomes: $\overline{\mathbf{C}} = (\overline{C}^1, \ldots, \overline{C}^N)$. Each column i of Matrix $\overline{\mathbf{C}}$ corresponds to the average coefficient vector $\overline{C}^i = (\overline{c}_1^i, \ldots, \overline{c}_q^i$ of event i. $\overline{\mathbf{C}}$ is the model of events learned from the training phrase and will be used in event perception.

4.2 Event Perception

The perception of events involves matching readings from all the devices in a real situation against learned models of all the events. For the same event, readings in real application may differ from those of its exemplars because of various reasons such as noise etc.. However, they may share some commonalities or signatures. The use of eigen space approach for event perception assumes that this commonalities or signatures of a given event is captured by the coefficients of the readings along principal component directions.

Suppose $\mathbf{R}(t) = (R_1(t), \ldots, R_M(t))^T$ is the readings from the M devices within time period $[0,T]$. We discrete $[\mathbf{R}(t)]$ into $N_{readings}+1$ readings at time instant 0, $\frac{1}{N_{readings}}$, ..., $\frac{N_{readings}-1}{N_{readings}}$. Let $\mathbf{R}(k)]$ denote the kth readings of all the devices at time instant k. By concatenating readings from all the time instant, we obtain the column vector $\mathbf{R}$ which gives readings of all the devices at all the time instant. By projecting this vector on the principal component directions we recover a vector of coefficients, $\vec{c} = (c_1, \ldots, c_q)$, that approximate the event to be perceived as a linear combination of eigen event basis. Upon the recovery of the real situation coefficient vector, the normalized distance Δ_i between $\vec{c}$ and model coefficients $\overline{C}^i$ is used to perceive the observed event. Here

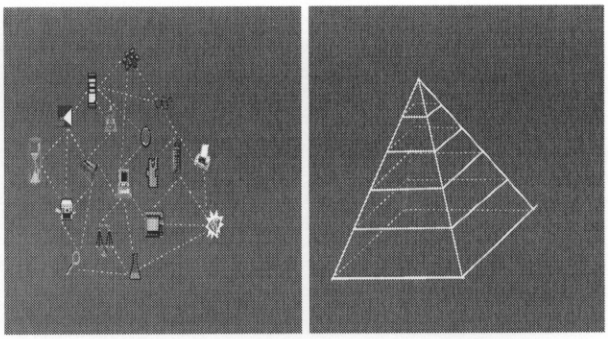
$$\Delta_i = \sum_{k=1}^{q} (c_k - \overline{c}_k^i)^2$$

The event i with the smallest distance Δ_i is considered the best match of the observed event.

4.3 Eigen Pyramid

We can imaging that in the inter-connected pervasive world, huge amount of devices will be involved. Events like what a person is doing can be perceived by considering only devices within an office. However, events like whether people within a building are having a meeting should be considered with all the devices within the building. In general, the bigger the scale of the events to be perceived, the more devices need to be considered. When the number of devices exceeds

a certain threshold, the strategy described above will not work because too much computational time is needed.



(a) (b)

Figure 3: *Illustration of the eigen pyramid event perception. (a) The first level input data come directly from the raw readings of all the devices. (b) The eigen pyramid constructed to perform event perception when huge amount of devices are involved.*

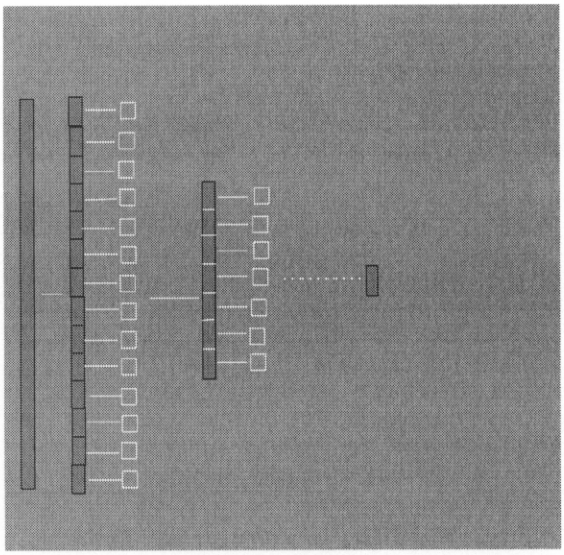


Figure 4: *Detailed graphical illustration of the eigen pyramid construction process.*

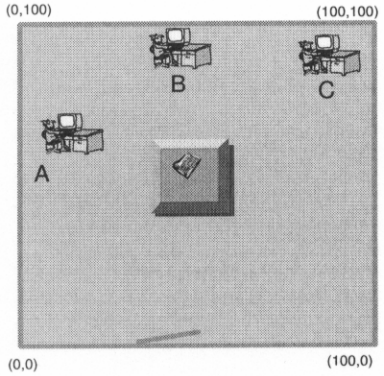
In order to perform event perception when the number of devices is huge, we propose a new strategy called "pyramid eigen space". Suppose $n_{acceptable}$ is the number of devices that can be handled by the above eigen space approach. Suppose N_{total} is the total number of devices to be considered. Our strategy is to first divide uniformally these N_{total} devices into different groups such that each group can be handled by the above eigen space method. Suppose $N_{total} = k(n_{acceptable} - 1) + u$, where $0 \leq u < n_{acceptable} - 1$. If $u = 0$, then we can divide the devices into $n_{acceptable} - 1$ groups. Otherwise, we can divide the first $k(n_{acceptable} - 1)$ devices into k groups with each group has $n_{acceptable} - 1$ members. Then we distribute the rest r devices into the first r groups obtained. Thus, we have divided devices into k groups, with each group has either $n_{acceptable} - 1$ or $n_{acceptable}$ members. For each group, we run the training data and detect their principal directions. Now, we collect the coefficient vector with respect to the principal directions for each training exemplar. The length of the coefficient vector is $N_{examplar} \times N_{events}$. Since the coefficients captures the differences in the training data, we will take coefficients of each exemplar as the input to the second level of the pyramid.

At the first stage, the length for each training vector (exemplar) is N_{total}. During the process above, they are divided into k groups. Each group generates $N_{examplar} \times N_{events}$ coefficients. Thus, the total length for the second level of input will be $k \times N_{examplar} \times N_{events}$, which is much less than $N_{total} = k(n_{acceptable} - 1) + r$. If $k \times N_{examplar} \times N_{events} > n_{acceptable}$, then we take this new data as input, and repeat the above data abstraction process to further reduce the amount of the data. Otherwise, we have reached a stage that a decision can be made. If $k \times N_{examplar} \times N_{events}$ is much less than $n_{acceptable}$ and a further eigen coefficient extraction is meaningless, then we take this $k \times N_{examplar} \times N_{events}$ numbers as the final coefficients of the training exemplar. If $k \times N_{examplar} \times N_{events}$ is big enough such that another round of eigen coefficient extraction is meaningful, then we run the PCA process to extract the new coefficients. These new coefficients will be taken as the final coefficients of the exemplar of the training events. After extracting the final coefficients of all the exemplars of all the events, the average of the final coefficients of all the exemplars with respect to a given event are taken as the model of the corresponding event.

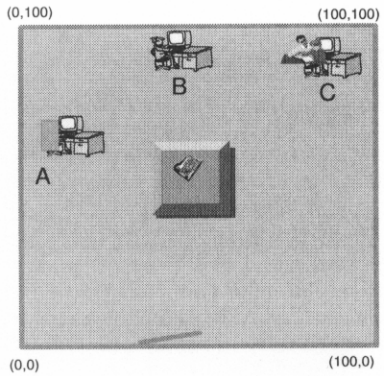
During the event perception phase, after the readings from all devices are available, we first extract coefficients with respect to the principal directions of the first layer of the pyramid formed during the training phase. Then take these coefficients as the input to the second layer of the eigen pyramid and extract corresponding coefficients. This process is continued until the top level of the eigen pyramid has been considered. The resulting final coefficients are the eigen pyramid signatures extracted from the raw device readings through the eigen pyramid. At last, we compare this final coefficient vector with those of the training models of all the events and select the closest one as the event perceived.

5. EXPERIMENTS

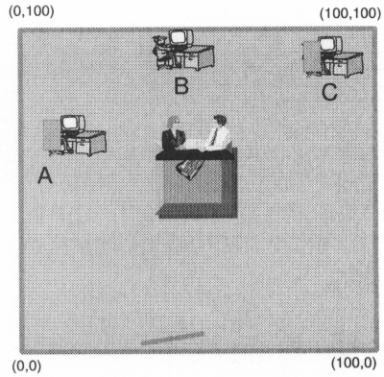
Simulation experiments have been performed to test the effectiveness of the eigenspace event perception approach. The environment is assumed to be an office with size 100×100 as shown in Figure 5. Three person a, b, and c share the office. They use tables A, B, and C respectively. There is a meeting table in the middle of the office where they can have a discussion there. There are 7 events to be perceived in our experiments: (1) a, b, c are working at their desks respectively; (2) a and b are discussion, either at A or at B, while c is working at Table C; (3) a and b are having a meeting at the meeting table, while c is working at C; (4) a and c are discussion, either at Table A or at Table C, while b is working at table B; (5) a and c are having a meeting at the meeting table, while b is at B; (6) b and c are discussion either at Table C or Table B, while a is working at table A; (7) b and c are having a meeting at the meeting table, while A is working at A.



(a)



(b)



(c)

Figure 5: *The environment for simulation test. The size of the environment is assumed to be* 100×100. *The centers of Table A, B, and C are at* $(10, 70)$, $(50, 90)$, $(90, 90)$, *respectively. The centers of the round table is at* $(50, 50)$. *The weight of a, b, and c are* 75, 100 *and* 150 *respectively. (a) The situation for the first event. (b) The situation for the* 4*th event. (c) The situation for the* 5*th event.*

To perceive events, we assume that there are two kinds of sensors in use. One is a camera that takes images of the environment and returns the average image difference values within a given time period. The image difference value between two consecutive images is calculated by counting the total number of pixels whose (r, g, b) values have a big disparity between the two images. The other is weighting device that gives the weight sensed. The image difference values generated by the camera are assumed to be the following: if a person is working at his own table, a difference value 75 is generated; if he is involved in a discussion, a value of 115 is returned; if he is at a meeting at the center table, a value of 175 is generated. We divide the environment into 100×100 grids. Each grid is equipped with a weighting device. The weight sensed by a weighting device is the sum of weights caused by all the persons within the environment. The weight caused by a person with weight W and distance r to the device is given by:

$$W \exp(\frac{r^2}{2\sigma})$$

where σ gives the sensitivity of the weighting device with respect to its distance to the person.

In the training process, for each event, we collect 5 exemplars. For each exemplar, we collect weighting data at 7 time instant. Each time instant has 100×100 weight readings corresponding to grids in the environment. The first element of the exemplar vector $r^{i,j}$ is the image difference readings. It is followed by weight readings at the 1st, 2nd, 3rd, 4th, 5th, 6th, and 7th time instant. The length of the column vector is $1 + 7 \times 100 \times 100$. Each corresponds to readings in one training exemplars. There are totally 7×5 training column vectors. They form the training matrix.

To form the training data, we give the base positions for Table A as $(50, 90)$, Table B as $(10, 70)$, Table C as $(90, 90)$, and round table as $(50, 50)$. The position of a person at kth time instant of the jth exemplars of ith event is given by the base position of the Table the person is close with respect to the ith event plus a random variable position $(\epsilon^x_{i,j,k}, \epsilon^y_{i,j,k})$. For example, in our test, $(1, 9)$, $(0, 5)$, $(6, 6)$, $(9, 1)$, $(4, 4)$, $(5, 5)$, $(7, 7)$ are the variances for the 1st exemplar of the first event.

Here is the Base position for persons at the 7 events:

	1	2	3	4
a	$(10, 70)$	$(50, 90)$ or $(10, 70)$	$(50, 50)$	$(10, 70)$ or $(90, 90)$
b	$(50, 90)$	$(50, 90)$ or $(10, 70)$	$(50, 50)$	$(50, 90)$
c	$(90, 90)$	$(90, 90)$	$(90, 90)$	$(10, 70)$ or $(90, 90)$

	5	6	7	
a	$(50, 50)$	$(10, 70)$	$(10, 70)$	
b	$(50, 90)$	$(50, 90)$ or $(90, 90)$	$(50, 50)$	
c	$(50, 50)$	$(50, 90)$ or $(90, 90)$	$(50, 50)$	

Table 1

After training, the eigen values are shown in Figure 6. From the figure, we can see that the first 9 eigen values are big compared to the rest of eigen values. Among them, the first 5 eigen values are the most meaningful eigen values.

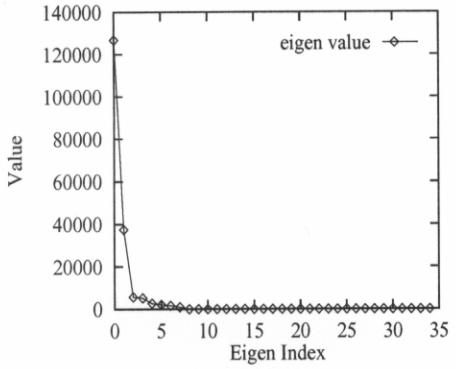


Figure 6: *eigen values*

Figure 7 shows the ratio of captured information, $\frac{\sum_{i=1}^{p}\sqrt{\lambda_i}}{\sum_{i=1}^{q}\sqrt{\lambda_i}}$, as a function of the number of principal components used in the event detection. We can see that the first 9 eigen directions captured almost all the information. The rest almost contributed nothing in terms of capturing information.

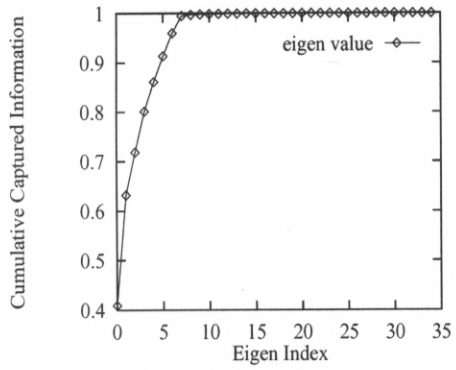


Figure 7: *Captured information*

Figure 8 shows one of the simulation results for a tester of event 1. Figure 8(a) shows the normalized distances between the tester and the models of events 1, ..., 7 obtained during the training phase. In other words, it shows Δ_1, Δ_2, Δ_3, Δ_4, Δ_5, Δ_6, and Δ_7, as a function of eigen indexes. We can see that distances between the tester and events 4 and 6 are much bigger than distances between the tester and events 1, 2, 3 5, and 7. We can also notice that distances between the tester and all the event models becomes almost unchanged for eigen indexes after 9. This is because that little information is captured by those principal directions associated with eigen values whose indexes are bigger than 9. We can also notice that the first 5 principal directions capture a great potion of information. Figure 8(b) shows in detail the distances of the tester and event models 1, 2, 3, 5, and 7. We can notice that the distance between the tester and event model 1 is smaller than the distances between the tester and event models 2, 3, 5, and 7 even at a very early stage. This fact illustrates the power of the eigen space event perception approach in the pervasive world. Figure 8(c) gives a detailed view of the difference of distances of the tester and event models 2, 5, and 7. It shows $\Delta_2 - \Delta_7$ and $\Delta_5 - \Delta_7$ as a function of eigen indexes.

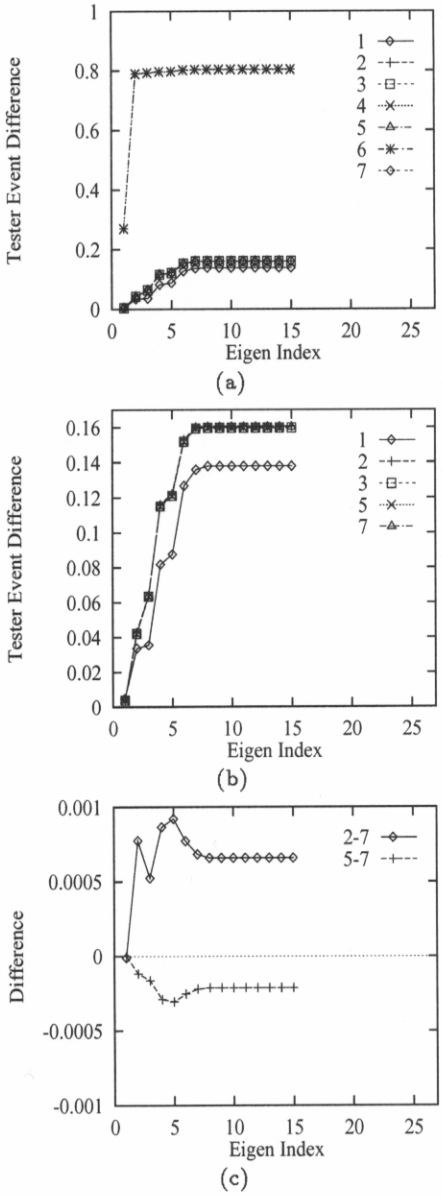


Figure 8: The testing result when a tester of event 1 is used. (a) The distance between the tester and event models obtained during the training phrase. (b) The distances between the tester and event models 1,2,3,5,7. (c) The difference $\Delta_2 - \Delta_7$ and $\Delta_5 - \Delta_7$.

Figure 9 shows another simulation result when a tester of event 6 is used. The normalized distances between the tester and the models of events 1, ..., 7, are shown. We can see that distances between the tester and events 4 and 6 are much smaller than distances between the tester and events 1, 2, 3 5, and 7. We can also notice that event 4 is more closer to the tester at beginning. However, this situation changes as the eigen index becomes bigger. This illustrates that in order to perceive events that are closely related, enough principal directions that can capture a great portion of in-

formations should be considered.

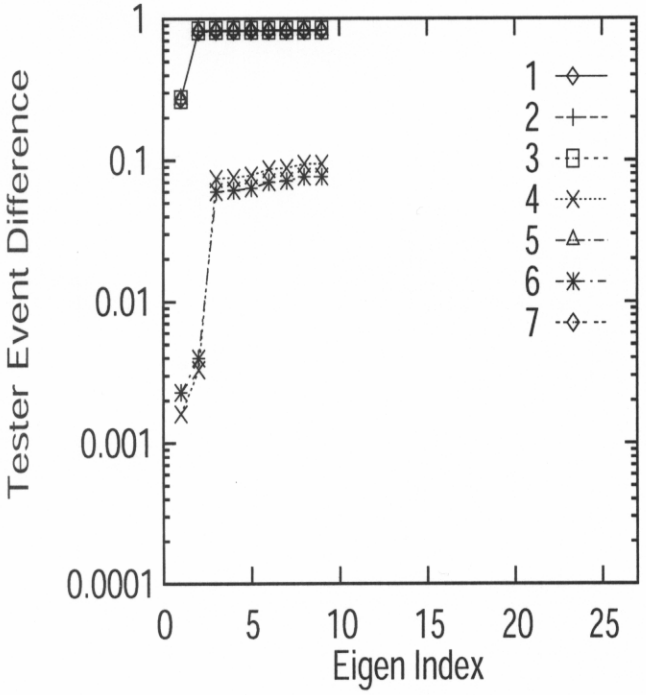


Figure 9: The testing result when a tester of event 6 is used.

6. CONCLUSION

In this paper, we described an Agent Mediated CSCW system - the Agent Buddy System. Agent Buddy is a CSCW system that can create a sense of group work and at the same time keep the privacy and maintain the security of each user. Multiagent negotiation process is used in the system to reduce fractions among group members during the geographically distributed team work. XML is used in the system to encode communication messages. To evaluate the performance of different Agent Mediated CSCW systems, we define the Agent Mediated CSCW version of Turing Test and suggest to use this test to compare different systems.

We point out that event perception is a very important task in Agent Mediated CSCW. We propose a method that uses an eigen space to perform the event perception task. For the case where the number of devices is huge, we propose a way of constructing an eigen pyramid which can be used to discriminate different events. We conduct experiments to test our method.

7. REFERENCES

[1] N. Badler. *Temporal Scene Analysis: Conceptual Descriptions of Object Movements.* PhD thesis, Department of Computing Science, University of Toronto, Toronto, Canada, 1975.

[2] R. Bajcsy. Active perception vs. passive perception. In *Third IEEE Workshop on Vision*, pages 55–59, Bellaire, 1985.

[3] R. Bajcsy. Perception with feedback. In *Image Understanding Workshop*, pages 279–288, 1988.

[4] T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible markup language (xml) 1.0. Technical Report http://www.w3.org/TR/REC-xml, World Wide Web Consortium Recommendations.

[5] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, (47):139–160, 1991.

[6] Connel. *An Artificial Creature.* PhD thesis, AI Lab, MIT, 1989.

[7] I. Ferguson. *Touring Machine: An Architecture for Dynamic, Rational, Mobile Agents.* PhD thesis, University of Cambridge, UK, 1992.

[8] T. D. Garvey. Perceptual strategies for purposive vision. Technical Report Technical Note 117, SRI International, 1976.

[9] M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, WA, 1987.

[10] R. Howarth. Interpreting a dynamic and uncertain world: High-level vision. *Artificial Intelligence Review*, 9(9):37–63, 1995.

[11] http://www.ibm.com/xml. Ibm dom parser. Technical report.

[12] http://www.lotus.com/home.nsf/welcome/sametime.

[13] B. Laurel. Interface agents: metaphors with character. In *The Art of human-computer interface design*, pages 355–365, Addison-Wesley Reading, MA, 1990.

[14] R. Mann, A. Jepson, and J. Siskind. The computational perception of scene dynamics. *Computer Vision and Image Understanding*, 65(2):113–128, February 1997.

[15] J. McGrath. *Groups, Interaction and Performance.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[16] A. Michotte. *The Perception of Causality.* Methuen Co. Ltd, London, 1963.

[17] M. Minsky. *The Society of Mind.* Heinemann, London, 1987.

[18] J. Muller and M. Pischel. Modelling interacting agents in dynamic environments. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 709–713, Amsterdam, The Netherland, 1994.

[19] H.-H. Nagel. From image sequences towards conceptual description. *Image and Vision Computing*, 6(2):59–74, 1988.

[20] J. D. Palmer and A. Fields. Computer supported cooperative work. *IEEE Computer*, May 1994.

[21] S. Park, E. Durfee, and W. Birmingham. An adaptive agent bidding strategy based on stochastic modeling. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-99)*, pages 147–153, 1999.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numbrical Recipes in C, Second Edition*. Cambridge University Press, United Kingdom, 1999.

[23] V. Reynolds. The origins of a behavioural vocabulary. *Journal for the Theory of Social Behaviour*, 6:105–142, 1976.

[24] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, 1977.

[25] S. Sen, S. Roychowdhury, and N. Arora. Effects of local information on group behavior. In *Proceedings of Second International Conference on Multi-Agent Systems*, pages 315–321, Kyoto, Japan, 1996.

[26] J. K. Tsotsos. Representational axes and temporal cooperative processes. *In Arbib, Hanson, Allen eds. Vision, Brain and Cooperative Computation*, pages 361–417, 1987.

[27] T. Tyrrell. *Computational Mechanisms for Action Selection*. PhD thesis, Center for Cognitive Science, University of Edinburgh, England, 1993.

[28] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley and Sons, Inc, USA, 1991.

[29] M. Wooldridge and N. Jenning. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[30] Y. Ye and S. Boies. Event perception in pervasive world. Technical report, IBM Research Technical Report, Yorktown Heights, NY, USA, 2000.

[31] Y. Ye, T. Huang, and A. Senior. *Intelligent Screen Saver Using Image Difference*. USA Patent, 1999.