EasyLiving: Technologies for Intelligent Environments

Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, Steven Shafer
{barry, brianme, jckrumm, amandak, stevensh}@microsoft.com

The EasyLiving project is concerned with development of an architecture and technologies for intelligent environments which allow the dynamic aggregation of diverse I/O devices into a single coherent user experience. Components of such a system include middleware (to facilitate distributed computing), world modelling (to provide location-based context), perception (to collect information about world state), and service description (to support decomposition of device control, internal logic, and user interface). This paper describes the current research in each of these areas, highlighting some common requirements for any intelligent environment.

1. Introduction

The EasyLiving project[17] at Microsoft Research is concerned with the development of an architecture and technologies for intelligent environments. An intelligent environment is a space that contains myriad devices that work together to provide users access to information and services. These devices may be stationary, as with acoustic speakers or ceiling lights, or they may be mobile, as with laptop computers or mobile telephones. While the traditional notion of a PC is a part of this vision, a broader goal is to allow typical PC-focused activities to move off of a fixed desktop and into the environment as a whole.

An intelligent environment is likely to contain many different types of devices. First, there are traditional input devices such as mice or keyboards and traditional output devices such as speakers or displays. To support richer interactions with the user, the system must have a deeper understanding of the physical space from both a sensory (input) and control (output) perspective. For example, it might be desirable for the system to provide light for the user as he moves around at night; to enable this, the system uses some form of perception to track the user, and must be able to control all of the different light sources. Input devices can include things such as active badge systems[20][21], cameras [3][17], wall switches, or even sensitive floor tiles[1]. Output devices can include home entertainment systems, wall-mounted displays, speakers, lighting, etc. Besides I/O devices, there will likely be devices dedicated to providing computational capacity to the system.

EasyLiving's goal is the development of an architecture that aggregates diverse devices into a coherent user experience. This requires research effort on a variety of fronts, including middleware, geometric world modelling, perception and service description. To motivate these areas of research, it is helpful to have a single concrete example scenario for reference.

2. Example Scenario

Tom is at home. He enters the living room sits down at a PC in the corner. He surfs through a selection of MP3's, and adds them to a playlist. He gets up and sits down on the couch. His session follows him to the large wall screen across from the couch. This screen is selected because it is available and in Tom's field of view. Tom picks up a remote control sitting on the coffee table and uses the trackball on it to request the room controls. They appear in a window on the wall screen, showing a small map of the room with the controllable lights. He uses this interface to dim the lights. Tom opens up his playlist and presses play. The music comes out of the room's large speaker system.

Sally enters the living room from the sliding doors to the outside and walks over to the PC. She has to manually log in, since she hasn't previously identified herself. She brings up a Word document that is an invitation to a shindig she and Tom are *hosting. Wanting Tom's input, she asks him if she can use the large room display. He uses the remote to release control of the wall screen, and she uses the room's controls on her PC to move her session to that display.*

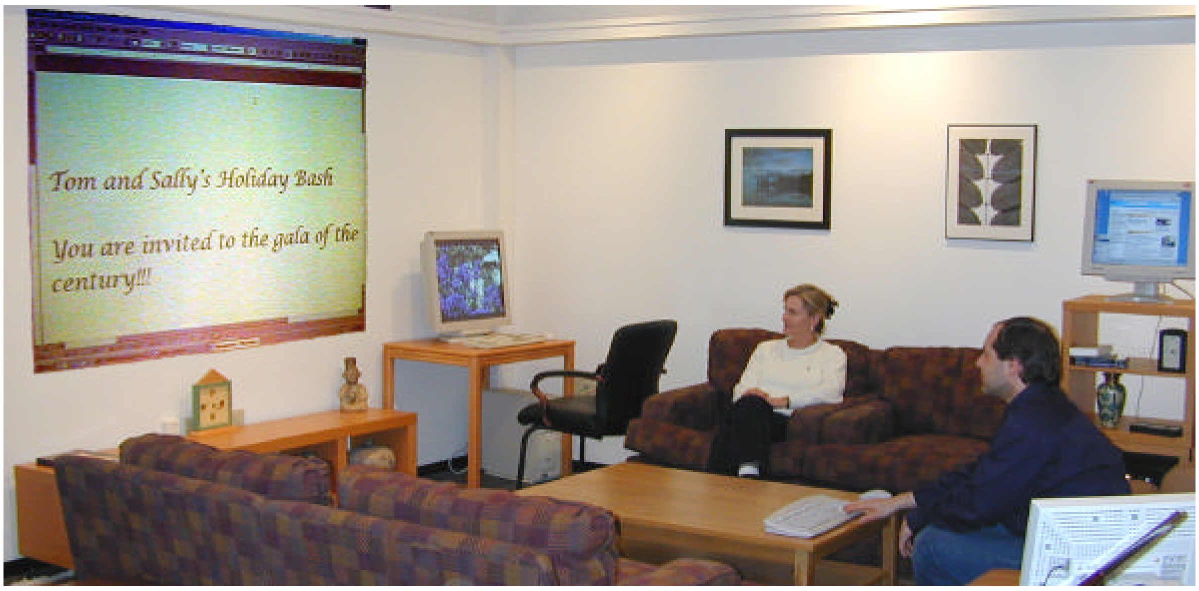

Tom and Sally's Holiday Bash
You are invited to the gala of the century!!!

Figure 1: Tom & Sally discuss a document in their Living Room

3. EasyLiving Technology Overview

To support desegregated computing, many of the traditional activities of an operating system must be supported across a distributed heterogeneous set of networked devices. In this paper, this class of software is referred to as Middleware.

As the number of available networked devices for a given user interaction increases, the complexity of identifying and selecting the appropriate devices for that interaction increases greatly. Furthermore, for the user to be able to specify which devices she wishes to use for a given task, a mapping between network and physical identity is exceptionally helpful. Note that rather than requiring Tom to know the precise name (e.g. "Light 37") of the lights, he selects them based on their location. The EasyLiving Geometric Model, capable of representing the physical relationships between entities in the world, supports these needs.

The need for perceptual information about world state further differentiates intelligent environments from traditional computing. Sensing devices allow the system to have information about the state of the world, such as locations of people, places, things, and other devices in the space. Having this information makes the interaction with the user seem more natural. When moving beyond the isolated desktop model, failure to give the system information about world state is likely to produce a complex or intrusive user experience. For example, if the system can send you an audible message anywhere, unwanted intrusions could occur if the systems fails to perceive such things as an important meeting or a sleeping child. Perception introduces significant complications for an intelligent environment, including the need to model uncertainty, perform real time data analysis, and merge data from multiple, possibly disagreeing, sensors. Stereo computer vision is currently used in EasyLiving.

Multiple dynamic devices motivate the need for the separation of hardware device control, internal computational logic and user interface presentation. Rather than tightly coupling input/output devices to applications, it should be possible to flexibly change the interaction mechanism without requiring modification of the underlying application. EasyLiving enables this kind of decomposition by providing abstract descriptions of their capabilities.

This paper describes the middleware, geometric modelling, sensing capabilities, and service description that comprise the EasyLiving system. In addition, some integrated applications and the services that run on top of these facilities are described. All of these technologies are discussed in light of the example scenario.

4. Middleware

In the example scenario, two possible models of device integrations can be used: peripherals in communication to a central machine or standalone devices in direct communication. In other words, a device (e.g. a light switch) could be wired to a dedicated computer controller, or it could be capable of communicating on a network. If devices are peripherals, they will not have the extra expense of providing their own user interface, however, if the central machine stops working, the peripherals cannot be operated. So, while the cost of a central server is likely to be lower, the difference is not great enough to offset the increased reliability afforded by standalone devices. Many small devices in collaboration can provide more cost-effective computing power for tasks like computer vision and speech analysis. Therefore, although both models provide complete solutions, the networked standalone device is the current likely future of ubiquitous computing and has been used as the basis for EasyLiving. Many other intelligent environments projects[4][7][15] have chosen a similar approach.

Given a collection of networked devices, the need arises for a mechanism that supports inter-machine communication. By utilizing a middleware package the effort required to build individual components that can communicate in this distributed environment is reduced. Several packages are currently available for this task, such as DCOM[8], Java[10], and CORBA[5], others[4] have recognized that Intelligent Environments place unusual demands on a middleware system. The following is a brief evaluation of the demands for inter-machine communication and dynamic configuration changes, and a description of the InConcert Middleware platform.

4.1 Inter-machine Communication

Current middleware environments built on synchronous semantics, like DCOM[8], Java[10], and CORBA[5], suffer from several failings. First, they force programmers to employ a multi-threaded programming model if they wish to avoid the latencies inherent in synchronous communications. For example, a single-threaded program that needs to interact with multiple peers would have to serialize its interactions with them rather than engaging in multiple interactions simultaneously. A single-threaded program will also be unable to do other useful work while waiting for a reply from a remote server. Worse yet, should the server fail or become unreachable, the program or device will be locked-up until a delivery time-out is reached.

A second failing of synchronous communication techniques is that pipelining of messages between two endpoints is very inefficient, even in a multi-threaded environment. If both the sending and receiving programs are multi-threaded, then by having each program fork multiple threads that communicate in parallel, pipelining can be approximated. However, for messages to have a well-defined arrival order, both the sending and the receiving programs must individually implement message serialization code. Furthermore, it would still not be possible to have only a single reply message for an entire batch of pipelined messages.

4.2 Dynamic Configuration Changes

Another problem stems from the manner in which processes describe the target for a message. DCOM[8], Java[10] require machine names as part of the address for the message. CORBA[5] provides for an object reference, but does not allow that reference to be updated dynamically. This results in delivery problems when the target is moved to another machine. Although not previously viewed as a common occurrence, note that users frequently transition between laptops, desktops, and home machines. This implies that components that are linked to a user may need to transition between a variety of machines in order to retain network proximity. Mobile devices also often change both physical location and network connectivity as they move with the user. These devices are added and removed from the collection of available hardware in a particular environment. Finally, for load balancing, many services are hardware independent and may be stopped on one machine and restarted on a different machine. In each of these situations, the clients involved need to have updated target addresses.

Beyond the problems of delivering messages to processes is the issue of message encoding. All of the discussed systems rely on fully decorated method names for communication endpoint bindings. This forces the clients to be updated in lock step with the server. This means changes must be done *en masse* rather than incrementally and that offline devices, like an out-of-reach laptop, must be updated upon joining the network.

4.3 InConcert

In conjunction with other groups at Microsoft Research, EasyLiving has been involved in the development of InConcert, a middleware solution that address these issues. InConcert provides asynchronous message passing, machine independent addressing and XML-based message protocols.

By using asynchronous message passing, InConcert avoids blocking and inefficiency problems. Furthermore, an asynchronous approach allows programs to handle offline and queued operation more naturally: clients are written to expect reply messages, if any are expected at all, to arrive at some arbitrary later time rather than as a return from the original request.

Inter-machine communication is handled by integrating a naming and lookup service into the delivery mechanism. When started, a component requests a name (an “Instance ID”) and while running provides the lookup service with a periodic keep-alive message. The name is unique to this instance of the component; it remains constant even if the instance is stopped and later runs on a different machine. Instance IDs are never reused. When sending messages, an instance includes its ID in the “From:” field of the message header. Receiving components can use that ID in the “To:” field of any response messages. When InConcert is asked to deliver the message it will resolve the ID by asking the Instance Lookup Service for the instance's current location.

Once the message is delivered to the correct process, its content is decoded from the XML description. XML provides the ability to version each field and add additional information that in other systems would cause the endpoint binding to fail. As new parameters become supported on servers, clients can either omit or include the information. If the server requires the new field, then the error message returned can describe the exact field required rather than reporting a simple binding failure.

By using the InConcert package, it is possible to develop new components for EasyLiving relatively quickly. This has made it possible to develop components that more accurately reflect the desired decomposition of interactions, as will be introduced in Section 7. It also allows components to be conveniently moved between hardware in order to tune the system performance. Finally, by designing the applications to handle asynchronous messages, the user experience is still responsive even when the device is isolated from all or part of the network.

5. Geometry

The desktop PC model assumes that the display, mouse, and keyboard are connected to a single machine and are all appropriately physically located. When working in a distributed environment, it is no longer viable to assume this static fixed device configuration, both in terms of device presence and physical configuration. Geometric knowledge, i.e. information about the physical relationships between people, devices, places and things, can be used to assemble a set of UI devices needed for a particular interaction. In the example scenario, geometric world knowledge provides three capabilities to Tom:

Physical Parameters for UI's: When Tom moves to the couch, his display is able to follow appropriately because the geometric model provides information which enables the selection of a visible display.

Simplified Device Control: When Tom starts playing the music, it is not necessary for him to select particular speakers or other AV components for the task. He was able to focus purely on the task, starting music playback, allowing the system to select devices based upon their location.

Shared Metaphor: When Tom turns down the lights, the provided map of the room allows him to quickly identify and control the needed devices based on their physical location. Without this representation, he would need to know particular names of the lights, or have some other way of mapping between physical and network identity. The geometric model provides this shared metaphor between the system and the user, allowing a more natural interaction.

5.1 Prior Work

Previous systems for geometric modelling have typically tightly coupled the sensor modality with internal representation and the application (e.g. using GPS, storing latitude/longitude, all for providing location-based reminders[13]). A more general system should decouple the sensor from the application, providing an internal representation which can support a wide variety of both. This section examines three broad classes of sensors and internal representations, highlighting the capabilities and disadvantages of each.

Outdoor Beacons: GPS and Cellular Phone-based location systems have been proposed for many scenarios, including determining driving directions and delivering reminders based on the user's location [13]. Both these sensors provide location data that can be translated into an internal representation of latitude, longitude, and elevation. Due to the physics of receiving beacon signals, these sensors are only useful in outdoor situations that are free of obstructions such as tall buildings. While it is possible to internally store the location of everything in lat./long./elev., the lack of an explicit uncertainty representation forces applications which query this model to assume some nominal uncertainty, typically the nominal GPS accuracy of 15m. This resolution is insufficient for most of the tasks in the example scenario.

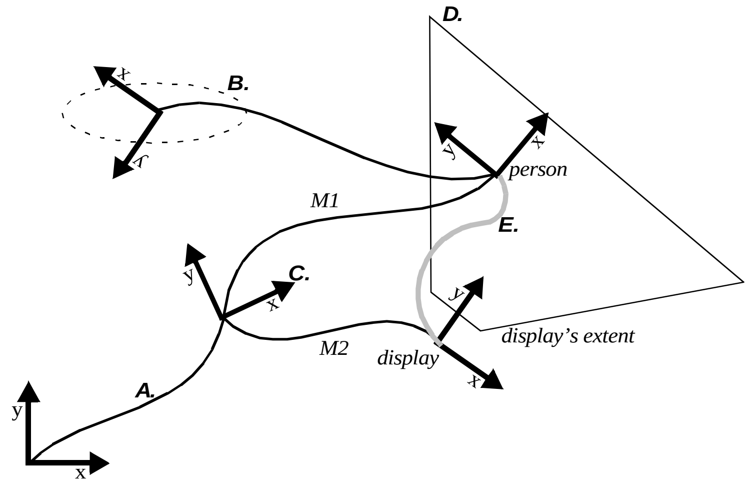
Indoor Beacons: In general, these systems consist of RF, IR or ultrasonic transceivers which can determine the presence[20] and perhaps location[21] of small (usually powered) tags which are attached to objects of interest in the world such as people, phones, printers, computers, etc. These systems represent geometry as a location in a single coordinate frame, such as a map of the building. To perform the example scenario, all items of interest (display, remote control, speakers, user, etc.) must be individually tagged. Additionally, if other positioning technologies are available, they are difficult to integrate, as they may not express their measurements in the same coordinate frame or with the same uncertainty. Active badge systems are useful for providing positional information, much as is GPS, but current systems[9] lack a general mechanism for expressing arbitrary geometric information, particularly that which includes uncertainty.

Network Routing: To avoid the perils of perception altogether, one can assume that network or data connectivity is equivalent to co-location[6]. This implies that if two devices can communicate directly (by RF, IR or other "local" transmission method), they are co-located. However, RF transmission (not to mention physical network protocols) can easily span rooms, floors or even buildings. Without some more precise model of geometry, this type of assumption will result in an excessively large set of potentially available devices, many of which may not actually be available or usable for any particular task due to the vagaries of the transmission method. Some systems base their notion of location on semantic tags applied to network or electricity connections, e.g. "Ethernet tap 324 is in Tom's living room." This is problematic both because it requires ongoing administration and because it can break the shared metaphor between system and user. For example, if Tom plugs a light into an outlet in the den, but places the light in the living room, this new light would not appear on his room controls due to the limited geometric representation.

5.2 EasyLiving Geometric Model

The EasyLiving Geometric Model (EZLGM) provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are myriad I/O, perception, and computing devices supporting multiple users. The EZLGM is designed to work with multiple perception technologies and abstract the application (including its user interface) away from any particular sensing modality. It is aimed at geometry “in the small”, that is, for applications which may require sub-meter localization of entities in the environment. Integrating this model with localization services for larger scales remains an open issue.

The base item in the EZLGM is an entity, which represents the existence of an object in the physical world. Measurements are used to define geometric relationships between entities. In particular, a measurement describes the position and orientation of one entity’s coordinate frame, expressed in another entity’s coordinate frame. Since objects in the physical world (as well as virtual objects like service regions) have some physical expanse, this can also be expressed as an extent in the geometric model using a polygon described in the coordinate frame of the entity. Additionally, measurements can have an uncertainty associated with them, expressed as a covariance matrix on parameters of the transformation[16].



A. Measurements are comprised of three components, x,y,theta, plus,

B. An uncertainty estimate, represented here as an equiprobability contour ellipse.

C. Each entity has a coordinate frame, in which measurements to other entities and extents are expressed.

D. A polygonal extent can be used to describe physical attributes (like size) or virtual attributes (like a service area).

E. Queries, like “What is geometric relationship between the display and the person?”, are resolved by combining measurements along a path, in this case, M1 and M2.

Figure 2: The EasyLiving Geometric Model

Once a set of measurements has been provided to the geometric model, the model can be queried for the relationships between entities’ frames. The measurements describe an undirected graph, where each vertex is the coordinate frame of an entity, and each edge is a measurement, as described above. If at least one path exists between two frames, then the graph can be processed to produce a single geometric relationship between the frames. Since a particular queried relationship may not have been previously directly measured, the response typically involves the combination of multiple measurements; uncertainty information is used to properly merge multiple redundant measurements as needed. Region-intersection queries are also supported, allowing, for example, selection of all devices within a particular radius of a user.

In the example scenario, the person tracking software continuously updates the measurement which describes the geometric relationship between Tom/Sally and the coordinate frame of the sensor which is observing them. Whatever process is responsible for keeping Tom’s session available on nearby devices can query EZLGM for all devices that have service areas that intersect with his location. The process first looks at types and availability to determine the set of devices which could provide the display, text input, pointing, etc. It then further prunes the list by considering the physical constraints (e.g. visibility) and electronic constraints (e.g. availability), in order to reach a set of usable, available, and physically-appropriate devices. Visibility can be checked by examining all entities along the line of sight between the user and the device and ensuring none have an extent present which represents something that physically blocks Tom's view. Then, once Tom's location is stable with respect to a set of devices, the session can be directed to automatically move.

5.3 Summary

EZLGM provides a mechanism for both determining the devices that can be used for a user interaction and aiding in the selection of appropriate devices. Note that no part of this example required any reference to the perception method which provided information about position: it could have been performed via some combination of cameras, badges, GPS, etc. with an appropriate uncertainty representation. Semantic location information can be powerful for many tasks, but it remains an open problem to gather and represent both semantic position tags and detailed geometric location in a single system.

6. Perception

Much of the information provided to the Geometric Model (and other attributed based directories) is data gained from the world through sensors. While much of this information could be entered into databases by hand, the more interesting case is when data is dynamically added and changed while the system is running. This data is gained from physical sensing devices that are attached to computers running perception components. These components support direct queries about information and keep data stores like the EZLGM updated as changes are detected.

6.1 Vision

Stereo computer vision is used as a way of tracking the location and identity in the example scenario. Vision is a natural sensing modality for this situation, because:

- Vision does not require that the room's occupants carry or wear any special devices.
- Vision can resolve the location of people in the room well enough to infer a person's intent based on his or her position. For instance, it can tell the difference between Sally's position at the PC and Tom's position on a nearby couch. Vision can even tell when either person stands or sits.
- Vision can maintain the identity of people in the room, allowing the room's devices to react to a specific person's personal preferences.
- Vision can be used to find objects in the room. For instance, cameras are used to locate the wireless keyboard on the coffee table.
- Vision can be used to make a geometric model of the room. Images from the people-tracking cameras are used to make a floor plan of the room.

While vision has unique advantages over other sensors for tracking people, it also presents unique challenges. A person's appearance in an image varies significantly due to posture, facing direction, distance from the camera, and occlusions. It can be particularly difficult to keep track of multiple people in a room as they move around and occlude each other. Although a variety of algorithms can overcome these difficulties, the final solution must also work fast enough to make the system responsive to the room's occupants. The current vision system, using two sets of stereo cameras mounted high on the room's walls, successfully tracks the location and identity of people in the room with an update rate of about 3.5 Hz.

6.2 Person Tracking

There has been much research in computer vision for tracking people. The EasyLiving person tracking software is described in greater detail in [11], which also contains numerous references to similar work.

The components of the tracking system are laid out in Figure 4. This tracker uses two color Triclops stereo cameras[18], each connected to its own PC. Two cameras are used so each part of the room is seen by at least one camera. The combination of color and depth from these cameras makes it easier to track people than if color or depth were used alone. These two PCs run the “Stereo Module” program, which processes the registered color and depth images to produce reports about the likely locations of people in the room. The Stereo Module first performs background subtraction to locate 3D blobs in each camera's field of view. The background is modelled with a combination of depth and color pixels in an effort to avoid the potential confusion caused by the moving video of the room's displays and by people sinking into the soft cushions of the couch. The 3D blobs are normally broken up over the regions of peoples' bodies. The Stereo Module merges the blobs by comparing different, hypothesized clusters of blobs to a simple model of a generic person shape. The Stereo Module also maintains color histogram models of each person in the room to help distinguish them from each other. The 2D ground plane location of the blobs is reported from each of the two Stereo Modules to a single instance of the “Person Tracker” which is used to integrate data from all the cameras.

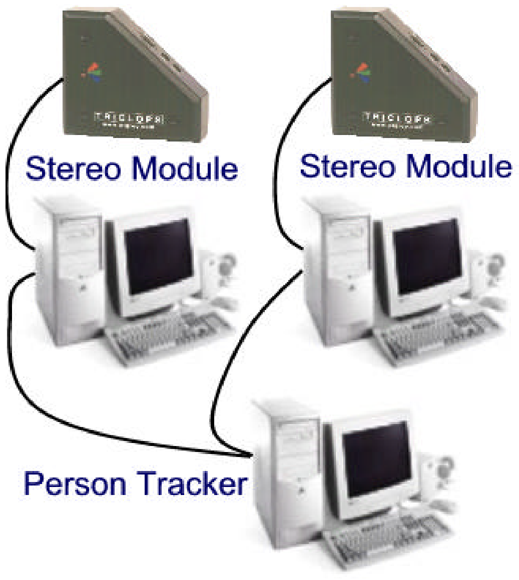


Figure 4: Visual Tracking SW Components

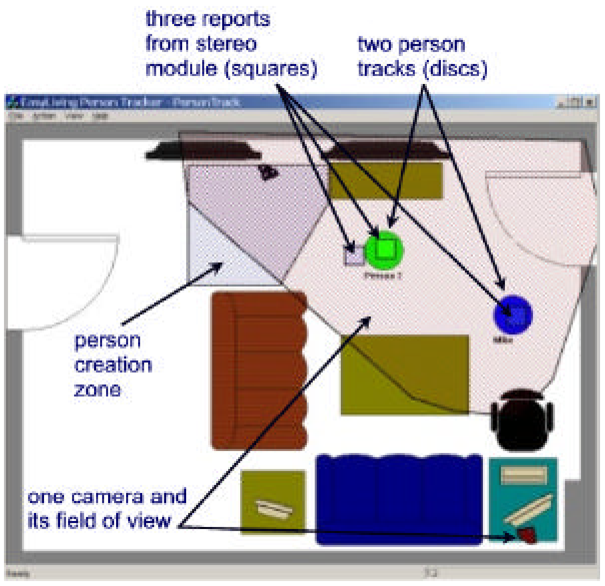

three reports from stereo module (squares)
two person tracks (discs)
person creation zone
one camera and its field of view

Figure 5: Person Tracking Display

The Person Tracker (Figure 5) uses knowledge of the two cameras' relative locations, fields of view, and heuristics on the movements of people to produce a final report on the locations and identities of people in the room. The Person Tracker is able to smooth over missing or mistaken reports from the two Stereo Modules by keeping track of the recent locations of each person in the room. If the Person Tracker becomes confused about the identity of a person, it appeals to the color histogram models maintained by the Stereo Modules. The Person Tracker also manages a special area in the room called the “person creation zone”. It is in this zone, normally established at the entrance to the room, that new instances of people are created and deleted as they enter and leave. The Person Tracker can also calibrate the locations of the cameras with respect to each other by watching a single person walk around in the room.

6.3 RADAR

Location information can also be gained from systems that track beacons instead of people. RADAR[2] is an in-building location-aware system being investigated at Microsoft Research. RADAR allows radio-frequency (RF) wireless-LAN-enabled mobile devices to compute their location based on the signal strength of known infrastructure access points. Knowing the location of the devices allows components that are running on that device to provide location aware interfaces. It also provides a means for inferring the user's location.

6.4 Identity (fingerprint reader)

One novel sensor that provides input to the system is a fingerprint reader manufactured by Digital Persona. This device is connected via USB to a machine with a database of fingerprints. When the user places a finger on the device, it automatically activates and sends messages that assert the identity of the person. This information can be used in combination with other components to assign a network identity to data that is currently being sensed. Knowing the measurement between the fingerprint sensor and the camera and the geometric extent in which a person can use the reader, allows the system to map the network identity to the reports from the vision system. This mapping can also be accomplished when the user logs in using a keyboard with a known location.

6.5 Device Tracking

The locations of moveable devices can be important for the behavior of the intelligent environment. For example, if a wireless keyboard is near a certain person, it can be assumed that keystrokes coming from that keyboard are being produced by that person. The keystrokes can then be properly routed to that person's active application. As shown in Figure 6, a camera mounted on the ceiling is used to locate the wireless keyboard on the coffee table in the room. The keyboard is detected in the image using a combination of color and shape cues.

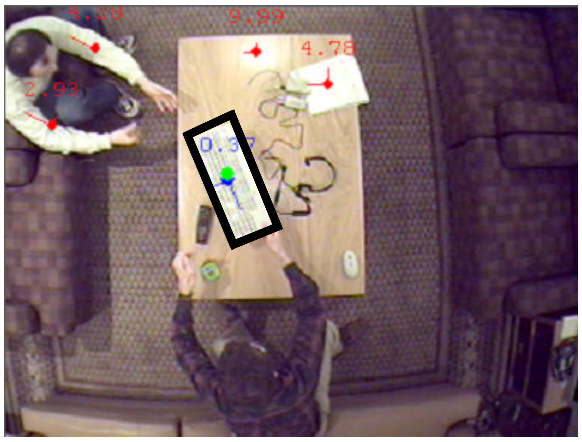


Figure 6: Keyboard Tracking

6.6 Integrated Perception

In the scenario above, these perceptual systems interact. When Tom and Sally enter the living room, they each pass through the person creation zone, at which point one of the Stereo Modules reports person-shaped blobs to the Person Tracker. The Person Tracker notes that the blob reports are coming from the person creation zone, and thus makes a new instance of a person to be tracked. The Person tracker starts keeping a history of the person's location, and reports the locations to the geometric model. The Stereo Module stores a color histogram to help disambiguate Tom and Sally if the Person Tracker becomes confused about who is who.

Tom and Sally are regarded as unknown people until they actively identify themselves somehow. Tom is assumed to have previously identified himself somewhere else in the house, and Sally logs on to a PC. Once authenticated, the system attaches each person's identity to its internal representation of that person.

This vision system works well for live demonstrations, with about 20 minutes of continuous tracking. During the demonstration, people enter and leave the living room, with their tracks being created and deleted appropriately. Tracking works well with up to three people in the room, depending on how they behave. With more than three people moving around, the frequent occlusions cause enough poor clusterings in the Stereo Module that the Person Tracker cannot maintain coherent tracks. Demonstrators are not required to wear special clothes, although similarly colored outfits can cause tracks to be misassigned due to indistinguishable histograms. The demonstrators can walk around, stand still, sit, and brush against each other without the system losing track of them. There are also large areas of moving video in the cameras' fields of view that the tracking system tolerates easily.

7. Service Abstraction

Like many intelligent environment systems[3][14], EasyLiving incorporates a variety of sensing techniques, software components for reasoning, and controllable devices, such as lights, computer equipment, and audio/visual hardware. Each of these pieces is encapsulated in a unique service, which encourages the separation of hardware device control, internal logic, and user interface presentation. Furthermore, abstract service descriptions allow each service to expose a set of attributes or commands so that other services may interact with it automatically.

For example, if current resources support pointing via trackball, mouse or visual gesture and, for each of these methods, there is a service that generates a pointing output, a web browser can be driven by any of those services, dependent upon user preference, context, or other selection mechanism. In the example scenario, when Sally uses the room controls to move her display, the service descriptions for the available devices are used to dandiacal provide her with possible destinations.

7.1 Prior Work

The concept of abstracting and describing services arises naturally when developing a system that involves automatic interaction between program components or exposure of device attributes. Several commercial systems under development, such as Universal Plug and Play[19], provide for device descriptions. However, they fail to differentiate between the interface presentation and service description. Mozer[3] has proposed separating device control from decision logic but did not allow for configuration changes. José[12] encoded context information into the XML service descriptions, but did not separate the service semantics from the service description.   Other intelligent environment systems[14] have not dealt with dynamic location-dependent services and automatically-generated UI.

7.2 Service Descriptions

Since an intelligent environment must support a changing collection of devices (and therefore services) it is necessary to handle service discovery. First, the system must discover the existence of newly available services. This is handled using InConcert's lookup capabilities. Next, it must determine the newly found service's capabilities.

Descriptions of services in the EasyLiving system are accomplished using a simple, open XML schema. In addition to ease of use, XML was chosen for two reasons. First, Extended Stylesheet Language (XSL) provides the ability to translate XML documents into multiple layouts. Second, it is straightforward to transform an XML-encoded description of a command into the XML-encoded command to be sent to the service.

The service description schema is designed to support queries about available commands and their legal values. Additionally, the commands are associated with human-readable tags. While not a complete solution, this is a first step toward the automatic generation of user interfaces for different modalities.

8. Demo Applications

The EasyLiving demo system utilizes the facilities described above to implement several applications within a one room intelligent space. This section describes some of these applications.

8.1 Room Controller

The Room Controller provides the user with direct access to the available services. The availability of a service is determined by intersecting the location of the user's current I/O hardware with the service's extent. The user interface is generated by examining each service description and displaying the appropriate XML documents. If there is no appropriate document, the

Room Controller generates a document by merging the published commands with a standard XSL stylesheet. In the example scenario, Tom uses a version of the Room Controller to adjust the lights and to start the music playback and Sally uses a Room Controller to move her session to the wall display. An example Room Controller UI is shown in Figure 7.

8.2 Remote Sessions

EasyLiving supports movable desktop sessions, similar to "Bat Teleporting"[9]. This facility can be controlled either automatically or by direct user action. Both methods utilize a service that handles the mechanics of session movement. The service that provides automatic behavior directs the session location based on the geometric relationship between the user and the available screens. Alternatively, the user can move the session using the Room Controller.

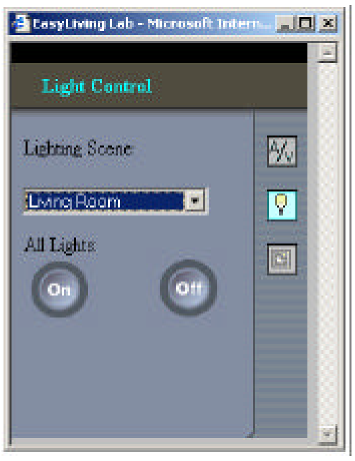


Figure 7: Room Controller UI

8.3 Mouse Anywhere

The lab is equipped with an RF mouse. There is no tag-based or vision tracking of this mouse. However, when the room contains a single person, the Mouse Anywhere service redirects the mouse commands based on the display service region the user currently occupies as determined by EZLGM. So, when the user brings the RF mouse near any display, the mouse controls the cursor on that display. Querying for the relative position of a person and a particular device can also be used to play an electronic version of the children's game "Warmer, Colder". Audio cues are provided to the user based on his movement towards or away from a random spot in the room.

8.4 Media Control

When a user is authenticated to the system, custom preferences are loaded that direct automatic behaviors. In the example scenario, one of Tom's preferences was a standing MP3 playlist. Similarly, users can have behaviors that direct various media types, for example, a CD, MP3, DVD or Videotape, that plays based on their location context. Defining automatic behaviors and preferences for an intelligent environment in a consistent user-friendly manner remains an open challenge.

9. Future Work

The EasyLiving project is building an architecture for intelligent environment. The design and implementation of this architecture is an ongoing effort. While some progress has been made, there are still a number of major issues to address.

Events: As the number of connections between services increases, polling ceases to be a viable mechanism for detecting changing state. Currently, it is not possible to register event requests like "Please inform me when entity 12 intersects the extent of entity 13" or "Please inform me when CD player is finished playing." Replacing polling with an asynchronous event system is a high priority for EasyLiving.

Lookup Services: Building robust lookup services that support discovery and scaling is still a major focus. While finding a service from a list of 20 is easy, building a system that can handle having thousands of services continuously updating their availability is a prerequisite to wide spread deployment of intelligent environments.

Extensibility: Moving from a single room to multiple rooms and hallways presents several new challenges. One geometric model may no longer suffice. Vision and other perception systems will need to cooperate and hand-off tracks of users between different disjoint spaces. Currently, it is unclear how services span boundaries between spaces or how these extents might be affected by network partitioning.

User Interface: As mentioned earlier, presenting the available services to the user in an understandable fashion and letting the user create and edit automatic behaviors are both on going work items.

The EasyLiving system can handle a single room and 10's of devices with dynamic changes to their configuration. One to three people can simultaneously use the facility. The system has evolved to the point that user interface issues can now be more rigorously examined. As EasyLiving evolves, it is expected that input and output devices will no longer be tied to a single machine or application but rather be able to flexibly support user interaction across a wide variety of tasks and modalities. Future work will build on this architecture, further exploring the migration of computing from the desktop and into everyday living.

Acknowledgements: Many thanks to Victor Bahl, Chad Ellman, Mary Czerwinski, Michael Hale, Steve Harris, Nebojsa Jojic, and Daniel Robbins for their invaluable contributions to the EasyLiving project.

10.References

1. Addlesee, M.D. et al, "ORL Active Floor", IEEE Personal Communications, Vol.4, No.5, October 1997, pp. 35-41.
2. P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF based User Location and Tracking System", Proceedings of IEEE INFOCOM 2000, Tel-Aviv, Israel, March 2000.
3. Michael Coen, "Design Principals for Intelligent Environments", Intelligent Environments, Papers from the 1998 AAAI Spring Symposium, March 23-25, 1998, Technical Report SS-98-02, AAAI Press.
4. Michael Coen, et al, "Meeting the Computation Needs of Intelligent Environments: The Metaglue Environment", Managing Interactions in Smart Environments, Springer-Verlag, 1999, pp. 201-213.
5. CORBA, "Remote Invocation", http://sisyphus.omg.org/gettingstarted/corbafaq.htm#RemoteInvoke .
6. P. Couderc, A.-M. Kermarrec, "Enabling Context-Awareness from Network-Level Location Tracking", Handheld and Ubiquitous Computing, First International Symposium, Springer-Verlag, 1999, pp. 67-73.
7. Anind Dey, et al, "A Context-based Infrastructure for Smart Environments", Managing Interactions in Smart Environments, Springer-Verlag, 1999, pp. 114-130.
8. Guy Eddon, et al, Inside Distributed COM, Microsoft Press, 1998.
9. Andy Harter, et al, "The Anatomy of a Context-Aware Application", Proceedings of the MOBICOM'99, August 1999.
10. Java, http://www.java.sun.com/ .
11. John Krumm, et al "Multi-Camera Multi-Person Tracking for EasyLiving", Third IEEE International Workshop on Visual Surveillance, July 1, 2000, Dublin, Ireland.
12. Rui José and Nigel Davies, "Scalable and Flexible Location-Based Services for Ubiquitous Information Access", Handheld and Ubiquitous Computing, First International Symposium, Springer-Verlag, 1999, pp. 52-66.
13. Natalia Marmasse, Chris Schmandt, "comMotion: a context-aware communication system", http://www.media.mit.edu/~nmarmas/comMotion.html .
14. Michael Mozer, "The Neural Network House: An Environment that Adapts to its Inhabitants", Intelligent Environments, Papers from the 1998 AAAI Spring Symposium, March 23-25, 1998, Technical Report SS-98-02, AAAI Press.
15. Nelson Minar, et al, "Hive: Distributed Agents for Networking Things", Joint Proceedings of ASA/MA, 1999.
16. R. Smith, P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty", International Journal of Robotics Research, Vol. 5, No. 4, Winter 1986, pp. 56-67.
17. Steven Shafer, et al, "The New EasyLiving Project at Microsoft Research", Proceedings of the 1998 DARPA / NIST Smart Spaces Workshop, July 1998, pp.127-130.
18. Triclops Stereo Cameras, Pt. Grey Research, http://www.ptgrey.com/ .
19. Universal Plug and Play, http://www.upnp.org/resources.htm .
20. Roy Want., Andy Hopper, "Active Badges and Personal Interactive Computing Objects", IEEE Transactions on Consumer Electronics. Vol 38. No.1, Feb. 1992, pp.10-20.
21. Andy Ward, et al, "A New Location Technique for the Active Office", IEEE Personal Communications, Vol. 4, No. 5, Oct. 1997, pp. 42-47.