Life Adaptive Computing:

Personalization of Computer Systems in the Mobile/Ubiquitous Age

Yuichiro Takeuchi

Submitted to the Department of Frontier Informatics
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
at The University of Tokyo

Thesis advisor:
Masanori Sugimoto
Associate Professor
Department of Frontier Informatics, The University of Tokyo

Abstract

Life Adaptive systems take the user's "life" as input. They adapt, morph, and evolve into forms that ideally fit the user.

In this dissertation I propose "Life Adaptive Computing", a new concept regarding how we humans interact with computers. Its general idea is to endow computer systems with the ability to deepen their user understandings through observing various details of the user's "life", and automatically adapt themselves to individual preferences, tendencies, and lifestyles. Just as humans gradually form a deep mutual understanding through each other's actions and words, Life Adaptive applications evolve toward optimal user experience, through careful observations of the user's daily life.

This dissertation first aims to establish the basic conceptual framework of Life Adaptive Computing, which should allow engineers/designers to easily integrate the concept into future application developments. Next, two examples of Life Adaptive applications are given, to offer a glimpse into the vast real-world possibilities of the concept. The design processes of these applications are also examined in detail, to demonstrate how using the conceptual framework can assist developers. On the whole, this dissertation should provide readers with a clear vision of what can be accomplished with Life Adaptive Computing, including even its limitations and potential risks.

Acknowledgements

I would like to express my deepest gratitude to the following people:

My mother, father and sister for their continuous support throughout my life. You have given me the rare privilege of being able to freely pursue my goals, and I fully owe to you everything that I accomplished in my past 27 years.

My outstanding friend Keishi, a fellow comrade in our (risky) quests toward fulfilling our different, but equally over-ambitious, dreams.

Thesis advisor Prof. Masanori Sugimoto and members of his lab, most notably Kazuhiro Hosoi and Sosuke Miura, for rigorously providing comments and suggestions on my work.

The people at Sony CSL, for showing me that a career in research, in rare cases, does not automatically lead to misery but can actually even be fruitful.

Friends Yoshiro, Osamu, Michinao and Ayuka, who helped me maintain sanity in a variety of ways, and in the course also convincing me that drinking uncountable bottles of sake is not a sin.

Kei, Hiroko, Atsushi, Yuko and everyone else, including some very important people whose names I cannot list here, who have supported this work or have contributed to making my life worthwhile in any way.

Table of Contents

List of Figures

List of Tables

1 Introduction: What is Life Adaptive Computing?

> *Life Adaptive systems take the user's life as input. They adapt, morph, and evolve into forms that ideally fit the user.*

In this dissertation I propose "Life Adaptive Computing", a new concept regarding how we humans interact with computers. Its general idea is to endow computer systems with the ability to deepen their user understandings through observing various details of the user's "life", and automatically adapt themselves to individual preferences, tendencies, and lifestyles.

Today we are witnessing an age of unprecedented progress in computer technology. The continuous tide of ever more powerful, compact, and inexpensive computational hardware has made a solid impact on the relationships between humans and computers, giving rise to new HCI (Human-Computer Interaction) concepts such as Ubiquitous Computing, Wearable Computing, Context-Aware Computing, etc. Life Adaptive Computing can be thought of as a yet another new HCI concept enabled by such technical advances, with an impact possibly comparable to the concepts listed above.

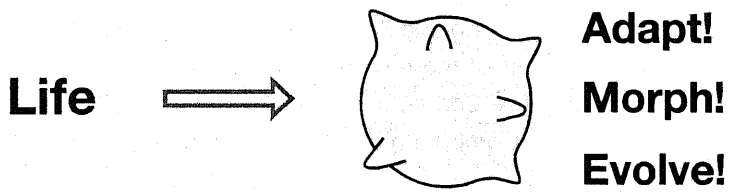

Figure 1. Life Adaptive Computing

Figure 1 illustrates the basic idea of Life Adaptive Computing. A Life Adaptive system (denoted by an amoeba-like blob, to the right of the arrow) takes the user's "life" as input, and by analyzing that information using a variety of means, it adapts, morphs, and evolves into forms that ideally fit the user. Even

seemingly trivial daily activities - shopping, taking photos, tying shoelaces, etc. - may have the power to serve as cues into the person's qualities (Figure 2). Just as humans gradually form a deep mutual understanding through each other's actions and words, Life Adaptive applications evolve toward optimal user experience, through careful observations of the user's daily life.

Figure 2. Examples of everyday activities used in Life Adaptive Computing

1.1 Relationships with existing HCI concepts

For a better theoretical understanding of Life Adaptive Computing, it must be made clear how the concept relates to, and can be differentiated from, existing HCI concepts. The two concepts most closely related to Life Adaptive Computing are Context-Aware Computing and Automatic Personalization. This section first provides brief descriptions of these two concepts, and by untwining their relationships with Life Adaptive Computing, aims to show the uniqueness and significance of Life Adaptive Computing within the field of HCI.

1.1.1 Context-Aware Computing

Context-Aware Computing refers to the idea of enabling computers to understand the "context" under which they are run, and to adjust their behaviors accordingly. Here, "context" is defined as any information that can characterize the interaction between a user and a system/application - location, time, user identity to name a few (More detailed definitions of "context" and "Context-Aware" can be found in previous work [1][2]). A frequently cited example of a Context-Aware device is a mobile phone that automatically switches to silent (or

vibration) mode when the user enters a movie theater, but the standard definition of Context-Aware Computing can also include more simple applications, like location-aware navigation systems.

Spurred by the proliferation of mobile computers and the resulting diversity in the situations of computer use, Context-Aware Computing has long been the subject of much attention as the "next big thing" in computer technology, especially the use of location information.

Furthermore, in this age where Ubiquitous Computing is quickly becoming a reality, we are constantly surrounded by a plethora of computers, not just desktops and mobile computers but also environment-embedded ones like sensor networks. Giving explicit orders to each and every one of these computers is impractical if not outright impossible even in this day, and thus it is essential for many computer systems to have a high level of autonomy, which Context-Aware Computing can serve as an effective tool to achieve.

1.1.2 Automatic Personalization

Automatic Personalization [3][4] is the idea of giving computer systems the ability to customize their functionalities, so that it best fits the attributes of the current user. Systems capable of Automatic Personalization adjust their behaviors by referring to the *user profile*, a formal representation of the distinguishing characteristics of each user. The central task in designing an Automatic Personalization mechanism is to find out an effective way to construct this profile, and possible methods range from simply having the user manually create the profile (this may seem to contradict the name *Automatic* personalization, but doesn't as long as the adjustment of system behavior is done automatically) to applying sophisticated data mining techniques to the user's past activities (mostly desktop activities, e.g. web history, search keyword history, etc.).

The concept of Automatic Personalization has long attracted not just the technical-minded but also the likes of entrepreneurs, consultants and marketers, due to its potential of offering effective CRM (Customer Relationship Management) solutions. Currently, the most successful example of real-world Automatic Personalization is the web-based item recommendation system, popularly used in online shopping sites such as Amazon.com. More recently, expectations for Automatic Personalization have been fueled even more by the growing prospect of personalized advertising.

1.1.3 Life Adaptive Computing

There is no doubt that Life Adaptive Computing is closely related to the above two concepts. But exactly *how* it is related is subject to multiple interpretations, as discussed below.

The first interpretation is to regard Life Adaptive Computing as a superset of Automatic Personalization (Figure 3). Here, Life Adaptive Computing is viewed as an extension of Automatic Personalization, where by incorporating Context-Aware techniques, the range of activities used for profile creation has been expanded to include the full scope of real-world activities. While the concept of Automatic Personalization, by definition, does not rule out the use of real-world activities, there has nonetheless been inadequate investigation into the area and previous work has focused almost exclusively on the use of desktop activities. Therefore, there is merit to this treatment of Life Adaptive Computing as a new concept, differentiable from conventional Automatic Personalization.

Since Life Adaptive Computing and Automatic Personalization both have the shared aim of "making computers adjust their behaviors to fit the user", treating Life Adaptive Computing in this way as an extension of Automatic Personaliization should seem natural from a user's point of view.

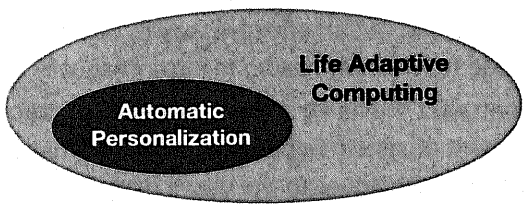

Figure 3: First interpretation

The second interpretation is to regard Life Adaptive Computing as a subset of Context-Aware Computing (Figure 4). Judging from the stream of past studies on Context-Aware Computing, there seems to be no definitive consensus as to what information should be included in the list of "contexts". However, if we include "user identity" into the list, as is done in Dey's categorization [2], Life

Adaptive Computing can be thought of as a module within (or a restricted version of) Context-Aware Computing, a mechanism that acts as a partial delegate of the functionalities of Context-Aware Computing by automatically gathering and adapting to information about each user.

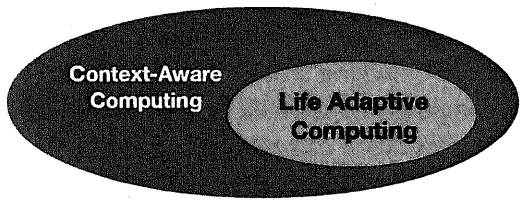

Figure 4: Second interpretation

The third interpretation treats Life Adaptive Computing as a brand new concept, partially derived from Context-Aware Computing and Automatic Personalization but is neither a superset nor a subset of these two (Figure 5). Here, Life Adaptive Computing borrows the real-world observation methods and the user-oriented approach of the respective concepts, but has sufficient disparities to be worthy of being considered a separate concept.

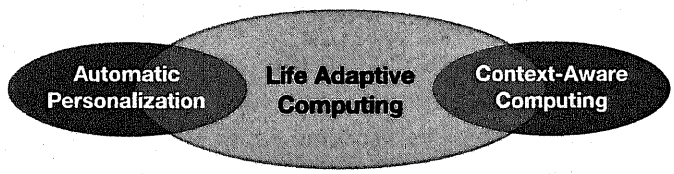

Figure 5: Third interpretation

None of the above three interpretations can be considered an outright fallacy. As with most HCI concepts, there is a certain level of vagueness in the definitions of Context-Aware Computing and Automatic Personalization, which inevitably creates room for equivocality in their relations to Life Adaptive Computing.

1.2 A world of possibilities

Notable HCI concepts of the past - Ubiquitous Computing, Tangible Bits, Computer Augmented Environments, etc. - have all presented captivating visions of how they could transform our daily lives, society, and the world. This section illustrates the wealth of future possibilities realized with Life Adaptive Computing, accepting the interpretation shown in Figure 3 where Life Adaptive Computing is treated as an extention of conventional Automatic Personalization. As noted earlier this interpretation should seem quite natural from a user's viewpoint, since Life Adaptive Computing shares with Automatic Personalization its objective of "making computers adjust their behaviors to fit the user", albeit with considerably richer adaptation capabilities.

Conventional Automatic Personalization, with all its technical limitations, has proven to be a remarkably successful concept market-wise. By embracing the real-world sensing skills of Context-Aware Computing, Life Adaptive Computing offers even brighter prospects, opening the door to a broad array of new applications. Now every minute detail of the user's daily life, not just desktop activities (e.g. web history, etc.), can serve as potential data sources for building user understandings (Figure 6). Described below are some examples of possible Life Adaptive applications.

Figure 6: Away from the desktop

Book recommendation based on TV viewing history

The Amazon.com item recommendation system, often cited as the quintessential Automatic Personalization application, makes recommendations by analyzing each user's past online activities. This system can be extended into a Life Adap-

tive application by focusing instead on alternative user activities, for example TV viewing history. In this case, a non-fiction book may be recommended to users who regularly watch documentary programs.

Shop recommendation based on real-world shopping records

In a similar manner, recommendation systems for a variety of domains can be designed. One example is a system where each user's *real-world* shopping activities are monitored, which are then used to recommend shops in the city that may match his/her preferences.

Training regimen design based on daily physical movements

Not all Life Adaptive applications need to be recommendation systems. It may be possible to develop a system that tracks the user's daily physical movements and uses that information to design a customized training regimen, to assist the user stay healthy and fit.

Matchmaking system based on daily activities

Survey results show that having shared hobbies or lifestyles greatly improves the chances of a couple's successful relationship. Matchmaking systems have been around for quite a while now, but a Life Adaptive matchmaker that focuses on similarities in daily activities may become the next breakthrough in the area.

Life Adaptive career consulting

Recent university graduates often have little experience out of school, which paves the way for some regrettable career decisions. A Life Adaptive career consulting system, which presents a personalized list of career options, may effectively curb the user from making the most disastrous choices.

The above examples represent only a tiny fraction of what can be realized with Life Adaptive Computing. Eventually a wide array of Life Adaptive applications, serving functions yet unimaginable at the current moment, can be expected to emerge in the hands of future engineers/designers.

1.3 Manifesto

The idea of Life Adaptive Computing, discussed in this thesis, is one that had actually been long shared by many engineers and researchers, albeit as a vague and abstract vision. The success of the Amazon.com recommendation system has allowed us to catch a glimpse into the future, where computer systems become capable of a much more sophisticated personalization, using the full range of user activities for adaptation.

However, an abstract vision by itself can only do so much. It needs to be materialized into a concrete concept, before it can make tangible contributions to the society through real-world implementations. This is precisely what I aim to achieve as the objective of this research, through exhaustive discussions leading to the establishment of the basic conceptual framework of Life Adaptive Computing.

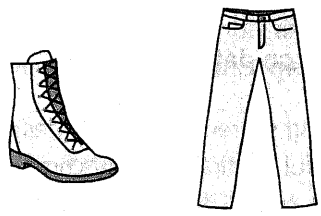

Figure 7: Life Adaptive artifacts

1.4 Relationships with existing artifacts

If we expand our view to outside the boundaries of Computer Science or HCI, we can list several existing artifacts that share similar characteristics with Life Adaptive applications (Figure 7). These artifacts, which include denim jeans, leather shoes, etc., adapt to each user by making use of the distinctive properties of their composing materials. In the case of denim jeans, the properties of the denim fabric are what provides them with their adaptive capabilities. Interestingly, even if two people with identical body shapes start wearing the same jeans on the same day, eventually their jeans will grow into dramatically different colors and forms, depending on their lifestyles. Put simply, denim jeans adapt not

just to individual body shapes but also to each user's "life". Thus Life Adaptive Computing can be regarded as an extension of these existing artifacts, where the only noteworthy difference lies in the type of material used to obtain their "adaptive" abilities.

Such a statement may seem to downplay the impact of Life Adaptive Computing, but there is actually a tremendous growth of potential here; Life Adaptive Computing, through the use of computers, makes adaptations possible in ways unthinkable using conventional materials like denim or leather.

1.5 Inspirations

I would like to conclude this chapter by stating my initial inspirations for this research. This project was heavily inspired by the works of the product designer Naoto Fukasawa [5], whose design focuses on touching unconscious actions and sensibilities that are "shared" among many people. Upon extensive study of his work from the perspective of a Computer Science/HCI researcher, I came to shape my initial idea that through aggressive use of the power of computers, a new class of product design may be realized, focusing on actions and sensibilities that are *not* shared, but are instead unique to each person. The concept of Life Adaptive Computing, presented in this thesis, is a product of further investigations into this initial idea.

From here, the paper will proceed as follows. The next chapter provides a series of formal discourse on Life Adaptive Computing, culminating in the establishment of its basic conceptual framework. Further ahead, two examples of Life Adaptive applications will be introduced, and examined in detail as case studies. The design processes of these applications will also be analyzed, to demonstrate how engineers/designers can make use of the framework in their future application developments. The thesis will conclude thereafter, following discussions on several supplementary topics. On the whole, this dissertation should give readers a clear idea about the possibilities of Life Adaptive Computing, what it can accomplish and how it will affect our future lives, while also exposing its limits and potential risks.

2 The Conceptual Framework of Life Adaptive Computing

This chapter presents a series of discussions, with the ultimate goal of establishing the basic conceptual framework of Life Adaptive Computing. Let us kick off the discussion with the following question: what is a conceptual framework, and why is it needed?

Compared to some existing HCI concepts such as Ubiquitous Computing, which introduce a brand-new paradigm to computing and encompass a wide array of ideas, Life Adaptive Computing is relatively narrow in its scope, referring to a particular class of applications. These applications (i.e. Life Adaptive applications) can be assumed to have some level of similitude in their basic structures, and by extracting and explicitly articulating the common features, we should be able to construct a rich base of knowledge, or a *framework*, that could serve as a starting point for future developments of Life Adaptive applications.

Furthermore, if the idea of Life Adaptive Computing turns out to have sufficient impact, attempts at developing toolkits for easy prototyping of its applications may emerge. A good framework should provide helpful leads for such attempts.

2.1 Basics

Figure 8 illustrates the basic procedure with which Life Adaptive applications offer their "adaptive" services. The procedure consists of three "phases". First, in the Activity Observation phase, the user's activities are recorded and accumulated using various technologies. That information is then passed on to the User Profile Update phase, where the user's profile is altered to better reflect his/her unique characteristics. Finally, in the Service Generation phase, the application generates and delivers services tailored to suit the user.

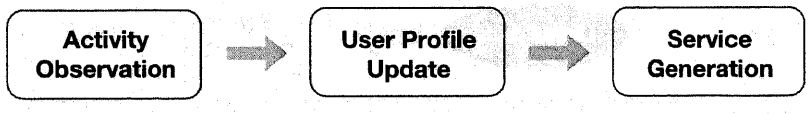

Figure 8. Basic procedure of Life Adaptive service delivery

This basic procedure applies to *every* Life Adaptive application, which is actually tautological since that is how Life Adaptive applications are defined - applications that perform observations of the user's "life", and offer customized services. In some applications, sensor inputs from Activity Observation could be directly passed on to Service Generation, resulting in the apparent absence of an explicit User Profile Update phase. However, even in those special cases the procedure could still be seen as following Figure 8, if we regard the User Profile Update as an "empty" process that does nothing. Therefore the diagram in Figure 8 can be viewed as a universal generalization of Life Adaptive service delivery.

Another thing to note is that the term "service" in Service Generation has a fairly broad meaning here. The first chapter has cited some examples of possible Life Adaptive applications, but the range of "services" that can be provided using this concept is much more diverse. Extreme future possibilities include a Life Adaptive chair which can change its form to best fit the user's physical attributes/health conditions, or Life Adaptive artwork that morphs to suit the owner's aesthetic tastes. The term "service" in Life Adaptive Computing encompasses such varied visions, which may diverge somewhat from its common definition.

In actual implementations of Life Adaptive applications, the phases in Figure 8 may not be performed in the order designated by the arrows. This can be shown using a simple example. Suppose we want to design a Life Adaptive application that collects each user's real-world shopping activity records (e.g. information of shops which the user has visited), and offers recommendations of shops by analyzing those records (Actually, this application will be introduced as one of the case studies later in this thesis). One possible implementation of this system will be as illustrated in Figure 9.

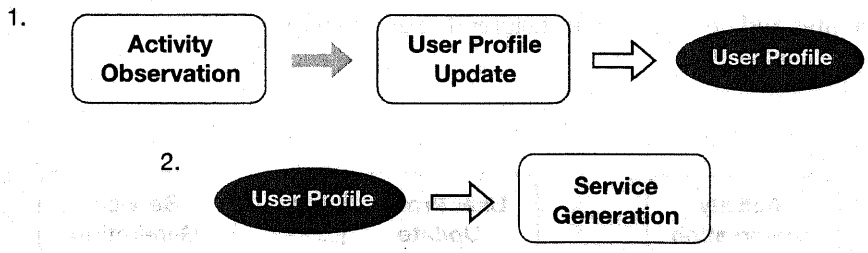

Figure 9. Possible implementation of the shop recommendation system

Here, the procedure shown in Figure 8 is broken into two *sub-phases*. In the first sub-phase, the user's shopping activities are recorded, which are subsequently used to shape his/her user profile. In the second sub-phase, actual recommendations are performed based on the updated user profile. The significance here lies in the fact that these two sub-phases exist on separate timelines. The first sub-phase takes place continually, while the second is triggered only upon spontaneous user requests. As a result, the actual chronological order in which the three phases are executed can be, for example,

[(Activity Observation → User Profile Update) x n] → Service Generation.

This is not to say that Figure 8 is incorrect, although it may give the impression of being so. Instead, this fact simply shows that the arrows in Figure 8 are not intended to represent the correct chronological order of the operations. Rather, they serve as abstract representations of the directions in which these three phases influence one another.

Figure 10. Basic procedure of Life Adaptive service delivery (2)

To make this more clear, a redrawing of Figure 8 is shown in Figure 10. Activity Observation results in accumulation of "Activity Data", which is passed on to User Profile Update. User Profile Update, in turn, results in the modification of the "User Profile", which is then passed on to Service Generation. Data handover within a Life Adaptive application is always done in this direction, and not in any other ways, regardless of the implementation.

It may be argued that if we interpret the arrows this way, there should also be an arrow pointing back to Activity Observation from Service Generation, as is illustrated in Figure 11. The logic here seems clear, since services offered by a

Life Adaptive application often end up affecting the user's activity. In the case of the shop recommendation system cited above, if the user visits any of the recommended shops it would signify direct influence on user activity, and long-term system use may have more far-reaching effects. However, the discussions in this paper will omit this arrow and stick to using the diagram in Figure 8. The rationale is that, even if Life Adaptive services induce profound changes in user activity, its effect will automatically show up in the outcome of Activity Observation without any need for special consideration. Since the conceptual framework provided in this paper is aimed at assisting developers of Life Adaptive applications, we can ignore elements to whom there exist no need for the developer to model or anticipate beforehand.

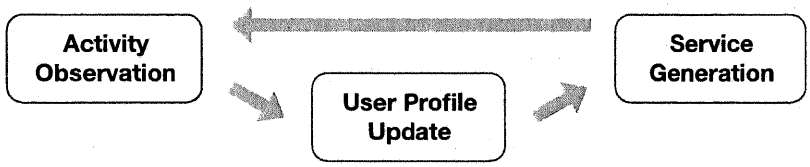

Figure 11. Basic procedure of Life Adaptive service delivery (3)

Now we will move on to more detailed discussions on the framework. As a general principle, we seek to establish the conceptual framework through in-depth, and separate, examinations of the three individual phases. In other words, we dismiss the significance of inter-phase relations. The appropriateness of this will be argued later in this chapter. Below is a list of the topics (seven in all) that will be discussed.

- **Activity Observation**
  - (Topic 1-1) Modes of observation
  - (Topic 1-2) Types of activities
- **User Profile Update**
  - (Topic 2-1) User profile anatomy
  - (Topic 2-2) Targets of adaptation
  - (Topic 2-3) Granularity of user profile
- **Service Generation**
  - (Topic 3-1) Service generation techniques
  - (Topic 3-2) Use of context information

2.2 Activity Observation

(Topic 1-1) Modes of observation

The ways in which Life Adaptive applications observe and record daily activities can be classified into two basic modes, *direct observation* and *indirect observation* (Figure 12, note that "Adapt!" is meant here to be an abbreviated expression of User Profile Update → Service Generation).

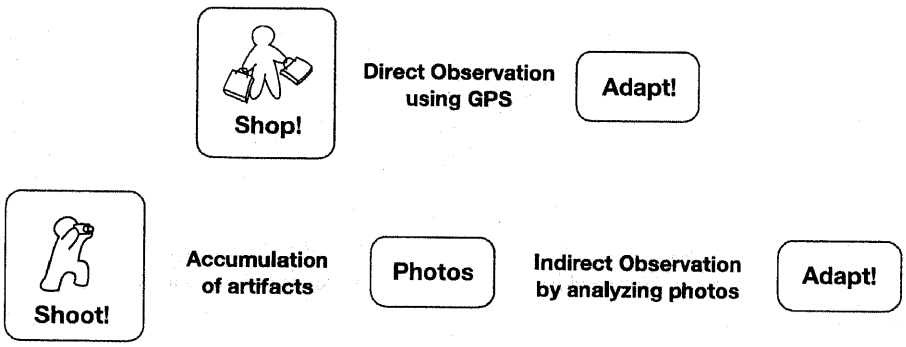

Figure 12. Direct observation (top) and indirect observation (bottom)

Direct observation refers to the approach of acquiring information at the time the activity is performed, mostly through the use of sensor technologies. Among the applications described later in this thesis as case studies, the intelligent city guide CityVoyager, which observes location changes using GPS, falls into this category. Recently, with the widespread use of various mobile computers and the increased technical and economic feasibility of environment-embedded sensor systems, direct observation is becoming more and more practical.

Indirect observation refers to the approach of retrogressively monitoring past activities, through the analysis of "marks" or "artifacts" left behind as results of the user's activities. Among the applications described in this thesis, the video summarization system TimeWarp, which examines personal photo libraries, falls into this category.

Designers of Life Adaptive applications must make decisions as to which mode of observation best fits their particular situations. For example, suppose we design a Life Adaptive application that uses information of books that the user has bought in the past. If we decide to use direct observation for this task, information must be acquired each time the user makes a purchase, of which one possible implementation would be to track credit card usage. If we decide to use indirect observation, on the other hand, we may instead focus on the user's bookshelf. By looking at what books are contained in the bookshelf, we should be able to make a pretty good guess about the user's past book purchases, if not 100% accurate. Here, each approach has its advantages and drawbacks, and it is up to the designer to decide which would be the more suitable choice.

(Topic 1-2) Types of activities

In principle, an infinite range of user activities can be used in Life Adaptive Computing. However, if we consider the current or near future state of technology, there are limits to the types of activities that can realistically be observed. Since acquisition of real-world activities has long been a subject of rigorous investigation in fields such as Context-Aware Computing, we can construct a pragmatic list of activities that can be used in Life Adaptive Computing as below, by referring to a rich body of previous work.

Creation
Creation of new data. Shooting photos/video, drawing pictures, writing, etc. Generally observed indirectly through the analysis of the created artifacts.

Consumption/Collection
Item purchases, data downloads, etc. Indirect observation is more practical, but direct observation can be performed by, for example, tracking credit card usage.

Location change
Location data in various levels of granulary, from latitude/longitude to cities/countries. Mostly observed directly using GPS, Wi-Fi, etc.

Speech
Information of the user's conscious/unconscious speech, extracted using various speech recognition techniques from microphone recordings.

Interaction with people
Information about people whom the user had met, spoken, etc. User identification using smart tags, biometric sensors, etc. may be necessary.

Interaction with items
Use history of products, applications, media data, etc. Appropriate observation techniques differ with the type of item.

Motion
Users' physical movements. Includes a wide range of activities observed using a combination of sensors, such as accelerometers, magnetometers, etc.

Other activities
Activities which do not fit into any of the above categories, such as eye gaze. In most cases, observation requires elaborate techniques.

Accompanying context
Information not of user activity itself but of context information accompanying the activities, such as temperature, ambient sound, etc.

Observation of these activities can be done either directly or indirectly; designers of Life Adaptive applications must make decisions as to which approach would better suit their specific state of conditions.

Maintaining a strong curiosity towards people's everyday lives, and making a habit of scrupulous observation, should be beneficial in coming up with ideas for Life Adaptive applications. Even trivial, mundane activities, such as drinking water from a bottle, can contain details unique to each person. If a link can be discovered between a particular activity and some quality about the person (Figure 13), it may serve as a clue to new designs of Life Adaptive applications.

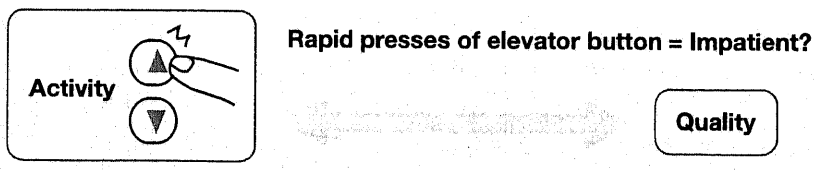

Figure 13. A "hidden link"

Based on the above discussions on Topics 1-1 and 1-2, the procedure in Figure 8 can be redrawn as in Figure 14.

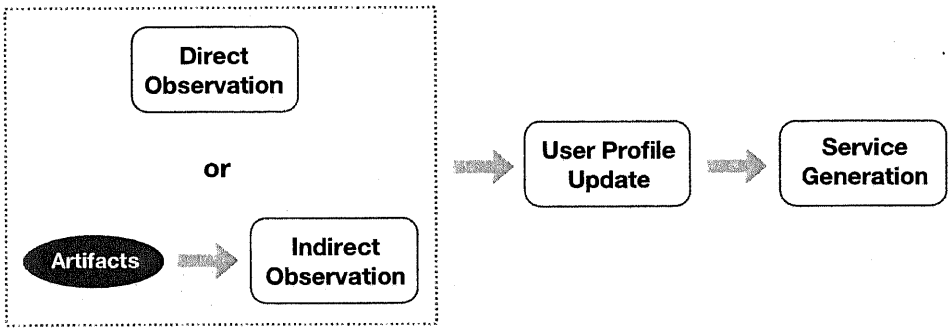

Figure 14: Basic procedure of Life Adaptive service delivery (4)

2.3 User Profile Update

(Topic 2-1) User profile anatomy

A user profile can take many forms - it can be raw chunks of activity data used without any preprocessing, or it can take a highly compressed form acquired by sending raw data through a series of abstraction processes. It should be good practice to compress activity data to the extent that the resulting data loss is permissible, since it will be advantageous both in terms of memory usage and protection of privacy (for compression naturally results in only the most relevant personal information being stored). This paper will not discuss the possible actual formats of user profiles, since they are heavily application- and implementation-dependent.

One thing worth consideration when designing a strategy for profile updates is whether "recent" activity data should be treated any differently from the more "older" data. For example, suppose we consider the design of a Life Adaptive dinner-place recommendation system, based on the user's past visits to restaurants. In such a case, it seems natural that information about the more recent visits should exert stronger influence on the characteristics of the user profile.

Such a mechanism for changing the treatment of activity data based on their time-stamps can be implemented in a variety of ways, and should be exploited when deemed of benefit.

Taking this one step further, we can see that it is possible for the user profile to contain a separate section, which reflects not the user's long-term activity data but instead real-time context information (Figure 15).

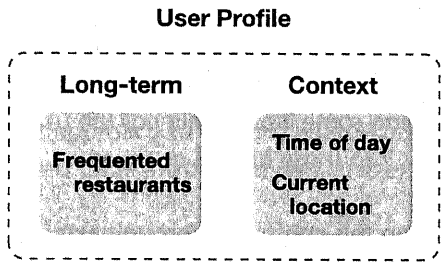

Figure 15: User profile with a "context" section

If we discard the long-term section altogether from the profile in Figure 15, an interesting thing happens - the Life Adaptive application actually turns out to be a Context-Aware application. Seeing this fact, it might be tempting to think that Context-Aware Computing is really a subset of Life Adaptive Computing. However, as shown in Section 1.1 this paper does not adopt that view, for the reason that Context-Aware Computing does not necessarily require the existence of a human user, as in the case of autonomous robots equipped with Context-Aware sensing capabilities. On the contrary, by definition Life Adaptive Computing always assumes the presence of a human user, which makes it impossible to include autonomous robots, and hence the whole concept of Context-Aware Computing, as part of Life Adaptive Computing.

In reality, treating real-time context as part of the user profile as in Figure 15 probably only adds to the complexity of the implementation, and is not a good idea. A better way to make a Life Adaptive application account for real-time context should be to regard context as an external entity outside of the user profile, as will be explained in Topic 3-2.

(Topic 2-2) Targets of Adaptation

What attributes of the user does a profile actually represent? Or more broadly, what do Life Adaptive applications actually adapt to? Taken literally, Life Adaptive applications should adapt to each user's "life"; in other words, the user's "life" should be the "target of adaptation". This sounds reasonable enough, since observation of the user's "life" is what enables the adaptive capabilities of Life Adaptive Computing in the first place. However, the truth may not be so simple. In Topic 2-1 we cited an example of a Life Adaptive application, a dinner-place recommendation system based on past visits to restaurants. In this example, the target of adaptation seems to be not so much the user's activities (or "life") themselves, but the *preferences* of the user that manifest itself through careful analyses of the activities. Thus it may also be possible to interpret the target of adaptation as being some inner quality of the user, with "life" merely serving as a looking mirror into those hidden qualities.

Here we attempt to clear up this confusion, through extensive discussions as to what can be the target of adaptation in Life Adaptive Computing. Findings here should make up one of the central pieces of the conceptual framework, since it provides developers with valuable insights into what types of applications can be (or can not be) constructed using Life Adaptive Computing.

We will open the discussion by considering the following two approaches.

Fundamentalist Approach
Pragmatist Approach

Fundamentalist Approach

In the Fundamentalist Approach, we dismiss any existing restrictions (technical or otherwise) and try to grasp what can "fundamentally" serve as the target of adaptation. We seek to obtain a list, of every information that can possibly be uncovered through the process of examining the user's "life" - factors which serve as the primary cause of the uniqueness evident in each person's actions. What we are after here may loosely be called the user's *personality*, and as such, existing literature in fields such as Psychology and Neuroscience seems capable of providing us with our desired knowledge.

However, at this point in time this approach is destined to fail. Personality is a subject rigorously studied but still largely unknown, subject to a number of contradicting explanations and theories. And the few relatively well-established models such as the Big Five Personality Traits [6], are too abstract, lacking a level of detail necessary to prove useful for Life Adaptive application developers.

Pragmatist Approach

The Pragmatist Approach aims to compile a "pragmatic" list of possible targets of adaptation, through a survey of existing Life Adaptive applications and examining what each previous application had strived to "adapt" to. Unfortunately this approach is also destined to fail, for the simple reason that currently there are not enough Life Adaptive applications out in the world to conduct a thorough survey. For this approach to be valid a large number of diverse Life Adaptive applications need to be introduced, which is expected to take at least a few more years of time.

Alternative Pragmatist Approach

The two approaches that we described, the Fundamentalist Approach and the Pragmatist Approach, have both been shown to fail. Does this mean it is currently impossible to attain comprehensive knowledge about the target of adaptation? Maybe so, and it might be a realistic option to avoid detailed discussions about this topic for now, since of the two approaches, at least the Pragmatist Approach should become feasible in the near future, with further introductions of Life Adaptive applications. However, here we proceed with the discussions by pursuing a third approach, on the belief that there is value in delivering discourse even if the argument may be somewhat incomplete. This third approach is named the Alternative Pragmatist Approach, due to its being a modification of the Pragmatist Approach.

Recall that in Topic 1-2, to obtain a list of activities that can be observed in Life Adaptive Computing, we referred to a body of previous work in Context-Aware Computing. This was only possible since Context-Aware Computing has a shared concern with Life Adaptive Computing in its need for real-world activity detection, where they are both subject to the same technical restrictions. The notable fact here is that the list of activities was compiled not through studies of existing Life Adaptive applications, but through examinations of past literature in an external field, which in this case was Context-Aware Computing. The lack of Life

Adaptive applications at the present time had forced us to explore separate fields with similar interests.

We take a similar strategy in the Alternative Pragmatist Approach, this time by trying to find an external field that is comparable to Life Adaptive Computing in its focus on "adaptation". Automatic Personalization first comes to mind, but the "target of adaptation" is a topic that has somehow been mostly neglected in the area and thus it fails to meet this condition.

Instead, we turn to the rich pool of knowledge in the field of industrial design. Modern industrial design, a concept traced back to 19th century Britain by most historians [7], has spawned numerous ideas and methodologies, many of them successfully being put into practice. In particular, some of the most prominent concepts proposed in the 20th century, such as Modulor [8] and Ergonomics, have explored the idea of designing products in harmony with the physical and psychological characteristics of humans. Due to the fact that modern design has been primarily concerned with designing mass-produced products, these concepts are cut out to realize design that "best fits a large number of people", and thus may seem like exact opposites of Life Adaptive Computing, which takes a more individualistic approach. However, a quite different interpretation is possible where we treat these design methodologies and Life Adaptive Computing as all having a shared objective of "designing products that suit a specific user profile", where the sole variable is the "granularity" of the user profile. Figure 16 illustrates this interpretation.

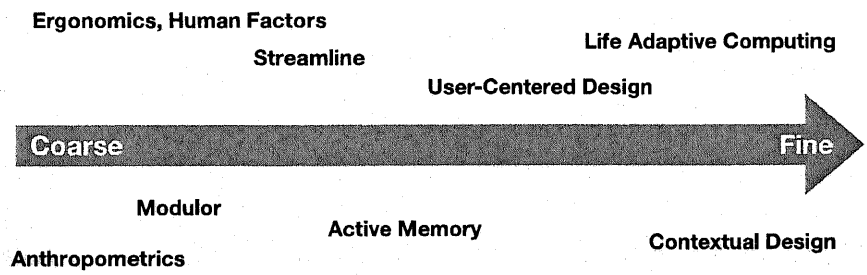

Figure 16: Design concepts and their granularities of user profile

For concepts such as Modulor and Human Factors, which aim for near-universal adaptation (adaptation to the masses), the user profile is "coarse" (i.e. representing a large number of people) and thus they are located at the left end of the axis. For concepts such as Life Adaptive Computing and Contextual Design [9], which adapt to individuals or small, distinct groups of users, the profile is "fine" and they are located at the right end of the axis.

This way, by focusing on the common ground between past design concepts and Life Adaptive Computing, we may be able to extract the possible targets of adaptation in Life Adaptive Computing, from the rich fabric of industrial design knowledge. If we assume that application designers will use Life Adaptive Computing for benefits generally in line with those pursued using past design methodologies, the primary "targets of adaptation" in Life Adaptive Computing should be quite similar to the "targets" sought out under existing design concepts. Therefore, a reasonable list of "targets" can be constructed through investigations into the concepts shown in Figure 16.

Below is the (provisional) list of the possible "targets of adaptation", compiled using the above approach.

Physical

Body Dimensions (includes information such as range of joint movement)
Motions (uniqueness evident in physical motions)

Psychological

Tendencies (ways of thinking, character, penchants)
Preferences, Sensibilities (may be either concious or unconcious)

Miscellaneous

Lifestyles (high-level information that cannot be reduced to either of the above)

Note that in this list, some terms are used without any care as to whether they can be considered scientifically sound concepts. For example, in psychology the existence of long-term "preferences" seems to be a notion that is far from infallible but susceptible to many criticisms. However, the intended audience of this conceptual framework are not scientists committed to basic research, but future developers of Life Adaptive applications who, like most people, probably take these concepts for granted. Therefore any detailed discussions into the scientific soundness of these terms are omitted here.

(Topic 2-3) Granularity of User Profile

In Topic 2-2, we made our first mention of the term "granularity of the user profile". Up to now we have implicitly assumed that in Life Adaptive Computing, a single user profile is associated with each user, as is clear from its rightmost positioning in Figure 16. However, there is no logical reason to prohibit the assignment of multiple profiles to a single user, or of a single profile to a group of users. Put another way, user profiles in Life Adaptive Computing should be able to take any of the following three levels of "granularities": Coarse, Man-Size, and Fine Granularity (Figure 17).

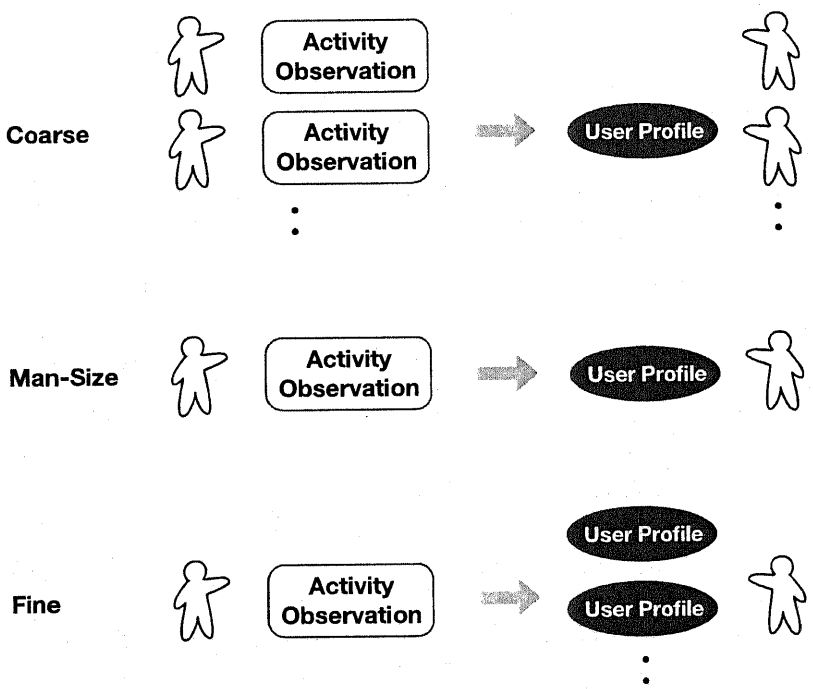

Figure 17: Coarse (top), man-size (middle), and fine (bottom) granularities

Coarse Granularity refers to the assignment of a single user profile to a group of users. In some Life Adaptive applications, it may be plausible to treat users who share a specific attribute (e.g. age, sex, social class, etc.) as having nearly iden-

tical characteristics, such that use of a common profile is justified. The major advantage of Coarse Granularity is because activity data from multiple users can be employed in shaping each profile, there are less concerns of crude, inadequate user modeling resulting from lack of activity data, compared to when assigning a separate profile to each user.

Man-Size Granularity is the standard granularity in Life Adaptive Computing, where a single user profile is associated with each user. Developers of Life Adaptive applications should first consider the use of this granularity, and only switch to others in the presence of a particular need or difficulty.

Fine Granularity refers to the assignment of more than one profile to each user. In general, a person's character is considered to be more or less plastic in relation to changing situations; for example, the same person can exhibit notably different traits in and out of the office. By adopting Fine Granularity and combining it with Context-Aware sensing technology, Life Adaptive applications can mold separate profiles for each of these different situations.

The above discussions on Topics 2-1, 2-2 and 2-3 leads us to the redrawing of Figure 8 as in Figure 18.

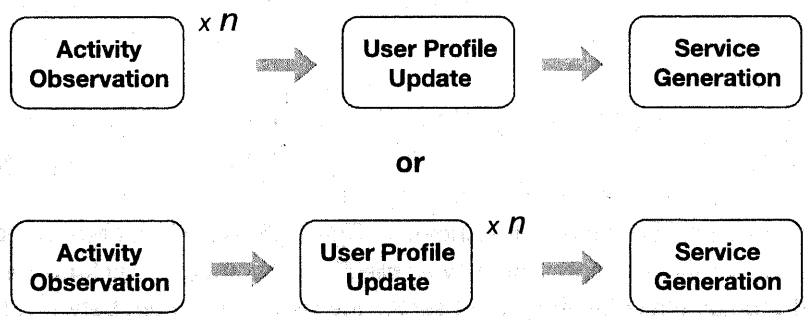

Figure 18: Basic procedure of Life Adaptive service delivery (5)

Service Generation

(Topic 3-1) Service Generation Techniques

The specifics of the Service Generation phase are generally application-dependent. Since Life Adaptive Computing is a concept leading to a number of applications serving diverse (possibly unlimited) purposes, techniques used for Service Generation could be equally diverse. Hence, coming up with good categorizations or general descriptions of the involved techniques is extremely difficult. But seen from another viewpoint, Service Generation can be regarded as a task of "finding the service that best matches the user profile" from "the collection of all possible service variations", which makes it similar to data mining problems.

Here we will explain two powerful filtering algorithms, popularly used to provide the core functionality in data mining applications. Due to the strong application-dependency of Service Generation, developers of Life Adaptive applications will have to design the details of this phase in a more or less ad-hoc manner. However, for many Life Adaptive applications, it should be possible to shape the specific techniques as modifications of the algorithms described here.

Content-based Filtering

The basic idea of content-based filtering [10][11] is to express the content of each data (in this case, service variation) and the user profile in a single format that allows direct comparisons, and pick out data which has minimal discrepancies with the user profile.

The most common implementation of this algorithm is using *feature vectors*. For instance, if we attempt to design a movie recommendation system which uses content-based filtering for Service Generation, the user profiles and the content of each movie may be expressed as shown in Figure 19 (Note that each movie and the user profile are represented by a collection of points, i.e. vectors). Here, Movie B should be recommended to User A, due to the similarities between the two feature vectors.

Content-based filtering is a robust technique applicable to a wide range of problem domains, on the one necessary condition that an effective format for objective content expression can be established.

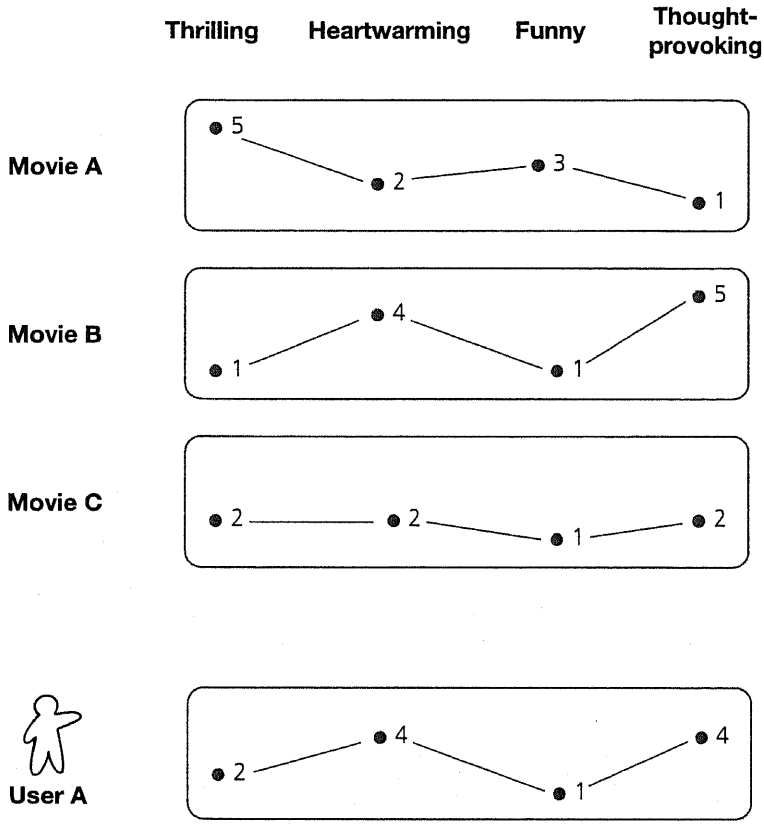

Figure 19: Content-based filtering

Collaborative Filtering

Collaborative filtering [12][13], as the name suggests, employs the profiles of a large number of users to perform filtering.

Suppose we attempt to design another movie recommendation system, this time using collaborative filtering for Service Generation. In this case, the profile will be expressed as a list, consisting of movies that the user had rated highly (i.e. expressed fondness) in the past (Figure 20). The algorithm will find other users with similar profiles (called *nearest neighbors*), and pick out data that are contained in the profiles of the nearest neighbors, but not in the user's profile. In Figure 20,

Movies B and D would be recommended to User A, since they have been rated highly by User C, who is a nearest neighbor of User A.

Collaborative filtering is known to be particularly effective for recommendation systems. Its main drawbacks are the need of a fairly large number of users, and the somewhat restricted problem domains to which it can be applied. There is a popular variation of the algorithm called item-based collaborative filtering, which is incorporated in CityVoyager, one of the applications described later in this thesis as case studies.

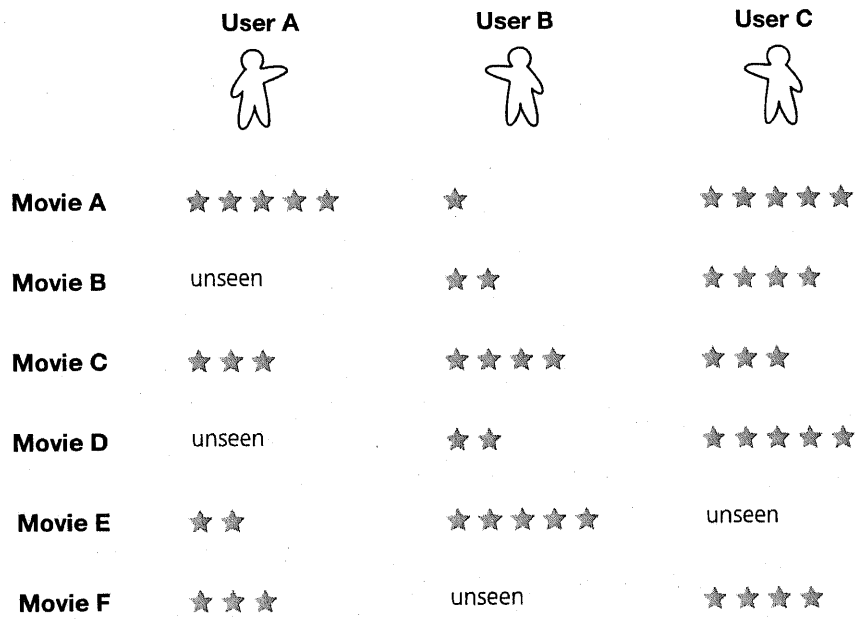

Figure 20: Collaborative filtering

(Topic 3-2) Use of Context Information

The chief advantage of Life Adaptive applications is their ability to adapt to long-term user properties. However, for most practical applications there would also be a need to make the application adapt to the *current* situation, i.e. real-time

context. This need can be fulfilled by introducing context information as a second input parameter (the first is the user profile) given to Service Generation. In this paper we will not describe exactly *how* context can be obtained (refer to previous work on Context-Aware Computing for details).

In topic 2-1 we described how it is possible to incorporate real-time context as part of the user profile. Treating context as an external entity independent of the user profile, as we have done here, should yield better results in terms of simplicity and manageability of the implementation.

Following the above discussions on Topics 3-1 and 3-2, Figure 8 can be redrawn as Figure 21.

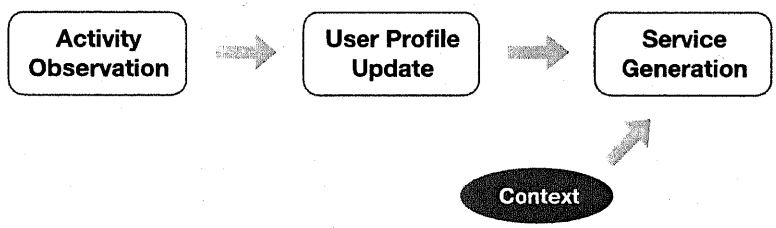

Figure 21: Basic procedure of Life Adaptive service delivery (6)

Now we have finished discussing all seven topics, which concludes our quest for establishing the conceptual framework of Life Adaptive Computing. One thing ignored in this framework is inter-phase relations: for example, choosing a particular activity for Activity Observation may lead to specific techniques for Service Generation being more effective or efficient than others. Comprehensive discussions on inter-phase relations are omitted on the ground that the current conceptual framework, with its detailed information about the individual phases, should make inter-phase relations sufficiently obvious to developers in actual Life Adaptive application developments.

Following all discussions provided in this chapter, Figure 8 can be redrawn as Figure 22.

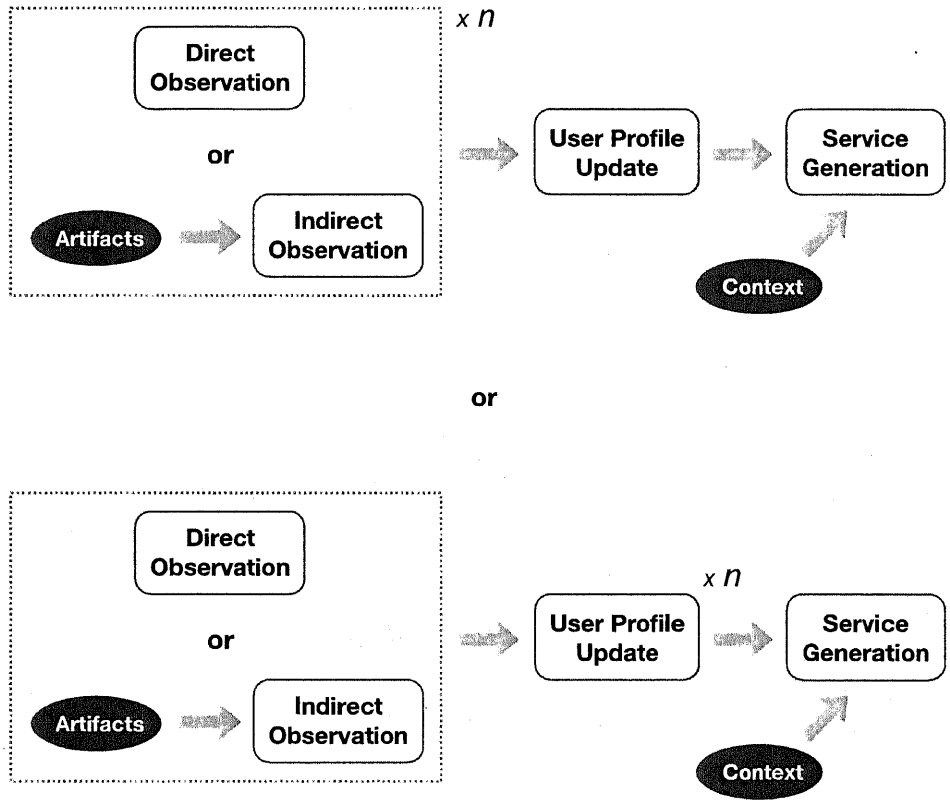

Figure 22: Basic procedure of Life Adaptive service delivery (7)

3 Case Studies Overview

In the upcoming chapters, two examples of Life Adaptive applications will be described. The descriptions will be lengthy and detailed spanning many minute specifics, to the extent that developers should be easily able to recreate these applications by following the descriptions.

The first application is called CityVoyager, an intelligent city guide system whose chief function is to provide shop recommendations to users, based on their past location data history (Figure 23).

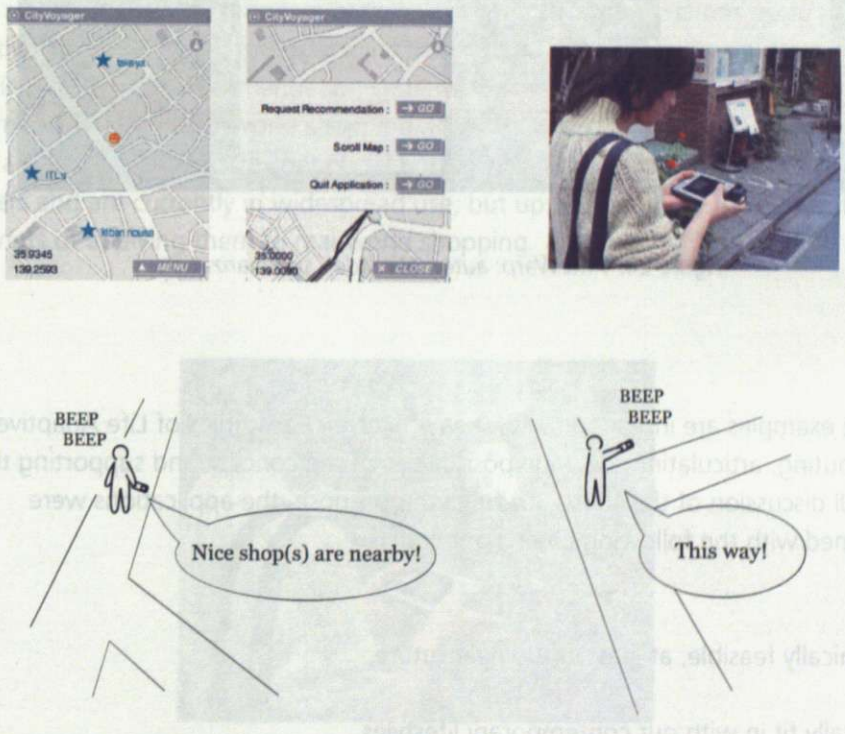

Figure 23: CityVoyager: the intelligent city guide

The second application is called TimeWarp, an automatic video summarization system which adapts to individual user's preferences by looking at his/her personal photo library (Figure 24).

Figure 24: TimeWarp: automatic video summarization

These examples are intended to serve as effective case studies of Life Adaptive Computing, articulating the wide possibilities of the concept and supporting the overall discussion of this thesis. To fulfill this purpose, the applications were designed with the following criteria in mind:

Technically feasible, at least in the near future

Naturally fit in with our contemporary lifestyles

Focus on innovative uses of "Life", nore mere visualizations

Employ a wide range of real-world activities

4 Case Study One CityVoyager: the intelligent city guide

Location-aware systems, which provide tailored services according to the user's current location, have long been a popular research topic. The most common type of location-aware systems is city guides [14], and with the growing popularity of handheld GPS receivers and GPS-embedded mobile phones, they are finally beginning to see widespread use. However, services offered by conventional location-aware city guides have mostly been limited to listing shops and restaurants in the user's nearby area, and the only way they could take into account each user's individual preferences was through manual parameter adjustments.

This chapter describes CityVoyager (Figure 25), an intelligent city guide system for GPS-equipped mobile devices which finds and recommends shops that match each user's preferences, and lets users navigate through those shops using an intuitive "metal detector" interface. Its basic idea is to apply the techniques of sophisticated item recommendation systems, used in online shopping sites such as Amazon.com, to real-world shopping. Recommendation systems in shopping sites are known to be highly beneficial both for customers and shopping site owners and are currently in widespread use, but up to now there have been few attempts of applying them to real-world shopping.

Figure 25: Using the CityVoyager system

The procedure for finding shops that match each user's preferences is based on the analysis of the user's past location data history, where a newly developed place learning algorithm is used that can detect users' frequented places by their proper names (e.g. "The Ueno Royal Museum"). The algorithm is computationally efficient and is also quick to converge, which is an important attribute since City-Voyager is concerned not with learning users' daily activities such as going to the office, but with learning shopping activities which are practiced only occasionally. Extreme care is given to refining the recommendation results, employing additional techniques such as prediction of user movement and consideration of the geographical conditions of the city, such as street layouts.

CityVoyager's acoustic "metal detector" interface makes it possible to use the system without having to pay constant attention to the device, as opposed to when using only GUI. The interface likens looking for shops that match the user's preferences to searching for metal objects using a metal detector. There is also an alternative "pointing" mode to the interface, which makes clever use of an electronic compass to provide users with more precise navigation.

A series of evaluation tests has been conducted at Daikanyama, one of Tokyo's most popular shopping districts. The results, reported in this chapter, validate the effectiveness of CityVoyager's overall approach.

4.1 Related Work

4.1.1 Location-aware systems

The earliest examples of location-aware systems, such as the Active Badge [15] and the ParcTab [16], were infrared-based systems intended for indoor use. More recent indoor systems realize much higher accuracy, by using Wi-Fi [17] or ultrasound [18]. All of these systems require the placement of signal emitters or receivers, throughout the interiors of the building where the system is used. In outdoor settings, GPS is the most common method for acquiring location, and is used in many city guide systems such as Cyberguide [14]. The GUIDE tourist guide system [19] uses Wi-Fi, which is another popular choice since many modern computing devices are equipped with built-in Wi-Fi capabilities. However, at the current moment there are few areas with a network of access points dense enough to allow robust location acquisition, which limits its usefulness. In a more recent example, the Place Lab project [20] has investigated the use of a combination of Wi-Fi, Bluetooth and GSM.

The most popular domain of location-aware applications is city/tourist guides, which list shops, restaurants and tourist attractions that exist near the user's current location. Many of these systems implement some level of user customization functionalities, but none matches CityVoyager in its ability to automatically, and dynamically adapt itself to detailed user preferences.

4.1.2 Place learning

Place learning, i.e. learning frequented places, has long been a hot research topic and several unique algorithms have been proposed. Marmasse and Schmandt's comMotion [21] identifies frequented places (buildings) by keeping track of positions where GPS signals became lost for an extended period of time. If signals were lost more than three times within a predefined radius, the system assumes the presence of a frequented place nearby. Ashbrook and Starner [22] uses k-means clustering of visits to find frequented places, where visits are defined as occasions where GPS signals were continuously lost, or where the speed of user movement was exceptionally sluggish. More recently, Liao et al. [23] proposed a much more mathematically sophisticated approach, using a hierarchical Markov model to learn users' movements and activities with high precision.

4.1.3 Recommendation systems

Recommendation systems have a wealth of academic examples, and have also found commercial success, as evidenced by the popularity of online item recommendation systems. Such systems, at their core, use either of the two major filtering algorithms, content-based filtering and collaborative filtering (these have already been described in this thesis at Chapter 2, Topic 3-1).

Attempts at applying recommendation systems to real-world shopping have been rare, with the only prominent example being the personal shopping assistant of Asthana et al. [24] But unlike CityVoyager, their system does not make use of the aforementioned filtering algorithms, which makes it unable to adapt itself to detailed user preferences.

4.1.4 "Metal detector" interface

CityVoyager's "metal detector" interface uses auditory cues to convey information in a nonobtrusive way, an approach which can be traced back to the various experiments in ambient display techniques [25, 26]. Also, the use of non-visual

information for GPS navigation has been preceded by Van Erp et al. [27], although their focus was on using vibrotactile feedback, and their system was targeted for traditional way-point navigation, not for city guide applications.

4.2 System overview

The chief functionality of CityVoyager is to first estimate each user's preferences from the history of his/her location data, and offer tailored shop recommendations upon request. Whereas recommendation systems for online shopping sites estimate user preferences from their past online activities, such as item purchases, CityVoyager estimates preferences based on real-world location history, collected while the user is shopping in the city. The recommended shops can easily be reached using the "metal detector" navigation interface, which allows users to find the recommended shops in the same way a metal detector would be used to find metal objects. CityVoyager is designed to be useful for various types of shoppers in various situations. For example, the system can help shoppers new to an area wanting to find shops that match their tastes, or shoppers more familiar to the area willing to try something new.

CityVoyager uses GPS to detect user location, which should be a reasonable choice given the current state of technology. However, it should be noted that CityVoyager's place learning algorithm can work with any kind of location detection technology, as long as each user's visits to shops can be detected reliably. This fact gives the system the potential to take advantage of future technical progress - for example, the only major change necessary when modifying CityVoyager to use Wi-Fi in place of GPS is to change the detection scheme of visits to looking for instances where user location had stayed roughly consistent throughout a stretch of time.

Figure 26 illustrates the system architecture of CityVoyager. The main hardware components are client devices carried by users, and a server that performs the actual recommendations. Any device, as long as it can connect to the Internet and can be equipped with GPS receivers (and ideally an electronic compass, for full implementation of the "metal detector" interface) can be used as the client device. The types of mobile computers most common today - PDAs, notebook PCs and mobile phones - are all able to fulfill this requirement. Considering the nature of the application, mobile phones would be the most ideal platform of these, especially since at this time adding GPS capabilities to mobile phones are becoming norm in some countries.

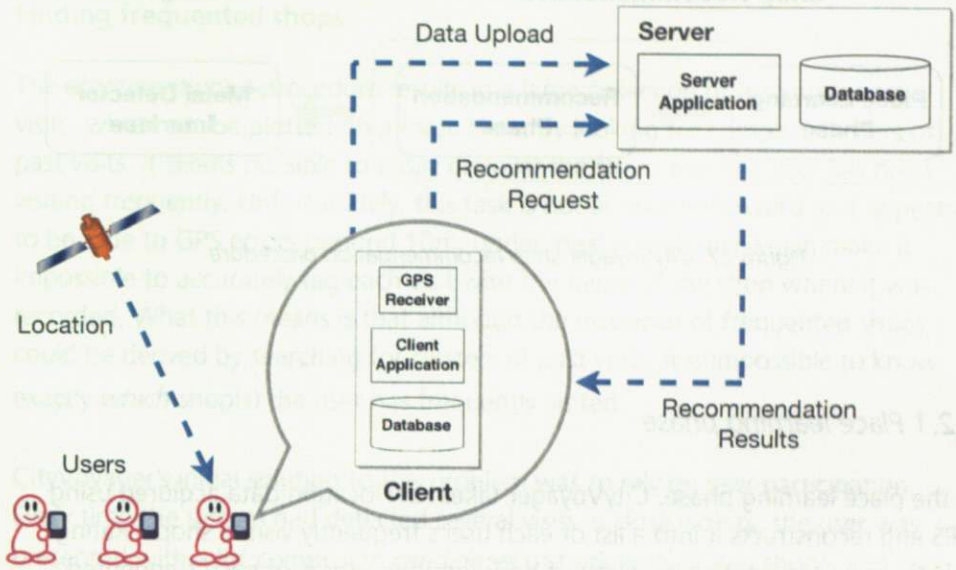

Figure 26: CityVoyager system architecture

The system analyzes raw location data acquired using GPS, transforming it into a list of each user's frequently visited shops, which is stored inside the server and used when making recommendations. If the client device has sufficiently high computational capabilities, this transformation of data can be done inside the client, and only the final list needs to be sent to the server. If that is not the case, the transformation will need to be done inside the server, which results in the need to send users' raw location data over the network. This introduces significant privacy risks, and should be avoided if possible.

Figure 27 shows the procedure with which the system offers its city guide service. First, the system picks out and recommends shops that match each user's preferences. This process consists of two phases, the place learning phase and the recommendation phase. After recommended shops are presented, users can navigate themselves through the shops using the "metal detector" interface. Note that while the place learning phase is a continuous process that is always active while the system is running, the other two steps, the recommendation phase and the "metal detector" navigation, will only be performed when the user has explicitly sent a request to the system ("Recommendation Request" in Figure 26). Below, each of these steps will be described in detail.

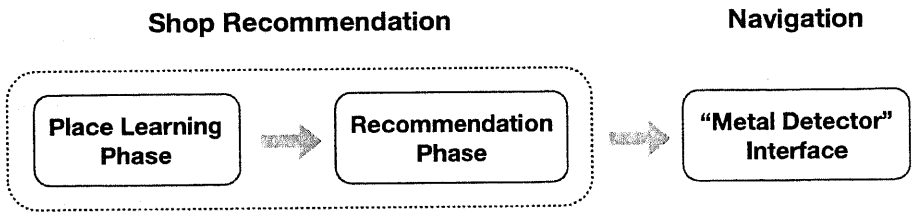

Figure 27: CityVoyager shop recommendation procedure

4.2.1 Place learning phase

In the place learning phase, CityVoyager takes raw location data acqiured using GPS and reconstructs it into a list of each user's frequently visited shops. Rating values, an approximate estimation of how fond the user is of each frequented shop, are also calculated automatically. This process can be further divided into two sub-phases: detecting visits to shops, and finding frequented shops.

Detecting visits to shops

CityVoyager treats the unavailability of GPS signals as evidence that the user has gone indoors. As GPS signals cannot penetrate through most building walls, visits to shops can be detected fairly accurately using this approach. However, since GPS signals can often become lost even when the user is outdoors (especially in urban areas where tall buildings easily block the signals), the possibility of false detections must always be kept aware of.

CityVoyager judges that the user has visited a shop when GPS signals have been continuously unavailable for a period of time longer than a threshold value, which in the current implementation is set to 5 minutes. The system then records the location of the visit, and also the length of time that signals had been lost, as the approximate duration of the visit.

The above method for detecting visits means that CityVoyager could only treat large-scale retail establishments, such as department stores and shopping malls, as one large shop instead of a collection of many small shops. There is currently no solution to this problem, except to use alternative technology that enables indoor location tracking such as Wi-Fi.

Finding frequented shops

The aforementioned procedure results in a large collection of the user's past visits, which can be plotted on a map. Here, by looking for compact clusters of past visits, it seems possible to easily discover the shops that the user has been visiting frequently. Unfortunately, this task is not as straightforward as it appears to be, due to GPS errors (around 10m, under ideal conditions) which make it impossible to accurately tag each visit with the name of the shop where it was recorded. What this means is that although the existence of frequented shops could be derived by searching for clusters of past visits, it is impossible to know exactly *which* shop(s) the user has frequently visited.

CityVoyager's initial solution to this problem was to rely on user participation. Every time the system had detected several visits in close vicinity, the user was presented with a list comprising *candidates* of frequently visited shops, and was asked to tell the system which of them he/she had actually been frequenting. However, simple user studies proved this to be too cumbersome for users, and the design was revised to make the system automatically estimate each user's frequented shops. In the process, a novel place learning algorithm, described below, was devised.

While previous algorithms take the approach of looking at the entire collection of past visits and trying to find prominent clusters, the CityVoyager algorithm focuses on a particular shop and the visits recorded nearby that shop, and tries to judge whether that shop was frequented by the user or not on a one-by-one basis. Therefore, the algorithm is able to give the proper names of users' frequented places (e.g. "The Univ. of Tokyo Hongo central cafeteria"), as opposed to previous algorithms which could only output latitude and longitude values. This difference is significant; CityVoyager relies on collaborative filtering for recommendation, which requires being directly able to compare one user's frequented shops with those of other users, and knowing proper names allows this direct comparison to be performed without any room for ambiguity.

Being able to detect frequented places with their proper names should be useful not just for CityVoyager but for a wide range of location-based applications, since most people are incapable of identifying a place with its latitude and longitude values. At the current moment, no other place learning algorithm seems to be able to achieve this function as seamlessly as the CityVoyager algorithm. Of course, it is easy to map latitude and longitude values afterwards with their

corresponding proper names, given an appropriate database. However, posterior mapping like this would likely lead to lower accuracy than the CityVoyager algorithm, which is designed to deal with proper names from the beginning.

As a prerequisite of the algorithm, the system needs a database containing the names and locations of the shops in the targeted area. This type of data is usually commercially available, and thus this requirement should not be a serious drawback for most urban areas.

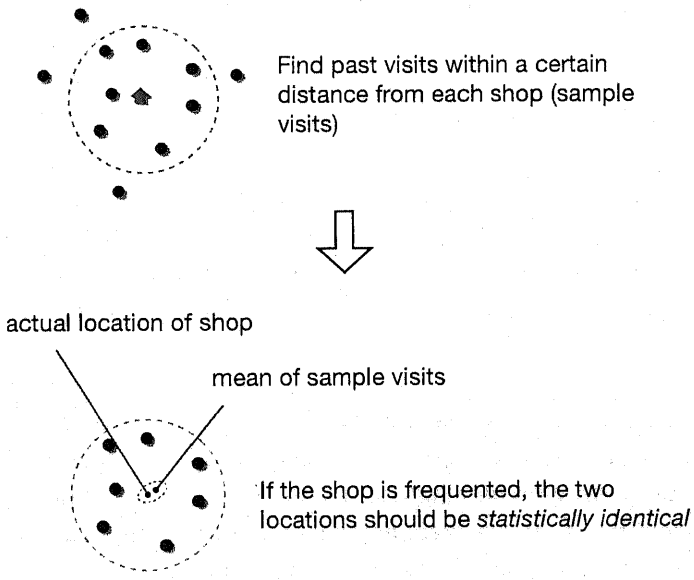

Figure 28: Place learning algorithm

The actual procedure of the algorithm is as follows. First, for each shop in the database, CityVoyager searches for all past visits within a predefined distance from the location of the shop. These are called *sample visits*. Then, by applying two-tailed *t*-test to the sample visits, CityVoyager tries to judge if it is plausible to believe that the shop has been frequented by the user or not. In other words, if the assumption that the shop has been frequented reasonably explains the past visits recorded nearby that shop, the shop is judged as a frequented one.

The algorithm is based on the assumption that the observed latitude and longitude values of visits to a particular shop will follow a normal distribution, with the actual shop location as the mean. Given this assumption, the latitude and longitude values of sample visits recorded around a shop will follow *t*-distributions. Thus, *t*-test can be used to evaluate if the means of the *t*-distributions are statistically identical to the actual shop location. If the shop is a frequently visited one, the two locations should be statistically identical (Figure 28).

If the two locations are found to be statistically identical, the shop is judged to have been frequented by the user, and is included in the user's list of frequented shops. Here, a rating value is calculated from the number of sample visits and the average duration of those visits, which is also written into the list. The list is stored inside the server and will be used when making recommendations.

The way rating values are calculated assumes that frequent visits to a particular shop indicate the user's strong fondness toward that shop. This should be valid in many cases, including the setting for the evaluations of CityVoyager where the system was used inside a small, urban area with little diversity of shop genres. However, in reality whether this assumption holds true or not should differ with shop genres, and parameters such as vicinity from the user's home or office should also be taken into account, as argued by Froehlich et al. [28]

As mentioned earlier, the place learning algorithm should preferably run inside the client device, so that there would be no need to send raw location data over the network, creating privacy risks. The low computational cost of CityVoyager's *t*-test based algorithm assures that this would certainly become possible in the near future with introductions of more capable mobile computers, if not at the current moment. Also note that the basic assumption of the place learning algorithm, that visits around a shop follow normal distributions, may not be appropriate in some cases, especially in architecturally crowded areas. This problem may be alleviated to a certain extent by using a more sophisticated probabilistic model, as is done in some state-of-the-art studies on place learning techniques. However, since the focus here is on learning shopping behaviors (not daily activities), the length of time available for learning is severely limited, which effectively negates the performance gains of using a sophisticated probabilistic model. As a result, in the case of CityVoyager the benefits of using a simple, quick-to-converge algorithm overweighs the drawbacks, especially considering its ease of implementation and computational efficiency.

4.2.2 Recommendation phase

Upon user request, the server offers shop recommendations using the lists obtained in the previous phase. The recommendation process consists of two subphases: filtering, and adding weights according to areas.

Filtering

Filtering of shops is done using the item-based collaborative filtering algorithm [29], a proven algorithm used in many existing online recommendation systems. First, similarities between shops are calculated using equation (1).

$$Sim(A,B) = \frac{\sum_u R_{u,A} R_{u,B}}{\sqrt{\sum_u {R_{u,A}}^2}\sqrt{\sum_u {R_{u,B}}^2}} \qquad (1)$$

Here, $R_{u,A}$ indicates the rating value for shop A by user u, and $R_{u,B}$ indicates the rating value for shop B by the same user u. The similarity between shops increases when there is an observed tendency that users who frequent shop A also frequent shop B. Since the collaborative filtering algorithm does not take into account the content of the data in any way, similarities can be defined even between shops of different genres, such as restaurants and clothing stores. The system picks out several shops which show the highest similarities with the user's frequently visited shops. These shops are regarded to have good chances of matching the user's preferences.

Calculating similarities does not need to be done in a short cycle. The similarities reflect users' long-term shopping activities, and it can be reasonably assumed that their changes within a short amount of time are subtle.

Adding weights according to areas

CityVoyager is for use in the city, which means there will be physical distances between the user and the recommended shops. In cities like Tokyo where many people shop on foot, the distances can easily become too demanding. Therefore, it must be made sure that the recommended shops would be easily accessible

from the user's current location. To meet this requirement, the city map is first divided into areas (Figure 29). Areas are defined so that any two points located in the same area are easily accessible from one to the other. The algorithm for this task, based on cluster analysis, is as follows:

```
1.  Divide the city map using a square grid. The grid
    should be fairly dense.
2.  Pick the grid elements which are located on places
    that can be walked through, such as streets, and
    define them as initial clusters.
3.  For all combinations of two clusters, do{
4.      For all combinations of two grid elements in the
        cluster, do{
5.          Calculate the distance between the two grid
            elements using Dijkstra's algorithm. Keep a
            record of the calculated distances.
6.      }
7.      Look for the two grid elements that give the
        largeest distance, and define their distance as
        the accessibility of the combination of clusters.
8.  }
9.  Merge the two clusters that combine to yield the
    smallest accessibility.
10. Repeat 3 ~ 9 until the number of clusters become
    sufficiently small.
```

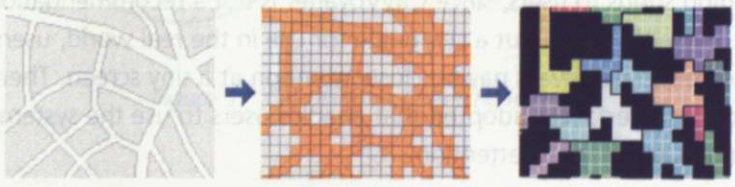

Figure 29: Definition of areas

The resulting clusters are chosen as the areas. Areas should be redefined anytime there are significant changes in the city landscape, such as openings of new streets, etc. Next, users' movements are modeled using a first-order Markov model, with these areas as nodes. Transition probabilities are calculated from periodically plotted user locations, where a higher probability indicates more chances of the user advancing to the area. The shops picked out by the filtering algorithm are weighted according to the areas in which they are located, with large weight values being added to shops in the same area as the user, or in areas with large transition probabilities. A few shops with the largest weights are picked out, and presented to the user as the final recommendations. This way, it is made certain that the shops recommended by CityVoyager can easily be visited from the user's current location.

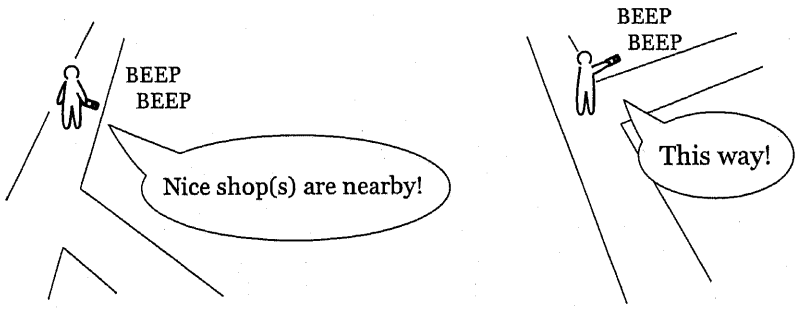

Figure 30: "Metal detector" interface

4.2.3 "Metal detector" interface

Shops picked out through the previous steps are shown using icons on a map, on the screen of the client device. While this alone fully serves the objective of recommending shops to users, since CityVoyager is not a recommendation system for online shopping but a city guide for use in the real world, users cannot be expected to be always paying close attention at a tiny screen. Therefore, it would help if an interface is adopted that enables users to use the system without requiring as much visual attention.

To this end, CityVoyager introduces a "metal detector" type interface. Whereas an actual metal detector detects metal objects, this interface detects shops that match each user's preferences. Repetitive beeping sounds are emitted when the user comes within a certain distance from a recommended shop, and the intervals between the beeps become shorter as the user moves closer (Figure 30, left). Since beeping sounds do not require the user to be looking at the screen to be noticed, the user can freely enjoy themselves in the city, only becoming notified when there is a shop nearby which may match his/her preferences.

In addition, a "pointing" mode has been added where users can "point" their mobile devices to find out the directions in which recommended shops may be located (Figure 30, right). In this mode, beeping intervals become shorter if recommended shops are located in the direction (±22.5°) in which the user is pointing the device. This function is realized by using an electronic compass, which is often embedded in high-end GPS receivers. Note that direction information acquired using an electronic compass is different from that acquired using normal (i.e., without an electronic compass) GPS receivers. The latter is the direction of user movement calculated from changes in user locations, and therefore is not necessarily equal to the direction in which the user is pointing the device. Since the "metal detector" interface needs to know which direction the device is being pointed to, and not the direction of user movement, an electronic compass-embedded receiver is required.

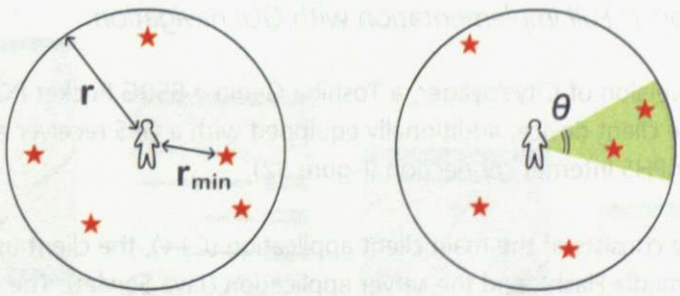

Figure 31: Calculation of beeping intervals

Beeping intervals are determined according to the distance to the closest recommended shop (r_{min} in Figure 31, left). If r_{min} is larger than a threshold value, no beeping sounds are emitted. Otherwise, one of three levels of beeping intervals is assigned, with lower intervals being assigned for smaller r_{min}. In the "pointing" mode, intervals are assigned by looking only at the recommended shops located in the direction in which the device is being pointed (Figure 31, right).

Although the "metal detector" interface is designed to be used in combination with a GUI screen, it may be possible to extend this idea to develop it into an effective navigation interface for visually handicapped people.

This concludes the description of how CityVoyager offers its city guide services. Several possibilities of future extensions of the system will be discussed at the end of the chapter.

4.3 Implementation

Two separate versions of CityVoyager have been implemented. In the first version, every function was fully implemented except for the "metal detector" interface. This version was developed to assess the accuracy of the new place learning algorithm, and to evaluate users' subjective satisfaction levels of the recommendation results. Evaluation of the "metal detector" interface was conducted using the second version, which was a wizard-of-oz system where recommended shops were actually chosen arbitrarily and hard-coded into the system beforehand.

4.3.1 Version 1: Full implementation with GUI navigation

For the first version of CityVoyager, a Toshiba Genio e 550G Pocket PC 2002 PDA is used as the client device, additionally equipped with a GPS receiver and a CF-type card for PHS Internet connection (Figure 32).

The software consists of the main client application (C++), the client user interface (Macromedia Flash), and the server application (Java Servlet). The main client application takes charge of the Place Learning phase, recording and analyzing each user's location data history. However, due to the relatively weak computational capabilities of the Genio, the place learning algorithm is actually run inside the server, and thus in reality the main client application only partially performs the Place Learning phase. As mentioned earlier this creates potential privacy risks,

and should be avoided if possible. The server application, in addition to running the place learning algorithm, performs recommendations upon user requests. Client-server communication is done over dial-up Internet connection. A MySQL database is installed in the server, which contains the names and locations of the shops located within a small area inside Daikanyama, Tokyo. The database also contains each user's list of frequented shops, and transition probabilities of the Markov model of user movement.

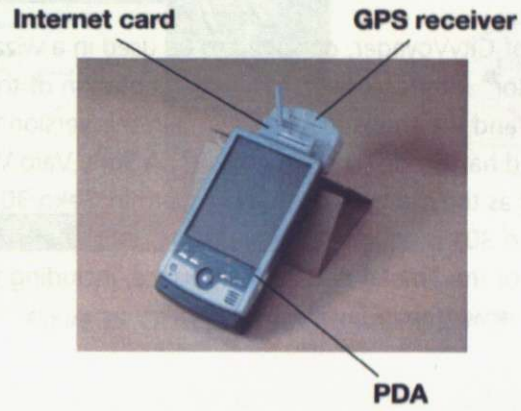

Figure 32: CityVoyager client hardware (version 1)

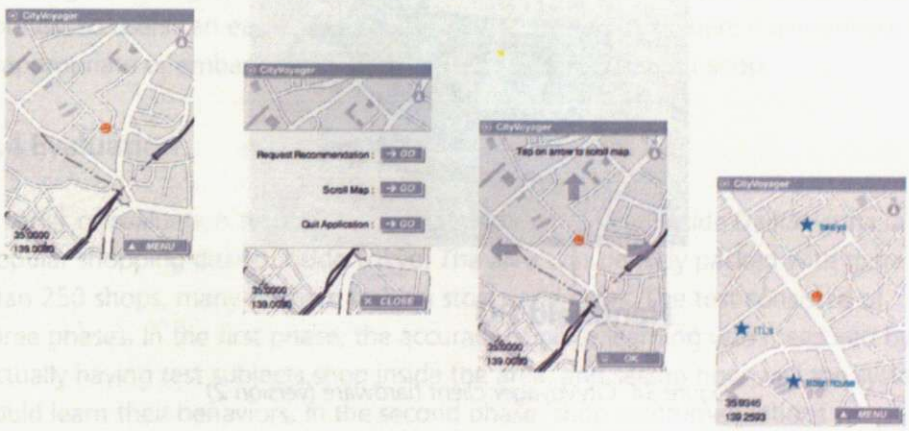

Figure 33: CityVoyager screenshots (viersion 1)

The user interface consists of several screens (Figure 33), each serving different functions to the user. Tapping on the bottom-right "menu" button on the default screen (Figure 33, far left) activates the menu (center left), from which users can send recommendation requests to the server by tapping again on "Request Recommendation". Recommended shops are first displayed in the form of a list, and their locations are shown on the map using star-shaped icons (far right). Note that the bottom-right button can take a number of different functions, depending on the situation (i.e. polymorphous).

4.3.2 Version 2: Wizard-of-oz system with "metal detector" interface

The second version of CityVoyager, designed to be used in a wizard-of-oz study of the "metal detector" interface, lacks full implementation of the shop recommendation function and the shops recommended in this version were actually arbitrarily chosen and hard-coded into the system. A Sony Vaio VGN-U71P handheld PC is used as the client device, with a Garmin Geko 301 GPS receiver (Figure 34). The Geko 301 is equipped with an electronic compass, which allows full implementation of the "metal detector" interface, including the "pointing" mode. The software is written in Java.

Figure 34: CityVoyager client hardware (version 2)

Figure 35 shows the main screen. A portion of the city map is displayed, where the current user location is indicated by a red dot and recommended shops are indicated by green dots. The map scrolls automatically, so that the current user location is always displayed at the center. Users can also manually scroll the map by clicking on the arrows, to search for recommended shops not in the vicinity.

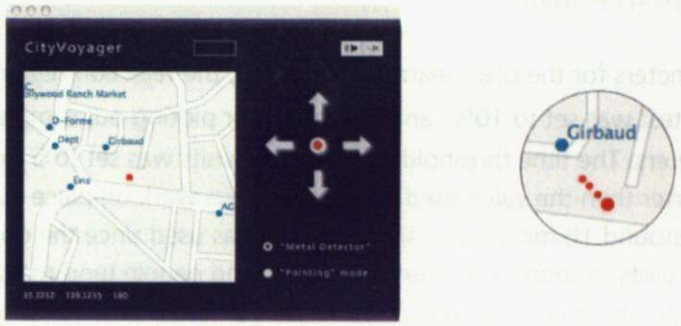

Figure 35: CityVoyager screenshots (version 2)

Users can switch between the normal "metal detector" mode and the "pointing" mode using radio buttons. In the "pointing" mode, the direction in which the device is currently being pointed is indicated on the screen using a row of tiny dots (Figure 35, right). The button on the top right enables/disables the beeping sounds, so users can easily turn off the sound in situations where making noise is inappropriate or embarrassing, such as when entering a quiet shop.

4.4 Evaluation

A series of evaluation tests was conducted at a small area inside Daikanyama, a popular shopping district inside Tokyo. The area was densely packed with more than 250 shops, many of them fashion stores and cafes. The test consisted of three phases. In the first phase, the accuracy of place learning was measured by actually having test subjects shop inside the area, and seeing how well the system could learn their behaviors. In the second phase, shop recommendations were given to users, and their subjective satisfaction levels were collected. Finally in the third phase, the effectiveness of the "metal detector" interface was assessed.

4.4.1 Place learning phase

10 users (ages 18–25, six male and four female) were asked to shop freely inside the test area at Daikanyama, with the CityVoyager client devices in their bags. The duration of the test was one or two sessions of half-day shopping, depending on the user. Due to the limited time available for the test, users were urged to visit as much shops as possible during the test. As a result a significant number of visits was collected, but to some extent this may have diverted users from their normal shopping behaviors.

As the parameters for the place learning algorithm, the rejection region of the two-tailed *t*-test was set to 10%, and the radius for picking out sample visits was set to 25 meters. The time threshold for detecting visits was set to 5 minutes, which is shorter than the value used in most previous work on place learning algorithms (around 10 minutes). A shorter value was used since the goal here was to learn visits to shops, not everyday places, and people tend to stay in shops for shorter periods of time compared to everyday places like home, office, etc. After the test, the (supposedly) frequented shops detected by CityVoyager were compared with the shops that each user had actually frequented.

An additional test was conducted employing one user (age 24, male) and two weeks of learning time, to see how the length of time used for learning affects the resulting accuracy. The additional test was conducted in the same area as the initial test, and with the same parameter values.

Initial test:		**Additional test:**	
Total number of shops detected by CityVoyager	: 17	Total number of shops detected by CityVoyager	: 11
Of which, shops actually frequented by the user	: 9	Of which, shops actually frequented by the user	: 8
Total number of shops frequented by the user	: 29	Total number of shops frequented by the user	: 24
precision =	53% (9 / 17)	precision =	73% (8 / 11)
recall =	31% (9 / 29)	recall =	33% (8 / 24)

Table 1: Place learning test results

Table 1 shows the results of the test. Two metrics, precision and recall, are used to measure the accuracy of the algorithm. Precision refers to the percentage with which shops detected as frequented by CityVoyager were actually frequented by each user, and recall indicates what percentage of users' frequented shops were successfully detected by CityVoyager. For the initial test of ten users, a precision of 53% and a recall of 31% were acquired. The additional test yielded a precision of 73% and a recall of 33%.

Discussions

A significantly higher precision (73%) was obtained for the additional test of two weeks, compared to that for the initial test (53%). From these results, it should be fair to say that the precision of place learning improves with the length of time used for collecting data. However, considering that CityVoyager is intended for learning users' shopping activities, which are not practiced daily but only occasionally, it is unreasonable to expect abundant learning time for all users (If a person goes shopping at a rate of two times per month, gathering two weeks worth of data would require seven months). Therefore in actual deployment, the precision may be more likely to stay closer to the results of the initial test (53%). Whether this is acceptable or not cannot be said immediately; it must be kept in mind that the objective of the system is not to achieve perfect place learning accuracy, but to offer effective recommendation. If the recommendations offered by CityVoyager prove to be beneficial to users, then the accuracy could be deemed adequate for the its purpose.

The recall value was low, both for the initial test of ten users (31%) and for the additional test (33%). This should mostly be due to the technical limitations of the GPS. As the test was conducted in a shopping district in Tokyo, notorious for being crowded with buildings, GPS signals were frequently lost even when the user was outdoors. This caused many visits to go undetected, since the system relies on the availability of GPS signals to detect the indoor/outdoor status of users. In the additional test, out of the 24 shops that had actually been frequented by the user, 12 of these shops had zero sample visits; In other words, visits were not recorded anywhere near those 12 shops, even though they were in fact reported by the user to have been frequently visited. For the other 12 shops where sample visits *were* obtained, eight of those shops were detected by CityVoyager, which yields a recall rate of 67%. These results imply that the fact that GPS was used for acquiring location had hindered the ability of the place learning algorithm, and the algorithm may produce higher recall values if alternative loca-

tion acquisition technology with better coverage than GPS were used, for example assisted GPS which is predicted to become common in future mobile phones.

Since none of the users who participated in the test lived close to Daikanyama, users could rationally be expected to visit shops such as fashion stores or cafes more often than everyday stores, such as grocery stores and drug stores. This may not always be the case in actual deployment, and the list of frequented shops for many users could end up being filled with everyday shops. Since frequent visits to a particular grocery store are unlikely to accurately represent the user's preferences in any way, failing to address this problem could result in disruption of the whole recommendation mechanism. One possible way to deal with this problem may be by not including everyday shops into user's lists of frequented shops, even when the system had detected one as being so. This way, frequent visits to everyday shops will simply be ignored and will have no effect on the recommendation results.

4.4.2 Recommendation phase

Using the activity data collected in the previous phase, shop recommendations were presented to users and their subjective satisfaction levels were solicited. Since the premise of CityVoyager is that it allows users to find shops that match their preferences more efficiently than without it, the results were compared with those from shopping without any type of external guides, where the same users were asked to freely walk around the city and visit shops that appear to match their preferences, judging from the shop's exterior, name, etc.

Two users (age 24, male and age 25, female), both from the group of ten users who had participated in the previous test, were asked to visit three shops for each of the above two methods (CityVoyager, and no external guides) and to give each shop a rating value in a scale of seven points. A scale of seven is commonly used in surveys, due to it being known that further increases in scale points do not significantly improve the reliabilities of subjective ratings. If CityVoyager is truly capable of offering effective shop recommendations, ratings for shops recommended by the system should be higher than those for shops visited without any external guides.

Table 2 shows the results of the test. It can be seen that the average rating value is higher for CityVoyager, than for shopping without external guides.

	CityVoyager	No guides
User 1	**1 3 7**	**5 4 3**
User 2	**4 6 3**	**2 6 3**
Average	**4.00**	**3.83**

Table 2: Recommendation test results

Discussion

A higher average rating was acquired for shops recommended by CityVoyager than for shopping without external guides, although the difference is not statistically significant due to the small number of users. It should be noted that in one of the recommendations provided by CityVoyager, a shop, which sells fashion items for women, was recommended to a male user (user 1) and subsequently received a rating of 1. Obvious mismatches like this could have been easily avoided by screening the recommendation results according to basic information about the user, such as age or sex. Had a simple screening mechanism like this been implemented in the system before the test, the final results may have been even more inclined in favor of CityVoyager.

4.4.3 "Metal detector" interface

To see how the "metal detector" interface compares with traditional GUI-only navigation, a wizard-of-oz study of the interface was conducted using the second implementation of CityVoyager. Although in this version the recommended shops had been arbitrarily chosen and hard-coded beforehand, users were told that the shops had been automatically selected by the system to reflect their individual preferences.

A total of 8 users (age 23–30, six male and two female) participated in the test, who were asked to freely walk around and shop inside the test area with the client devices in their hands. The users were not forced to visit the recommended

shops, and instead were told to enjoy shopping as they usually do when they come to the city, and only follow the recommendations when they felt like doing so. The test lasted approximately one and a half hours for each user. Users were asked to use the "metal detector" interface for half of the duration of the test, and use GUI-only navigation for the other half. The order in which users were told to use the two interfaces was switched with each user: four users were told to use the "metal detector" interface first, and the other four were told to use GUI-only navigation first. This was done to neutralize the effect that the order of use may have on the results of the test.

Familiarity with the test area differed greatly with the user. User 1 said this test was his first visit to Daikanyama, while several other users claimed that they have often come and shopped in the area.

A team of test administrators (including the author) followed the users and observed their behaviors throughout the test, although we tried to be careful not to follow users too closely. We wanted to have users actually enjoy shopping in the area to gather their honest opinions about the system, and we were afraid that close stalking might prevent them from doing so. Also, the system kept a record of all GUI interactions during the test. After the test, users were asked to answer questionnaires shown in Table 3, and to provide comments.

From the test, it was found that users scrolled the map (i.e., pressed one of the arrow buttons on the main screen, see Figure 35) almost eight times as more often when using GUI-only navigation, compared to when using the "metal detector" interface (31 scrolls on average for GUI-only, four scrolls on average for "metal detector" interface). It was also found that users visited recommended shops more often when using the "metal detector" interface (four visits on average for GUI-only, six visits on average for "metal detector").

Table 4 shows the results of the questionnaires. It can be seen that the "metal detector" interface received better (higher) average scores for question 1 and 3, while GUI-only navigation received better average scores for questions 2, 4, and 6. The average scores for question 5 were the same for both interfaces.

Discussion

The fact that users scrolled the map eight times as more often when using GUI-only navigation, combined with the scores obtained for question 1 where users

Please answer each of the six questions listed below in a scale of seven points, both for the "metal detector" interface and GUI-only navigation.

Q1. While using the system, how much attention do you think you were paying to the screen?
(1: most attention) - (7: least attention)

Q2. To what extent did using the system prevent you from being aware of your surroundings?
(1: most prevention) - (7: least prevention)

Q3. To what extent did using the system add to the "fun factor" of shopping?
(1: greatly reduced) - (7: greatly added)

Q4. To what extent are you willing to consider this system for everyday use?
(1: not willing at all) - (7: very willing)

Q5. To what extent were you able to smoothly decide your shopping routes with confidence?
(1: not smooth at all) - (7: very smooth)

Q6. To what extent, do you think, did using the system cause you to appear strange or geeky to other people in the city?
(1: very strange) - (7: not strange at all)

Please answer the following question, regarding the use of the "metal detector" interface.

Q7. To what extent did the auditory cues serve as a hint of the existence and the locations of the recommended shops?
(1: did not help at all) - (7: were very helpful)

Table 3: Questionnaires

"metal detector"

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
user 1	5	6	4	2	4	2	3
user 2	1	3	3	4	1	2	5
user 3	3	3	6	5	7	3	3
user 4	5	5	4	7	7	7	6
user 5	5	5	2	1	4	1	3
user 6	1	2	5	5	3	2	5
user 7	2	3	6	2	5	7	6
user 8	4	5	6	4	5	4	5
average	3.3	4	4.5	3.8	4.5	3.5	4.5

GUI-only

	Q1	Q2	Q3	Q4	Q5	Q6
user 1	3	5	3	6	4	3
user 2	3	4	3	5	1	3
user 3	5	6	4	2	7	4
user 4	2	7	4	6	6	7
user 5	3	4	3	3	3	2
user 6	1	3	7	5	5	4
user 7	2	2	4	2	5	7
user 8	3	5	4	3	5	5
average	2.75	4.50	4.00	4.00	4.50	4.38

Table 4:Questionnaire results

answered they paid less attention to the screen when using the "metal detector" interface, implies that there was a drastic difference in the styles in which the two interfaces were used. Presumably, while users looked for the locations of the recommended shops through manually scrolling the map when using GUI-only navigation, when using the "metal detector" interface they could instead learn about the recommended shops through auditory cues which could provide a more peripheral awareness of the existence and locations of the recommended shops, and hence they had to pay less attention to the screen and performed less GUI manipulations. The relatively high scores for question 7 show that the auditory cues of the "metal detector" interface served as a fairly helpful hint in finding the recommended shops.

The result that users tended to visit recommended shops more often when using the "metal detector" interface seems to support this view, since being able to notice the existence of the recommended shops without having to be constantly looking at the screen should quite possibly lead to less chances of users missing a recommended shop when they come near one. However, it must be noted that the density of recommended shops were not homogeneous within the test area, and the fact that users visited more recommended shops may just mean that they happened to be in regions that were relatively dense with recommended shops by chance when using the interface, and may not necessarily reflect the effectiveness of the navigation method.

Upon hearning users' answers for question 1 that they paid less attention to the device screen when using the "metal detector" interface, it seems natural to expect that the "metal detector" interface would also enjoy a better average score for question 2, where users were asked the extent to which they were able to be aware of their surroundings while using the system. The logic is that, if users had to pay less attention to the screen, obviously they would have been better able to focus on the surrounding sights and sounds in the city. Surprisingly, however, GUI-only navigation actually received a better average score for this question. Users commented that they felt the auditory cues actually forced their attention to the device at times, contrary to the intention of its design. The sound itself was carefully synthesized so as not to be too alarming to users and to be able to convey information at the periphery of user attention, but in areas fairly dense with recommended shops, beeping sounds were emitted ceaselessly in short intervals, and the sound stopped being peripheral and started to draw full attention of users. Hence, although users did not have to be constantly looking at the screen, they were sometimes irritated by the auditory cues and were not able to focus on their surroundings as much as had been expected.

This problem of the auditory cues becoming too alarming at times apparently was not only attention-consuming but also embarrassing to users, as they pointed out in their comments. This may have been the reason behind the low scores the "metal detector" interface received for questions 4 and 6, where users were asked how much they are willing to consider the system for everyday use, and if they thought using the system made them appear strange/geeky to other people in the city.

It may be possible to solve this problem by relying less on changing beep intervals by lowering the rate in which intervals are reduced, so that beeps are never emitted too frequently, and instead using different tones of beeps to denote the distances from the recommended shops. Another possible solution is using vibration in place of sound.

Looking at the scores for question 3, in which users were asked the perceived "fun factor" of the interface, users appear to have had more fun when using the "metal detector" interface than when using GUI-only navigation. In the comments, one user referred to the added fun by likening searching for recommended shops using the "metal detector" interface to searching for "treasure boxes" using a metal detector. On the whole, while the unpolished implementation of the acoustic cues have led to some negative feedback from users, it seems fair to say that the "metal detector" interface has succeeded in providing nonobtrusive and intuitive navigation, with an added entertainment value.

Since no rules were imposed and no suggestions were given on when to use the normal "metal detector" mode and when to use the "pointing" mode, the users seemed to be using these two modes in a variety of ways. One user claimed that he used only the "pointing" mode for the entire duration of the test, while another user said he used the "pointing" mode first to get a rough idea of which way he should go, and then switched to the normal "metal detector" mode to let the system notify when he is close to a recommended shop. From observations, users appeared to be often switching to "pointing" mode when waiting for traffic signals.

Overwhelmingly positive comments were received about the benefits of the shop recommendation feature, regardless of the navigation interface used. Especially, several users who had claimed to be relatively unfamiliar with the test area (including user 1, who said that this test was the first visit to Daikanyama) praised how using the system had helped them enjoy shopping in a new town.

Let us finish the discussion by pointing out the advantages of using CityVoyager compared to conventional location-aware city guides. Conventional location-aware systems list shops that are closest to the user's current location, which should be helpful in areas where shops are relatively sparse. But in shopping areas like Daikanyama there tend to be a huge number of shops within walking distance, and presenting shops closest to the user will not be so much of a help. CityVoyager can make more effective recommendations in such cases, by narrowing down the search space based on individual preferences. Of course, many conventional systems allow some level of manual user customization, such as letting the user specify which types of shops he/she is interested in. However, the extra user interaction can easily become a nuisance, and furthermore, past studies on collaborative recommendation systems have found that using static categories like shop types are unable to account for detailed user preferences. Also, since most conventional location-aware systems use straight-line distance to determine the "closeness" of shops, there will be instances where a shop that has been presented as being close by the system actually is not close at all, depending on the layouts of city streets. Such problems may be rare in cities with clean street layouts like Manhattan, but will often be encountered in cities with complicated, disorderly layouts like Tokyo. CityVoyager takes street layouts into account when determining the "accessibility" of a shop, so the shops presented by the system is guaranteed to be easily accessible from the current location of the user.

4.5 Summary

This chapter described CityVoyager, an intelligent city guide system which finds and recommends shops that match each user's preferences, and lets users navigate through those shops using an intuitive "metal detector" interface. Evaluation results validate the overall approach, including the accuracy of the place learning algorithm, the appropriateness of the recommendation results, and the effectiveness of the "metal detector" navigation interface. There are many possible extensions to the system. For example, it may be possible to modify the system to use additional context information, such as the time of day or walking distance, to provide more timely recommendations. Restaurants could be recommended when the recommendation was requested around noon, or cafes when the user has walked long distances. Another possible extension is using time tables of events or movies. By combining these information with location data history, it may be possible to analyze what type of events or movies are preferred by each user, which can be used to offer more precise recom-

mendations. For example, the system could recommend a movie theater, when it is showing movies that are likely to match the user's preferences.

5 Case Study Two TimeWarp: automatic video summarization

Home video production has soared in popularity over recent years, thanks to decreases in camera prices and the emergence of online video sharing services such as Youtube. This has led to increased demand for quick and easy video editing methods, which require minimal effort on the part of the user. Existing solutions include beginner-friendly video editing software such as iMovie, and web-based video editing services like eyespot and jumpcut. However, although these systems offer simple video editing and provide helpful functions such as adding fancy visual effects, they are not capable of automatically analyzing video content, and thus are unable to produce high-quality results without extensive user participation in the process. Automatic video summarization systems, whose aim is editing and summarizing video to modest lengths while retaining its content with little or no explicit user manipulation, would be a more ideal solution. However, although considerable progress has been made in this area, the results are still unsatisfactory for practical use, which can be attributed, at least partly, to the conventional approach of trying to find a single set of summarization rules that apply to all users, not taking into account differences in user preferences.

This chapter describes TimeWarp, a user-adaptive video summarization system which takes into account individual preferences by analyzing the contents of each user's personal photo library. Nowadays it is common practice, especially among people of younger generations, to store thousands of photos inside their PCs, taken using their digital cameras. These personal photo libraries contain rich information about the users' tastes, personalities, and lifestyles, and due to the various similarities in characteristics between video and still photos, it may be possible to infer each user's preferences on video summarization from these libraries. The system uses image classification techniques to determine "important" sections of the movie, defined as sections that show contents or themes which frequently appear in the photos in the user's personal photo library. For example, if a user has many pictures of a cat inside his/her library, video sections showing cats would be considered "important". Summarization is performed by deleting "unimportant" sections and reconstructing the video, while taking care not to overly disrupt the continuity of the video.

A prototype of the system has been implemented, with which a series of evaluations were conducted to assess its effectiveness. The results show TimeWarp's

potential to serve as a powerful automatic/semi-automatic video summarization solution.

5.1 Related work

The central task in video summarization is identifying the "important" or "exciting" sections. This is relatively easy when the movie belongs to a domain with a fairly rigid and known structure. For example, Ju et al. [30] proposed a system for summarizing videos of conference presentations, using the intensity of the presenter's movements to find the "important" sections. In another example, Yow et al. [31] developed a system which identifies "exciting" sections of soccer match broadcasts by tracking the position of the ball in relation to the goals. Summarization systems which are not targeted towards a specific domain, or targeted towards a domain without a clear structure, are more difficult to develop. The Informedia [32] system is an example of such a system, which mainly relies on speech recognition and analysis. More recently, Ma et al. [33] and Mei et al. [34] have adopted psychological theories to video summarization, to capture viewers' anticipated attention levels or the original intentions of the person who had shot the video. In spite of the various techniques used, most past video summarization systems share the same approach in that they try to define a single set of summarization rules that fits all users, and have generally been indifferent to users' individual preferences. Some systems have tried to adapt to each users' personal tastes and needs by introducing user-adjustable parameters [35], but parameters for video summarization are generally difficult to understand for people who do not have deep knowledge of computer vision.

TimeWarp uses image classification techniques to determine the "importance" of video sections. Image classification is a relatively new area of computer vision research, concerned mainly with annotating images with words that correctly describe their contents. Mori et al. [36] proposed a technique to automatically learn correlations between image features and words using a training data set of already annotated images, which are then used to annotate new images with words. More recent approaches apply image segmentation techniques such as Blobworld [37] to image classification [38], or use multiresolution analysis of images [39], to achieve improved accuracy.

5.2 System overview

Figure 36 illustrates the procedure with which TimeWarp performs video summarization. The user's photo library is first sent through the photo library categorization process, where photos in the library are grouped into several major categories by their contents. In the video segmentation process, the input movie (the movie to be summarized) is divided into shots, which are further divided into short segments. Next, in the image classification process, key frames extracted from the segments are analyzed and judged if their contents fall into any of the major categories found in the user's library. "Important" sections of the movie are determined through these judgments. Finally, in the summarization process, the relatively "unimportant" sections are deleted, and the remaining sections are reconstructed into a coherent summary of the original movie.

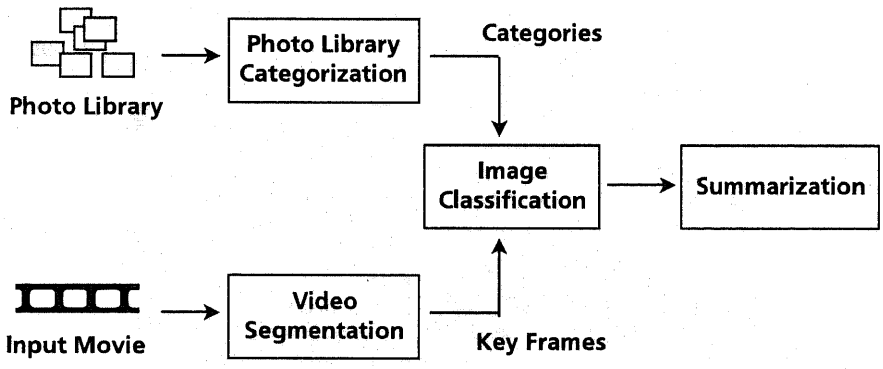

Figure 36: TimeWarp video summarization procedure

5.2.1 Photo library categorization

The objective of the photo library categorization phase is to group photos in the user's library into several major categories, according to their contents. Ideally, this process should be done fully automatically, but since it is impossible to achieve high accuracy using current technology TimeWarp requires users to manually take part in the categorization. However, since the objective of the system is to offer a quick and easy method for editing video, the user's burden

should be reduced as much as possible, even if risking some loss of accuracy. To this end, TimeWarp incorporates the following procedure.

Categorization is done by having the user slide photos inside a window (Figure 37), so that photos with similar contents are located close to each other. Since a personal photo library can contain thousands, or tens of thousands of photos, it is unreasonable to require the user to categorize every single photo in the library. Instead, the system randomly picks out photos from the user's library, and asks the user to categorize (by sliding photos) this *subgroup* of the library, consisting of around several hundred photos at most. Since the objective here is not to achieve a complete categorization of the library but to find only the most significant categories, using a subgroup of limited size should be acceptable.

Figure 37: Photo categorization interface

When the user is finished with the categorization and presses the "cluster" button, the system applies standard hierarchical clustering to group photos that are located closely on the screen (Figure 38). These clusters are regarded as the categories of the photo library. Since only the major categories is needed here, a limit is introduced to the number of categories that can be defined (eight in current implementation). Each category is given a name of "Category i" (i is a

number starting from 0). Category names can be changed freely, like for example "Flowers", "Cats", or "Architecture".

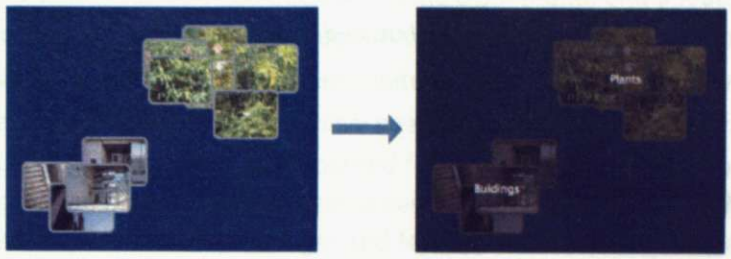

Figure 38: Clustering photos

The decision to use the type of interface described above instead of the common file and folder metaphor (where grouping photos is done by dragging photos inside folder icons) is due to the fact that this type of interface is better at supporting trial and error, since dividing and merging existing groups can be done more easily. In addition, the interface resembles a real-world situation where people group photos (or any other kind of paper documents) by moving them on a table, and thus should be intuitive for most users.

The photo library categorization process does not need to be conducted every time the system performs a summarization. It is necessary only when there is a significant change in the user's photo library, such that adding a new category or deleting an existing one is required to correctly reflect the characteristics of the library. In most cases, adding or removing several photos from the library will not create the need for a new categorization.

5.2.2 Video segmentation

First, the original movie is divided into *shots*, through the common method of looking for sudden changes in color histograms. Each shot is further divided into *segments* by splitting the shot using a fixed time interval (one second in current system). The first frame of each segment is picked out as the key frame of the segment (Figure 39).

Figure 39: Shots and segments

5.2.3 Image classification

Using image classification techniques, the key frame of each segment is judged if it belongs to any of the categories that had been defined in the photo library categorization process. For this task, we use the classification algorithm proposed by Jeon et al. [40], slightly tweaked to suit our particular task. The following equations are used to determine $P(\omega|b_1, ..., b_9)$, the possibility that a key frame containing regions b_1 through b_9 belongs to category ω.

$$P(\omega|b_1, ..., b_9) = \frac{\sum_{J \in T} P(J)P(\omega, b_1, ..., b_9|J)}{P(b_1, ..., b_9)} \quad (2)$$

$$= \frac{\sum_{J \in T} P(J)P(\omega|J)\prod_{i=1}^{9} P(b_i|J)}{P(b_1, ..., b_9)} \quad (3)$$

where

$$P(\omega|J) = (1 - \alpha_J)\frac{\#(\omega, J)}{|J|} + \alpha_J \frac{\#(\omega, T)}{|T|} \quad (4)$$

$$P(b|J) = (1 - \beta_J)\frac{\#(b, J)}{|J|} + \beta_J \frac{\#(b, T)}{|T|} \quad (5)$$

For each key frame, $P(\omega|b_1, ..., b_9)$ is calculated for all categories that had been defined by the user in the library categorization process, which are compared to see which category the content of the key frame falls into. $P(\omega, b_1, ..., b_9|J)$ denotes the probability that photo J of training set T (in this case, T is the

subgroup of the user's photo library that the user had classified) belongs to category ω, and at the same time contains regions b_1 through b_9.

Regions are determined as follows. First, every photo in the training data set T is divided into 9 *sub-images* of identical size using a 3 x 3 grid, and a feature vector (a vector consisting of HSV color histogram values and Tamura texture descriptors) are calculated for each sub-image. Each feature vector is plotted in the *feature vector space*. After joint histograms have been calculated for every photo in T, k-means clustering is applied to the plotted feature vectors, and each formed cluster is given an index number. When a new key frame is to be classified, the key frame is divided into 9 sub-images and feature vectors are calculated, in the exact same manner as the photos in the training data. Then, for each of the sub-images, the feature vector of the sub-image is compared with the means of the aforementioned clusters, and the index number of the closest cluster is assigned to the sub-image. Since there are 9 sub-images for each key frame, this results in 9 index numbers being assigned to each key frame. These index numbers denote the regions contained in that key frame. Figure 40 illustrates this procedure.

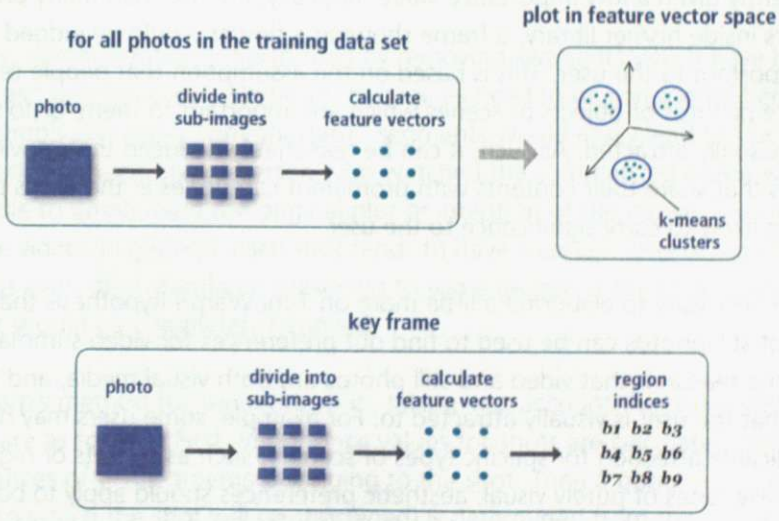

Figure 40: Determining regions

The above procedure for determining regions is a standard one used in many existing image classification systems. Some recent researches incorporate sophisticated image segmentation techniques such as Blobworld [37], instead of using a simple grid as is done here. The use of a simple grid was chosen primarily due to the fact that since the system is dealing with video here and must classify a large number of key frames, there is a serious need to reduce calculations.

Parameters α_J and β_J are set to 0.1 and 0.9 respectively, using the original settings of Jeon et al. $P(J)$ is flat for all photos. $\#(\omega, J)$ takes 1 if photo J is of category ω, and 0 if otherwise. $\#(\omega, T)$ is the number of photos belonging to category ω in the training data set. $\#(b, J)$ is the number of times region b appears in photo J, $\#(b, T)$ is the number of times region b appears in the training data set.

After $P(\omega|b_1, ..., b_9)$ has been calculated for every category, an importance value is assigned to each segment, where high values are assigned when the key frame can be judged as belonging to a category (i.e. the value of $P(\omega|b_1, ..., b_9)$ for one of the categories is significantly higher than those for other categories). Note that a key frame does not necessarily have to belong to a category. If there is no single category for which the value of $P(\omega|b_1, ..., b_9)$ is exceptionally high, the segment is judged as not belonging to any category ("unclassifiable") and is consequently given a low importance value. Basically, if a user had many photos of flowers inside his/her library, a frame showing a flower would be judged as being important to the user. This is based on the assumption that people tend to take more pictures of objects or scenes which are important to them, or to which they are visually attracted. As such, it can be reasonably deduced that movie segments that share their contents with prominent categories in the user's photo library are likely to carry significance to the user.

It may be necessary to elaborate a little more on TimeWarp's hypothesis that libraries of still photos can be used to find out preferences for video summarization. One reason is that video and still photos are both visual media, and both reflect what the user is visually attracted to. For example, some users may have an inexplicable affection for specific types of scenery, such as sunsets or night city views. These types of purely visual, aesthetic preferences should apply to both still photos and video. Another, more important reason is that a user's photo library offers a glimpse into that user's daily life. There was a time when taking photos was something only done at special occasions, such as weddings or vacations. However, many users of recent compact digital cameras embrace a

completely different style of taking photos: taking along cameras everywhere, everyday, and recording any-thing that has caught the slightest attention. Such a user's personal photo library will inevitably contain rich information about the user's daily life, and by taking a close look at the photos, a lot of things can learned about the user, about his/her lifestyle, values, and preferences. This can be understood by visiting a photo sharing website such as Flickr, and looking through streams of photos uploaded by a single user. If many photos of the same cat can be found in the stream, it can reasonably be assumed that the user is living with the cat, and the user's affection for the cat will seem obvious. If a stream contains many photos of a small boy, the user may be assumed to be the boy's parent, and the user's emphasis on family values can be understood. A stream of photos con-taining various photos taken at mountains point to a specific lifestyle and values, while a stream containing many photos taken at nightclubs point to another. By analyzing a user's photo library, TimeWarp is trying to capture not just the user's tastes on photos, but the user's values and preferences that apply to the user's life in general. If the fact that a user loves a cat has been found out, effective video summarization may be possible by emphasizing sections which show a cat, and de-emphasizing sections which do not.

5.2.4 Summarization

Basically, summarization is performed by removing segments which have been judged as "unimportant". However, two issues need to be considered here. One is that simply removing "unimportant" segments would easily lead to fracturing the video too much, that a person who watched the summarized movie would be unable to understand the original plot or intention of the movie. Two is that in home videos in general, each shot tends to have a certain level of significance, and thus every shot should be preserved to some degree in the final movie and no shot should be completely removed.

TimeWarp's method for summarization, taking into account the issues described above, are as follows. First, importance values for shots are calculated from the importances of the segments belonging to the shot. Then a quota (i.e. the length to which the shot will be shortened) is determined from the importance and the original length of each shot. Shots with low importance values will generally have shorter quotas, and will be shortened more drastically. Shortening of shots is done so that the *continuity* of the shot is preserved as much as possible; the system selects a continuous strip of video within the shot, which is

no longer than the assigned quota and contains segments with the largest importance values. Figure 41 illustrates the above procedure.

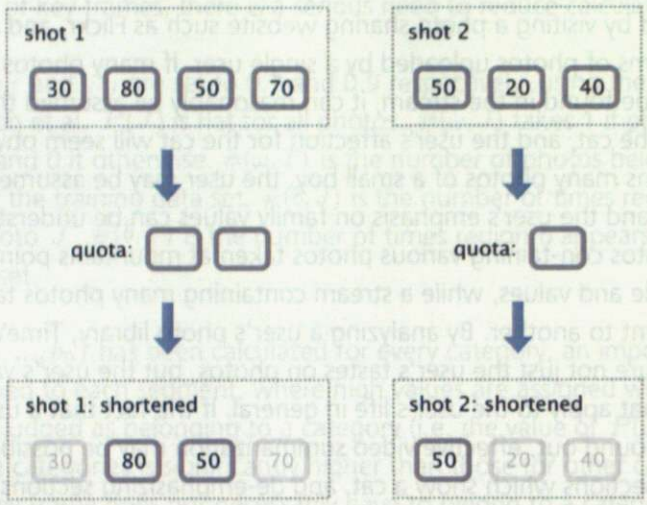

Figure 41: Summarization procedure

The user is allowed to set the "target time", to which the system aims to shorten the original movie. The sum of quotas for all shots will be equal to the target time. When the user increases the target time, the quota for each shot will also increase, but the ratio of quota lengths between shots will remain unchanged.

While not incorporated in the current system, adding animated transition effects such as fadeouts is considered for future implementation. Fancy visual effects like these are often adopted in home videos, and incorporating them into the system may help improve the perceived quality of the outputted movie.

5.2.5 Semi-automatic summarization

So far this chapter has described TimeWarp as a fully automatic video summarization system, which requires no explicit user manipulation other than the photo library categorization part. However, the current implementation also allows numerous manual adjustments, which can be exploited by users who want more control over the summarization results. For example, users are allowed to add/remove shot boundaries, change importance values of shots/segments, and manually adjust the prominence of each category in the photo library. Of these, the last feature should be the most useful and important. Increasing the prominence of a category will lead to sections whose contents fall into that category to be more likely to remain in the final, summarized movie. This is a much more intuitive parameter setting scheme compared to using parameters such as "skin percentage", "movement", or other various computer vision terminology as in existing systems, since the categories are those that had been defined by the user himself/herself.

5.3 Implementation

A prototype of TimeWarp has been implemented as an Objective-C Cocoa application for Mac OS X. Below, the basic functions of the implemented system are described, through a typical user scenario.

When the application is started, two windows, the media player window and the workspace window, appear on the screen (Figure 42). If the user's photo library had been already analyzed in the past, the system is ready to perform summarizations at this point. If not, the user must go through the photo library categorization process. This is done on a separate window, called the "palette". The palette is activated by choosing "Open Palette" from the main menu (Figure 43).

The user first prepares a subgroup of the library, by randomly drawing photos from his/her library onto the palette window. This is accomplished by clicking the "Draw" button on the palette window. The number of photos that are drawn with one click can be adjusted using the slider, placed to the right of the button. Photos can also be added by directly dragging image file icons onto the palette.

After the subgroup has been constructed, the user groups the photos in the subgroup by sliding them inside the palette window. When the grouping is finished,

Figure 42: Media player window (left), workspace windoe (right)

Figure 43: Palette window

the user can press the "Cluster" button to let the system automatically cluster nearby photos into categories. If the user is not satisfied with the formed categories, pressing the "Separate" button will separate them. Finally, after categories have been defined, selecting "Train Model with Palette" from the main menu will let the system train its internal model using the categories formed on the palette, and when this training is finished the system is ready to perform summarizations. Since photo library categorization does not have to be done every time the system is run, the palette is normally hidden and is activated only upon user request.

Now that the user's photo library has been analyzed and the system is ready to perform summarizations, the user can import the movie to be summarized by pressing the "Import" button on the media player window and selecting a file, or dragging a movie file icon onto the media player. The media player window has all the basic controls of a standard media player, including play/pause, rewind/fast

forward, volume adjustment, etc. Pressing the "Divide" button on the media player window initiates the video segmentation process, where the movie is divided into shots/segments, and an importance value is calculated for each shot (in a scale of A - F), and for each segment (in a scale of 100 - 0). Key frames of the segments are displayed on the workspace window in chronological order, and by default, the key frames are grouped by shots, and the height at which each key frame is displayed is adjusted according to the importance of the shot to which it belongs (Figure 44, top). This allows a quick and easy understanding of which shots are judged as important and which ones are not. By double clicking on the workspace window, the mode changes and importance values of individual segments are shown, using numbers and heights of key frames (Figure 44, bottom). For users who need more control, manually adding/removing shot boundaries, and adjusting importance values of shots/segments can be done on this window.

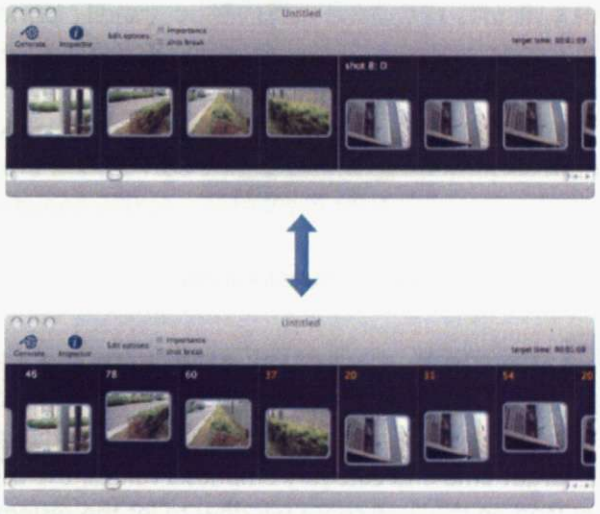

Figure 44: Importance values of key frames

The inspector panel (Figure 45) can be opened by pressing the "Inspector" button on the workspace window. The "target time", and emphasis values for categories, can be adjusted using this panel. The default target time is set to half of the original movie's length. Segments that will be cut off using the current

target time can be distinguished by looking at how their importance values are displayed; if the importance value is shown using orange numerals, the segment will be cut off and will not appear in the summarized video (Figure 44, bottom). As explained earlier, by default importance values for the segments are calculated automatically. However, users can affect this calculation of importance values by adjusting the emphasis values and deciding to emphasize or diminish segments with key frames belonging to certain categories.

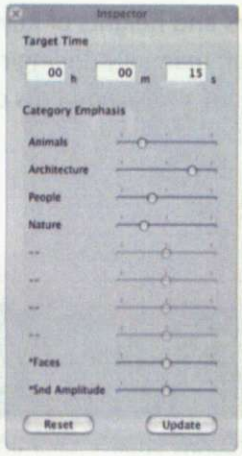

Figure 45: Inspector panel

After the input movie has been divided into shots/segments and the user has (if necessary) set the target time and category emphasis values, the user can press the "Generate" button on the workspace window to let the system generate a summarized movie. A new media player window pops up, resulting in a total of three windows (or four, if we include the palette window). The summarized movie can be exported into a QuickTime .mov file.

5.4 Evaluation

Two phases of evaluation tests were carried out, to assess the effectiveness of the prototype system. In the first phase the accuracy of the image classification algorithm was evaluated, and in the second phase, user studies were conducted by actually having users use the system to perform summarizations. (Note: the image classification algorithm used in this test was a preliminary version, which slightly differs from the current version described in 5.2.3)

5.4.1 Image classification performance

Two separate photo libraries were arranged for the test, one labeled as "simple", and the other as "normal". The "simple" library was a set of photos which had been deliberately chosen so as to be easy on the image classification algorithm, whose photos share characteristics such as simple compositions, sufficient lighting, clear contrast, etc. Figure 46 shows several examples of the photos in this library. The "normal" library, on the other hand, was actually a subset of my photo library. Being a typical personal photo library, the photos in the "normal" library tend to have complex compositions with many artifacts, do not necessarily show the subject clearly, and occasionally are blurred as results of inadequate lighting or camera shakes. Figure 47 shows several examples from this library. Each library contained 200 photographs in total.

Figure 46: Examples from "simple" library

Figure 47: Examples from "normal" library

Using the above two libraries, image classification was performed to key frames derived from three QVGA-size video clips, each lasting three minutes and shot in different conditions: clip1 was shot in Kashiwa campus of the University of Tokyo, a calm campus with large modern buildings, clip2 was shot around Kashiwa station, a somewhat urban area, and clip3 was shot in Daikanyama, a busy shopping district inside Tokyo.

The "simple" library was categorized into the following 6 categories.

Nature (shows plants, animals, etc.)
Graphic (contains graphic elements like signs, has many colors)
Architecture-Exterior (outdoor shots that focus on architecture)
Architecture-Interior (indoor shots)
Outdoor-Ground (generic outdoor shots aimed at the ground)
Outdoor-Sky (generic outdoor shots aimed at the sky)

The "normal" library was categorized into the following 8 categories. Due to the increased variety in the contents of the photos, the categories had to be defined in a more abstract manner in order to fit into the upper limit of 8 categories.

Nature (shows plants, animals, etc.)
Landscapes (shows dynamic landscapes such as rivers, valleys, oceans, etc.)
Friends (photos taken when partying, drinking, etc. with friends)
Exotic (photos showing foreign cultures; most of these were taken on vacations)
Colorful (photos containing many or bright colors)
Beautiful (photos perceived as beautiful to the user)
Outdoor (generic outdoor shots)
Indoor (generic indoor shots)

In either case, uncategorizable photos were deleted from the subgroup. However, this did not occur often since in each case there were categories to which "generic" shots could be assigned (for example Outdoor-Ground in the "simple" library, or Outdoor in the "normal" library).

The categorization of the library was done by the author throughout the test. The "correct" classification results for each clip were also determined subjectively by author, and the accuracy was calculated by comparing the system's classification results with the "correct" results.

Table 5 shows the results. The numbers under "subgroup" refer to the size of the subgroup of the library used in the photo library categorization process. Numbers under "clip1" through "clip3" refer to the percentage with which the system correctly classified the key frames (average value of three trials). Here, "correctly classify" means to correctly determine the category to which a key frame be-longs, or in cases where a key frame does not belong to any category, correctly determining it as "unclassifiable".

library	subgroup	clip1	clip2	clip3
simple	50	28.9	20.6	19.1
	100	31.0	23.4	18.2
	150	32.1	20.7	11.8
normal	50	28.7	25.5	11.8
	100	29.1	28.5	11.2
	150	36.3	25.2	11.2

Table 5: Image classification test results

It can be seen that in general, classification performance tends to improve with larger subgroup sizes. Also, clip1 has yielded the best results among the three video clips, while clip3 yielded the worst. Obviously the fact that clip3 had been shot in a busy shopping district is responsible for this; frames of video shot in busy, urban places tend to have more complex compositions with many objects, both of which make the situation more difficult for image classification algorithms. These findings are consistent with previous work on image classification.

The results for the "normal" library seem surprisingly good, especially considering that the "normal" library was grouped into a larger number of categories than the "simple" library. This is puzzling since in theory, the greater diversity of photos in the "normal" library should have negative impact on classification accuracy. Looking more closely into the results, for the "normal" library, the majority of the correctly classified key frames was found to belong to one of the "generic" categories (Outdoor and Indoor), and the accuracy for key frames belonging to non-generic categories were actually much lower. Since the generic categories were defined in effect as "none of the above", they tend to include photos with a wide range of visual features, and as a result many key frames without strong characteristic elements fit into these categories. However, this means that generic categories are not really of much help in determining the user's preferences, and so the apparent high accuracy for the "normal" library may actually be useless for TimeWarp.

These overly generic categories (Outdoor and Indoor in the "normal" library) can disrupt the overall summarization performance, since they tend to include many photos and be wrongly judged as showing contents which are "important" to the user, where in fact they tend to be relatively "unimportant". Therefore, these types of categories must be de-emphasized using the inspector panel, or prevented from being created in the first place. The design decision of limiting the number of definable categories could have been responsible for the forming of these overly generic categories, and if so, increasing or eliminating the limit altogether may be necessary.

Although presently impractical due to the problem of calculation time, it may be possible to further improve classification accuracy by using higher resolution video, or incorporating image segmentation techniques into the process.

Note that in this test the categorization was done by a single user (the author), and if different users conducted the same test, there would naturally be differences in the categorization results, which may affect the overall tendencies in the classification performance discussed above.

5.4.2 Summarization performance

9 students at the University of Tokyo were asked to each shoot a 10-minute, QVGA-size introduction video of their main campus (either Hongo or Kashiwa

campus). The students were then asked to summarize the recorded video using TimeWarp.

To assess how well the summarized video reflects each student's preferences, each student was asked to watch the video that they had shot and specify any number of "stress points", points in the video that the student thinks are important and want to have retained in the summarized video. Then, the students summarized their videos to one third of their original lengths using the system, after which the number of the "stress points" that had been correctly retained was counted and recorded. To make comparisons, summarized video clips that do not take individual preferences into account were also produced, by shortening each shot in the video to one third of its original length (this was done by simply cutting off 33% both from the beginning and the end of each shot). After the summarized clips had been generated, students were asked to watch the videos (they were not told which video was generated by TimeWarp) and to submit comments.

Through the results of the previous test, it has been found that the "normal" library, despite yielding seemingly high classification accuracy, may not produce sufficient classification performance for effective summarization using the current system. Therefore, although this may have deviated the test from actual use conditions, each student was asked to categorize the "simple" library (subgroup size: 100) and perform summarizations, instead of using his/her actual personal photo library. This causes a problem in that the number of photos in each category does not reflect the student's preferences, so the students were further asked to adjust the emphasis values for each category using the inspector panel, to account for their individual preferences towards the photos in each category. The average time students needed to complete the categorization was 10 minutes and 18 seconds.

Table 6 shows the results. It can be seen that video summarized by TimeWarp retained the "stress points" slightly (3.1%) better than the "no preferences" video, generated without consideration of user preferences. However, the increase is modest, and is not enough to claim statistical significance. This was echoed by students' comments where many students said they could not tell which video was the one generated by TimeWarp, and even in cases where they could they conceded that the differences were subtle. Part of the reasons for this may be the unexpectedly good results of the "no preferences" video. Since these clips were generated by simply taking the middle 33% of each shot, the fact that the percentage of retained "stress points" was much higher than 33% (which would likely have been achieved if random summarization was conducted, where

video segments are randomly cut off) suggests that users tend to capture "important" scenes in the middle rather than in the periphery of shots, which may be used to further improve the performance in future implementations of the system. Compared to 33%, TimeWarp's retrieval rate (47.7%) is much higher, which proves its effectiveness over random summarization.

user	"stress points"	retained "stress points"		
		our system	no preferences	random
1	12	5	2	-
2	9	7	3	-
3	18	7	5	-
4	6	5	4	-
5	10	5	6	-
6	25	11	12	-
7	14	6	8	-
8	24	12	13	-
9	12	4	5	-
total	130	62 (47.7%)	58 (44.6%)	33.3%

Table 6: Summarization test results

Further analysis of the "stress points" specified by the students revealed that at many of those points, the subject was something that did not belong to any of the defined categories. For example, many of the "stress points" showed people but since the "simple" library contained only a few photos of people, the system was unable to understand the significance of these points, and they could not be retained. Other subjects which appeared in students' "stress points" but were rare or nonexistent in the "simple" library include small birds, cars, food, etc. Initially the small diversity in the "simple" library was thought to be a positive attribute since it leads to higher classification accuracy, but in reality it had a major draw-back of being limited in the ability to cover users' diverse interests.

Further evaluations, using users' actual personal photo libraries which should have more diverse photos, are planned for the future.

Since the test was not a precise reflection of real-life use conditions due to the fact that actual personal photo libraries were not used, it is still unknown if using personal photo libraries are really effective for video summarization. However, it did become known that user-specified preferences on photo libraries could have a positive impact on video summarization, which backs up the basic assumption that there exists a link between users' preferences on still photos and video.

5.5 Summary

This chapter described TimeWarp, a video summarization system that uses personal photo libraries to account for individual user preferences. Results of the evaluation tests show the potential of the system to serve as a powerful automatic/semi-automatic video summarization solution. Future work includes further evaluation of the effectiveness of the system under more diverse conditions, using various photo libraries and video clips. Also, there are plans to incorporate several proven conventional video summarization techniques, such as speech recognition and face detection, to further improve the effectiveness of the system.

6 Design Process Examination

This chapter follows the design processes of CityVoyager and TimeWarp, illustrating how the conceptual framework discussed in Chapter 2 can assist the various decision makings involved in the development of Life Adaptive applications.

6.1 CityVoyager

Suppose that we decide to build an application that can "offer effective shop recommendations to users in the city". This type of application development, which starts by defining the objective of the application, is known as problem-driven design. An alternative approach where development starts with the fundamental technology is called technology-driven design. Brief descriptions of these two approaches are given below.

Problem-driven design

In problem-driven design, development begins by first drawing up the final objective of the application, such as "solve problem ○○" or "offer service △△", with all subsequent decision makings concentrating on how to achieve that objective. Technical details are usually left undetermined in the beginning, and sorted out later as the development proceeds. This approach is similar to those commonly taken by business consultants and product designers.

Technology-driven design

In technology-driven design, development takes the form of seeking useful applications for a particular technology. This type of approach is common in basic research, where technical innovations are usually given priority over developments of practical applications.

Note that the above two approaches are in no way meant to be the only viable paths of application development; they are simply given as examples of the more widely-adopted variations. The discussions hereafter in this section will follow the route of problem-driven design. The process will be drastically different, if technology-driven design had been chosen.

6.1.1 Decision of adopting Life Adaptive Computing

At this stage only the objective of the application is given, and none of the details are in place, including whether or not to develop CityVoyager as a Life Adaptive application. The list of "targets of adaptation" included in the framework should be beneficial when considering the use of Life Adaptive Computing. If an application involves the "adaptation" to any of the listed "targets", then perhaps Life Adaptive Computing may be suitable for its design.

In the case of CityVoyager, it is clear from its objective that adaptation to individual "user preferences" will be necessary, and hence the adoption of Life Adaptive Computing can be naturally determined.

6.1.2 Initial design process

Now the decision has been made to develop CityVoyager as a Life Adaptive application. As described in the framework, Life Adaptive applications offer their "adaptive" services through a procedure consisting of three "phases": Activity Observation, User Profile Update, and Service Generation. Technical details must be filled in for each of these phases.

There is no particular order in which the specifics of these phases should be determined. Since the phases are closely interrelated, it would be difficult to design the internals of each phase in complete isolation, and in most cases design of these phases should end up taking a more or less simultaneous manner.

Activity Observation & User Profile Update

The central tasks in designing the Activity Observation phase are choosing the appropriate activity from which CityVoyager can perform the necessary estimation of "user preferences", and also determining how it can be observed. The list of activities provided in the framework can be referred here for a realistic range of choices.

In the case of CityVoyager, due to its comparability to online recommendation systems, it is fairly easy to come up with the idea of observing "the user's visits to shops" (which falls into "location change" in the list of activities). However, devising a practical method to obtain this information is more difficult since simply using GPS is insufficient due to its large errors, and alternative location detection

techniques with better accuracy are not yet feasible with current infrastructure.

CityVoyager's solution to this problem has been devised via a two-fold process. First, the format of the user profile is determined as "a list of shops that the user has frequented in the past". Next, a hypothesis is set forth that by using pattern recognition techniques, this user profile can be obtained even from error-prone location data. Devising clever technical solutions like this is one place where the conceptual framework is unable to present any useful clues, and success fully relies on the developer's expertise.

The user profile takes man-size granularity here. In most cases, considering the use of other granularities should only be required if necessities arise afterwards.

Service Generation

The design of Service Generation techniques is straightforward, since CityVoyager shares its objective of offering recommendations with conventional recommendation systems, and the use of similar techniques can be considered. In general cases, the internals of Service Generation is application-dependent and will not be as easy.

The last step of initial design is to add various refinements to the system, such as the use of real-time context, to improve the quality of user experience. Considering the fact that CityVoyager is used in the city, incorporating information about the current user location would be a clear improvement.

This concludes the description of CityVoyager's initial design process. Of course the completion of the initial design by no way means the development itself is finished, and this process must be followed by numerous cycles of prototype creation and evaluation. Furthermore, the technical problem-solving involved in the process is something which cannot be covered by the framework. However, it should be clear that using the conceptual framework gives a level of coherency to the design process, and also assists the developer in several crucial design decisions by presenting the possible range of choices.

6.2 TimeWarp

As with the case of CityVoyager, the following descriptions regarding TimeWarp will follow problem-driven design, with the objective of the application as "to offer accurate automatic video summarization".

6.2.1 Decision of adopting Life Adaptive Computing

The objective of TimeWarp, set as above, is still a little too abstract to determine whether it should be developed as a Life Adaptive application. We first attempt to rewrite the objective in a more detailed manner, by asking the question: What makes an accurate automatic video summarization system?

Surveys of existing automatic video summarization techniques may lead to one observation, that conventional systems' indifference to user preferences are what had led to their low accuracy. Following this observation the objective of TimeWarp could be clarified as "to offer automatic video summarization tailored to each user's preferences", which makes it eligible to be developed as a Life Adaptive application.

6.2.2 Initial design process

Activity Observation

Again, the list of activities in the framework can be referred in deciding what activity should be observed in TimeWarp. Here, we focus on indirect observation using the user's personal photo library, interpreting digital photos as "artifacts" left behind as results of a person's past activities (shooting photos, "creation" in the list of activities).

Note that this decision of using photo libraries as cues into the user's preferences on video summarization exploits a relatively vague connection between the observed activity and the "target of adaptation", compared to the rather straightforward case of CityVoyager. Being able to design applications based on such vague connections make up one of the advantages of Life Adaptive Computing, but revealing such connections requires extremely keen insights on part of the developer. As discussed in Chapter 2, Topic 2-2, being acute observers of everyday lives may help the developer in coming up with such insights.

User Profile & Service Generation

The user's preferences must be extracted from the photo libraries into some concrete form, which will constitute the user profile in TimeWarp. For this task, it may be necessary to look ahead and have a general idea about what technique can be used in Service Generation. Once the decision has been made to use image classification techniques for Service Generation, the technical requirements of the process will help build the specifications of the profile.

Designing the specific techniques for Service Generation, in this case involving the use of image classification, is another place where the developer's technical expertise must be called to use. However, the two filtering algorithms described in Chapter 2, Topic 3-1 may alleviate the difficulty to some extent. TimeWarp's Service Generation procedure can actually be derived as a (rather extensive) modification of the content-based filtering technique.

As the final stage of initial design, we consider adding refinements to the overall process. TimeWarp implements several manual parameter setting schemes, which enable users to make precise adjustments.

We have seen that using the conceptual framework can assist the development of Life Adaptive applications in various ways. However, resolutions of some technical details are still left out to rely almost completely on the abilities of the developer, such as devising the specific method of Service Generation. Since developments of Life Adaptive applications can demand considerably high levels of technical expertise, it is not unthinkable for a valid initial idea being out of reach due to the ineptitude of the developer.

Also note that the designs of CityVoyager and TimeWarp described here are not the only way of achieving their respective initial objectives but are simply possible examples among many alternatives, such as focusing on credit card usage instead of location changes in the case of CityVoyager. Design, by nature, cannot be completely rational and some level of arbitrariness will always remain.

7 Concept Sketches

The two case studies in this thesis have focused on introducing applications that can realistically be developed with the current/near-future state of technology. This makes the applications somewhat technically conservative, and more bold examples may be needed to fully map out the future possibilities of Life Adaptive Computing. Below, we give several ideas for Life Adaptive applications, which may be realized with future technical advances.

Overlaid SNS

Overlaid SNS is a new type of Social Networking Service (SNS), an alternative to conventional services which have the tendencies to lure users increasingly deeper into the virtual world. The system is a real-world oriented SNS where physical activities gathered through a mobile device act as catalysts for online community generation. The name "Overlaid SNS" refers to the fact that whereas in conventional services the virtual (online) world exists more or less separated from the real world, in this system the virtual world is literally "overlaid" onto the real world (Figure 47).

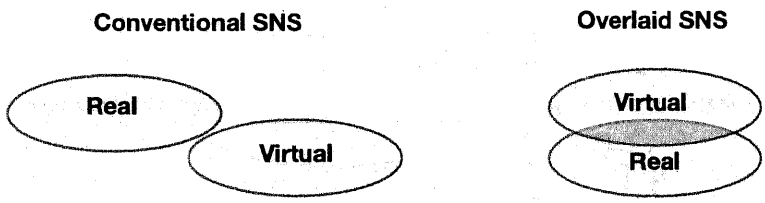

Figure 47: Overlaid SNS

Conversation support system

Coming up with good conversation topics is not always easy, especially when talking with people not yet well acquainted. The Conversation Support System (Figure 48) can help in these situations by automatically generating conversation topics from past user activities (locations, purchases, etc).

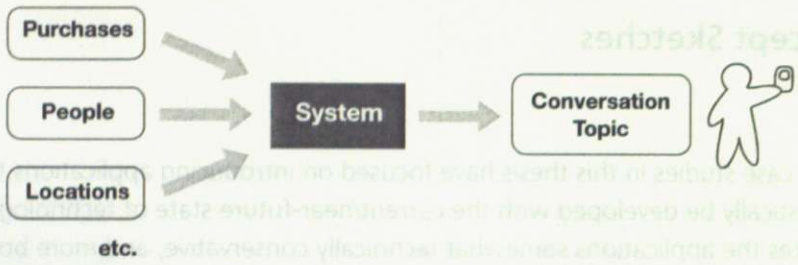

Figure 48: Conversation support system

Life adaptive storyteller

The Life Adaptive Storyteller (Figure 49) weaves and tells stories on the user's mobile device, using both text and graphics. The user's real-world activities dynamically influence the story line. Sometimes, the user may be required to perform particular activities to advance the plot.

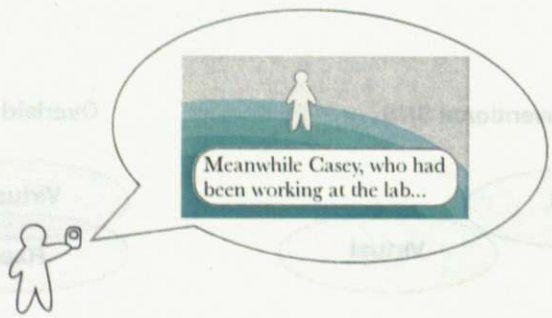

Figure 49: Life Adaptive storyteller

Personal color ID

The Personal Color ID system generates personalized color palettes by tracking the user's fashion. A color palette is generated by analyzing user images, recorded daily with a camera-embedded mirror (Figure 50). The palette can act as a personal ID, by being translated onto various everyday objects.

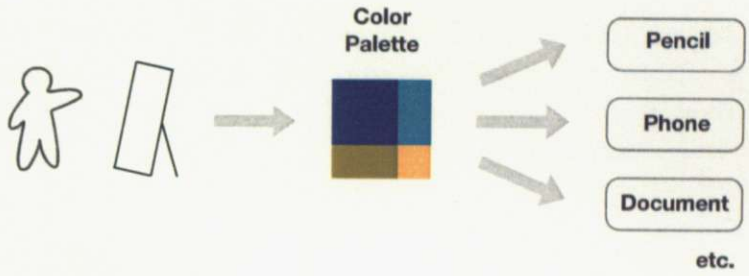

Figure 50: Personal color ID

The Life Adaptive applications described in this dissertation, including the case studies and the above concept sketches, are mostly software-centered. However, since control of hardware through software technology is becoming increasingly easier and more sophisticated lately, it may not be long before we will see applications that can literally change its physical form to adapt to each user.

8 Conclusion

In this dissertation I have proposed and described Life Adaptive Computing, a novel HCI concept which allows computer systems to dynamically adapt themselves to ideally suit each user. As demonstrated by the case studies, using this concept can lead to a new class of practical applications, which offer services unattainable within the scope of conventional concepts.

Life Adaptive applications are not limited to typical software applications; with recent technical advances, Life Adaptive applications which "adapt" by literally changing their physical forms are expected to become reality in the near future. The conceptual framework discussed in this thesis should assist designers/engineers in developing such innovative Life Adaptive applications, which would in turn contribute to further sophistication of the conceptual framework by giving valuable feedback.

The dissertation has mostly focused on the advantages, the positive aspects of Life Adaptive Computing. But depending on its use this concept can also pose some serious threats, most significantly invasion of privacy. Since Life Adaptive applications "adapt" through close observations of the user's daily life, privacy risks are an inevitable part of every Life Adaptive application.

To alleviate this risk, the case studies presented in this thesis have been designed based on the principle of "least activity observation" - to only monitor activities that are integral and indispensable to the services offered by the application. In the case of CityVoyager observation is limited to the user's location data, and in TimeWarp the only personal information exploited is photo libraries. The point is to make clear what activities will be utilized by each Life Adaptive application, so that users can make educated choices when using an application; they can consider whether the service offered by the application is worthy of the invasion of privacy it requires. If the service is attractive enough users may decide that the sacrifice is justified, just like many people willingly relinquish some privacy when using credit cards.

Note that this approach is drastically different from the ones commonly used in experiments on Ubiquitous Homes, or Context-Aware Environments. In those experiments, attempts are made to monitor every type of activity that can be ob-

served, and construct a central database of personal activities that can be used for a variety of applications. In such cases, there is great risk that users will not be able to make conscious choices as when following the "least activity observation" principle, since the connection between the service and the personal data used for it becomes unclear. Therefore, although the notion of creating a central database of activities may be tempting for developers, it could turn out unacceptable for widespread use.

Future developments of Life Adaptive applications are expected to trigger rigorous public debate about its aforementioned risks, and someday a general consensus may be reached about the acceptable level of privacy invasion.

References

[1] Schilit B., Adams N., Want, R. Context-Aware Computing Applications. In Proc. of WMCSA'94, pp.89-101.

[2] Dey, A.K., Salber, D., Abowd, G.D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. HCI Journal, vol.16, pp. 97-166. 2001.

[3] Riecken, D. Personalized Views of Personalization. Communications of the ACM, vol. 43 no.8, pp.26-28. 2000.

[4] Adomavicius, G., Tuzhilin, A. Personalization Technologies: A Process-Oriented Perspective. Communications of the ACM, vol.48 no.10, pp.83-90. 2005.

[5] Fukasawa, N. Naoto Fukasawa. Phaidon Press, 2007.

[6] Goldberg, L.R. The Structure of Phenotypic Personality Traits. American Psychologist, vol.48, pp.26-34. 1993.

[7] Woodham, J.M. Twentieth-Century Design. Oxford University Press, 1997.

[8] Le Corbusier, Yoshizaka, T. Modulor 1. Kajima Publishing, 1976.

[9] Beyer, H., Holtzblatt, K. Contextual Design. Morgan Kaufmann Publishers, 1998.

[10] Chen, L., Sycara K. WebMate: Personal Agent for Browsing and Searching. In Proc. of AGENTS'98, pp.132-139.

[11] Lang, K. NewsWeeder: Learning to Filter Netnews: In Proc. of ICML'95, pp.331-339.

[12] Goldberg, D., Nichols, D., Oki, B., Terry, D. Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, vol.35 no.12, pp.61-70. 1992.

[13] Shardanand, U., Maes, P. Social Information Filtering: Algorithms for Automating "word of mouth". In Proc. of CHI'95, pp.210-217. 1995.

[14] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M. Cyberguide: A Mobile Context-Aware Tour Guide. ACM Wireless Networks, vol.3, pp. 421-433. 1997.

[15] Want, R., Hopper, A., Falcao, V., Gibbons, J. The Active Badge Location System. ACM Trans. on Information Systems, vol.10 no.1, pp.91-102, 1992.

[16] Want ,R., Schilit, B., Adams, A., Gold, R., Petersen, K., Goldberg, D., Ellis, J., Weiser, M. The ParcTab Ubiquitous Computing Experiment. Technical Report CSL-95-1, Xerox PARC.

[17] Bahl, P., Padmanabhan, V.N. RADAR: An In-Building RF Based User Location and Tracking System. In Proc. of INFOCOM 2000, pp.775-784.

[18] Ward, A., Jones, A., Hopper, A. A New Location Technique for the Active Office. IEEE Personal Communications vol.4 no.5, pp.42-47. 1997.

[19] Cheverst, K., Davies, N., Mitchell, K., Friday, A. Experiences of Developing and Deploying a Context-Aware Tourist Guide: the GUIDE Project. In Proc. of MOBICOM 2000, pp.20-31.

[20] LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., and Schilit, B. Place Lab: Device Positioning Using Radio Beacons in the Wild. In Proc. of PERVASIVE 2005, pp.116-133.

[21] Marmasse, N., Schmandt, C. Location-aware Information Delivery with Commotion. In Proc. of HUC 2000, pp.157-171.

[22] Ashbrook, D., Starner, T. Learning Signifcant Locations and Predicting User Movement with GPS. In Proc. of ISWC'02, pp.101-108.

[23] Liao, L., Fox, D., Kautz, H. Learning and Inferring Transportation Routines. In Proc. of AAAI 2004, pp.348-353, 2004.

[24] Asthana, A., Cravatts, M., Krzyzanouwski, P. An Indoor Wireless System for Personalized Shopping Assistance. In Proc. of WMCSA'94, pp 69-74.

[25] Mynatt, E.D., Back, M., Want, R., Baer, M., Ellis, J.B. Designing Audio Aura. In Proc. of CHI'98, pp.566-573.

[26] Wisneski, C., Ishii, H., Dahley, A., Gorbet, M., Brave, S., Ullmer, B., Yarin, P. Ambient Displays: Turning Architectural Space into an Interface between People and Digital Information. In Proc. of CoBuild'98, pp.22-32.

[27] Van Erp, J.B.F., Van Veen, H.A.H.C., Jansen, C., Dobbins, T. Waypoint Navigation with a Vibrotactile Waist Belt. ACM Trans. on Applied Perceptions, pp.106-117. 2005.

[28] Froehlich, J., Chen, M.Y., Smith, I.E., Potter, F. Voting With Your Feet: An Investigative Study of the Relationship Between Place Visit Behavior and Preference. In Proc. of UbiComp 2006, pp.333-350.

[29] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-based Collaborative Filtering Recommendation Algorithms. In Proc. of WWW10, pp.285-295. 2001.

[30] Ju, S. X., Black, M. J., Minneman, S., Kimber, D. Summarization of Video-taped Presentations: Automatic Analysis of Motion and Gesture. IEEE Trans. on Circuits and Systems for Video Technology. 1998.

[31] Yow, D., Yeo, B. L., Yeung, M., Liu, G. Analysis and Presentation of Soccer Highlights from Digital Video. In Proc. of Second Asian Conference on Computer Vision. 1995.

[32] Smith, M. A., Kanade, T. Video Skimming and Characterization through the Combination of Image and Language Understanding Techniques. CMU Technology Report. 1997.

[33] Ma, Y., Lu, L., Zhang, H., Li, M. A User Attention Model for Video Summarization. In Proc. of Multimedia 2002.

[34] Mei, T. Hua, X. Zhou, H. Tracking Users' Capture Intention: A Novel Complementary View for Home Video Content Analysis. In Proc. of Multimedia 2005.

[35] Parshin, V., Chen, L. Video Summarization based on User-Defined Constraints and Preferences. In Proc. of RIAO 2004.

[36] Mori, Y., Takahashi, H., Oka, R. Image-to-Word Transformation based on Dividing and Vector Quantizing Images with Qords. In Proc. of MISRM 1999.

[37] Carson, C., Belongie, S., Greenspan, H. Malik, J. Blobworld: Image segmentation using Expectation-Maximization and its application to image querying. IEEE Trans. on Pattern Analysis and Machine Intelligence. 1999.

[38] Duygulu, P., Barnard, K., Freitas, J. F. G., Forsyth, D. A. Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary. In Proc. of European Conference on Computer Vision. 2002.

[39] Li, J., Wang, J. Z. Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach. IEEE Trans. on Pattern Analysis and Machine Intelligence. 2003.

[40] Jeon, J., Lavrenko, V., Manmatha, R. Automatic Image Annotation and Retrieval using Cross-Media Relevance Models. In Proc. of SIGIR 2003.

[41] Pass, G., Zabih, R. Comparing Images Using Joint Histograms. Journal of Multimedia Systems. 1998.

List of Publications (selected)

Takeuchi, Y., Sugimoto, M. A User-Adaptive City Guide System with an Unobtrusive Navigation Interface. Personal and Ubiquitous Computing, Springer London, DOI 10.1007/s00779-007-0192-x. 2007.

Takeuchi, Y., Sugimoto, M. A User-Adaptive City Guide System based on Location Data History. IEICE Transactions on Information and Systems, vol.J90-D, no.11, pp.2981-2988. 2007.

Takeuchi, Y., Sugimoto, M. User-Adaptive Home Video Summarization using Personal Photo Libraries. In Proc. of CIVR 2007, pp.472-479.

Takeuchi, Y., Sugimoto, M. Video Summarization using Personal Photo Libraries. In Proc. of MIR 2006, pp.213-222.

Takeuchi, Y., Sugimoto, M. An Intelligent City Guide with a "Metal Detector" Interface. In Adjunct Proc. of UIST 2006, pp.97-98.

Takeuchi, Y., Sugimoto, M. CityVoyager: An Outdoor Recommendation System based on User Location History. In Proc. of UIC 2006, pp.625-636.

Takeuchi, Y., Sugimoto, M. An Outdoor Recommendation System based on Uesr Location History. In Proc. of UbiPCMM 2005, pp.91-100.