Realtime Personal Positioning System for Wearable Computers

Hisashi Aoki[◊], Bernt Schiele and Alex Pentland
MIT Media Laboratory
{aoki,bernt,sandy}@media.mit.edu

Abstract

Context awareness is an important functionality for wearable computers. In particular, the computer should know where the person is in the environment. This paper proposes an image sequence matching technique for the recognition of locations and previously visited places. As in single word recognition in speech recognition, a dynamic programming algorithm is proposed for the calculation of the similarity of different locations. The system runs on a stand alone wearable computer such as a Libretto PC. Using a training sequence a dictionary of locations is created automatically. These locations are then be recognized by the system in realtime using a hat-mounted camera.

1. Introduction

Obtaining user location is one of the important functions for wearable computers in two applications. One is automatic self-summary, and the other is context-aware user interface. In self-summary, the user is wearing a small camera and a small computer, capturing and recording every event of his/her daily life. The computer might be able to summarize the recorded data and eventually producing a "diary" [1,2]. For example, "At 8:15, left home. At 8:17, happened to meet Bob on the street. At 8:44, arrived at the office…" The computer might be also able to navigate the way to an unknown place. Furthermore, important reminders might be shown on one's worn display when the user is in a particular location, such as supermarket [3,4,5,6].

As for context-awareness, Thad Starner et al. discussed the following example [7]:

> *"For example, if the user is in his supervisor's office, he is probably in an important meeting and does not want to be interrupted for phone calls or e-mail except for emergencies."*

To realize these functions, a computer needs to know user's locations. Outdoors, a Global Positioning System (GPS) may track movements. Indoors, active tags may take this role [8,9,10], however, sticking transmitters on all the doors is obviously a difficult task, and GPS is not available inside buildings.

The other approach to recognize location is to use a camera. If barcodes are stuck on every door, the computer may find the place easily by reading the code [10]. Even if there is no barcode, object recognition or text recognition may help the computer detect the place [11]. However, many of these computer vision approaches analyze still pictures. So accurate segmentation of object or character is needed. Starner et al. showed a place detection method by introducing an HMM of transition of rooms [7]. Clarkson et al. showed a method to segment and label audio-video sequences from a worn camera and microphone [12].

In this paper, we propose a realtime personal positioning system that only uses a small wearable camera as shown in Figure 1, and a standalone PC.

In previous work [13], we demonstrated that a dynamic programming algorithm [14], can be used to recognize not only the user's location but also the approaching trajectory. In [13], we tested the accuracy of the system with about 100-second video sequences that were manually chosen and segmented. The images were handled off line on an SGI workstation.

In this paper, practical changes have been done added to the previous work to run on wearable computers. Firstly, the proposed system runs at about 7 frames per second, on a standalone PC such as a Toshiba Libretto without wireless communication to remote database. This overcomes delays caused by database reference over limited bandwidth. Secondly, no segmentation of the video sequences is needed. Thirdly, we introduce a method to choose suitable trajectories for the "location dictionary" automatically. And at the recognition stage,

[◊] Now at Toshiba Corporation R&D Center, Japan the user doesn't need to tell the system when to look up on the dictionary. Lastly, larger sets of training and test data are used for the evaluation. The dictionary is extracted from a real video sequence of 16 minutes and 30 seconds, and testing is done with independent video sequences of a random walk of 7 minutes and 30 seconds. We introduce and use accuracy scales similar to the ones used by [7].

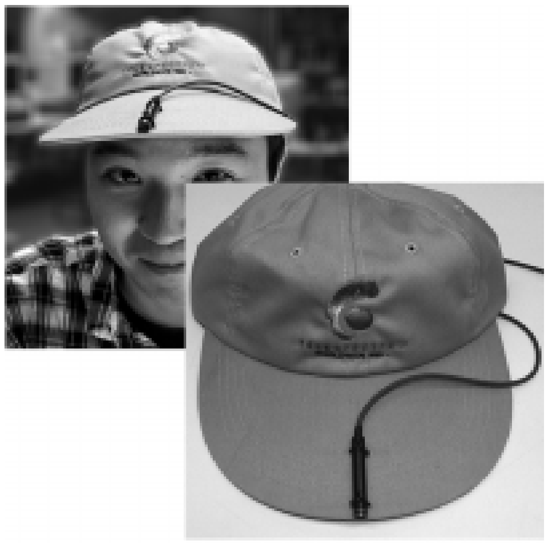

Figure 1 A small wearable camera.

In the following section, an overview of the system is given. In Section 3, the histogram calculation of each frame and the similarity measurement of video segments are explained. Then in Section 4, the method to automatically select a location for the location dictionary is discussed. The experimental results are shown in Section 5.

2. System Overview

The system consists of a notebook PC, PCMCIA video capture card and wearable camera (see Figure 1 for the wearable camera). It completely runs in realtime on a standalone PC and does not need access to a remote server. User operates the system in two phases, which are training phase and recognition phase. Between these two phases, the location dictionary is calculated offline. At training phase, while the user is walking about in a building, a chromatic histogram is calculated and recorded for every frame (Figure 2). After recording, the program looks for suitable segments for the trajectory dictionary in the histogram sequence. From the top of list, trajectory segments are stored in the dictionary. This process is discussed in Section 4.

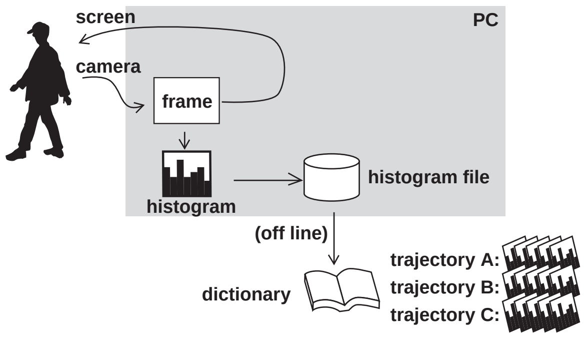

Figure 2 Data flow at recording mode.

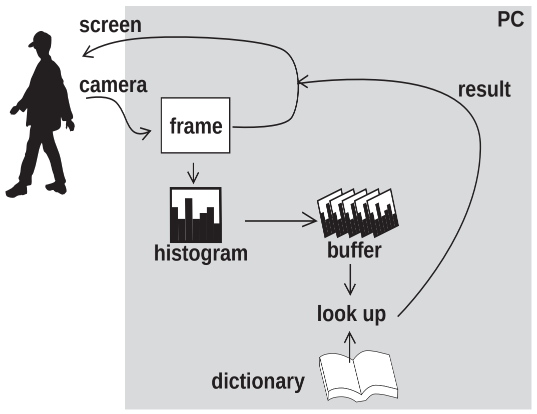

Figure 3 Data flow at analysis mode.

Subsequently, the user walks about in the recognition phase. Again the chromatic histogram is calculated and matched to the trajectories in the dictionary. When a trajectory is similar enough to the recent frames, the system tells the user that he/she has walked through there before, and shows him/her the image of the detected location (Figure 3).

We use an ordinary notebook PC that has an Intel Pentium 166 MHz processor and supports MMX instructions. Training and test video sequences are recorded by a camcorder to allow repetitious analysis and evaluation. However, the system layout is compatible and substitutable with a small computer such as Toshiba Libretto and a camera shown in Figure 1. Libretto 100CT has the same processor speed as the notebook PC we used.

The program is written in C++ and partially in assembly language to provide faster access to MMX instructions. Since MMX supports 64-bit calculation at a time, histogram and some other calculations can be done as fast as 10 Hz.

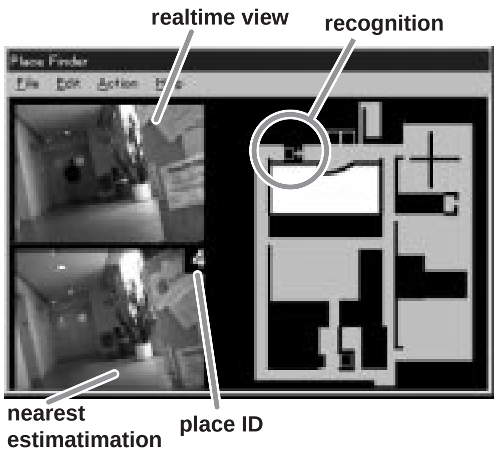

Figure 4 Screen shot of the proposed personal positioning system.

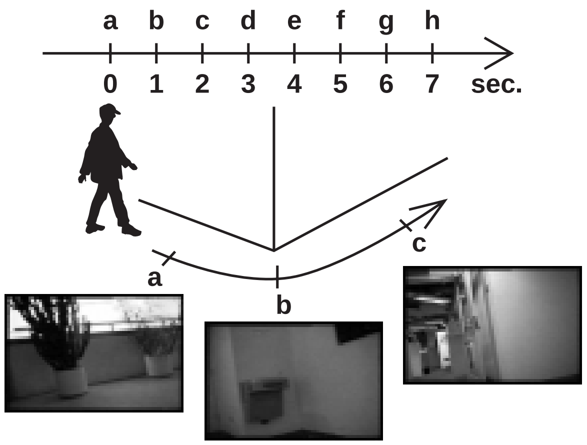

Figure 5 Sparse sampling causes bad performances.

3. Histogram Similarity

In order to implement a vision system onto a wearable computer, we have to keep in mind the following:

- The system should be robust against brightness and noise.
- Calculation for each frame should be fast enough in order to enable high rate of sampling.
- Representation of each frame should be as compact as possible.

The frame representation should not be affected by brightness since lighting can be different from time to time, day or night, even indoors. The second point concerns speed. If the calculation is so heavy that the system can handle only one frame in two seconds, the image sequences can be very different even if the user is passing through the same path at training phase and at test phase. For example in Figure 5, dictionary trajectory for location A is made out of frame (a), (c), (e), (g) and test sequence is sampled as (b), (d), (f), (h). If the sampling frequency is high enough, (a) and (b) are expected to be similar and the system may succeed in matching the two sequences. But if the gap between (a) and (c) is two seconds, (a) and (b) can be quite different which will cause the system to fail.

Frame features are stored in a location dictionary. This means, due to limited space of storage, the smaller the size of feature is, the more locations can be stored. Also, the larger the size of the feature, the slower the performance of the system will be.

3.1 Hue histogram

We calculate hue histograms as frame features. Pixel values are represented by hue (*H*), saturation (*S*), and brightness (*B*). HSB, YUV and RGB units are converted to each other. We linearly translated RGB unit to YUV, and then calculated H value by taking the arctangent of (*U*,*V*). The hue (*H*) histogram is used as a frame feature. We calculate 36-bin histogram, resulting in a 36-dimensional feature vector **f** for each frame.

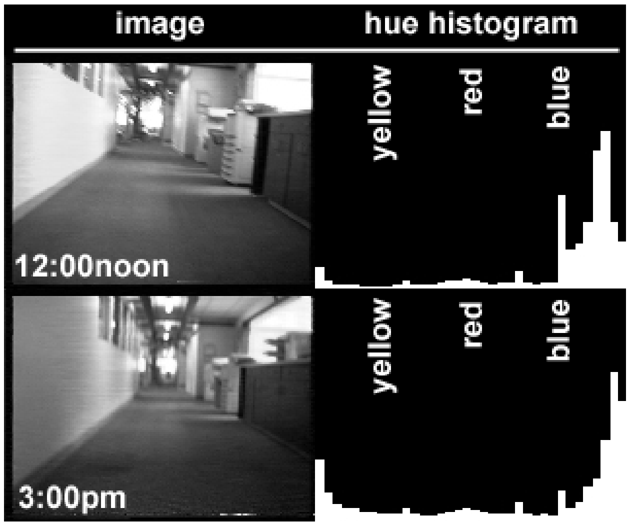

Figure 6 Chromatic (hue) histogram

Figure 6 shows the example of chromatic histogram. As seen in Figure 6, chromatic histogram is robust against angle of camera and time of day [15].

A sequence of **f** represents sequentially incoming video frames. We denote a sequence of **f** as **F**:

$$\mathbf{F}_{se} = \left(\mathbf{f}_s, \mathbf{f}_{s+1}, \ldots, \mathbf{f}_e\right)$$

where $\mathbf{f}_s$ is the chromatic histogram of the s-th frame and $\mathbf{f}_e$ of the e-th frame.

In the system, a histogram element is represented in a byte (8-bit). Therefore, **f** occupies 36 bytes in the memory. The system uses a table of tangent value for 5, 15, 25, 35 degrees to calculate to which histogram bin a pixel belongs. This part of the system is implemented in MMX without using floating point calculation. This calculation runs at about 10Hz on 166MHz PC.

3.2 Video segment similarity

Once $\mathbf{F}_{se}$ is calculated, similarity between two segments is calculated in the following way. The distance matrix between segment *i* and segment *j* is introduced as:

$$\mathbf{D}_{ij} = \begin{pmatrix} d\left(\mathbf{f}_{s_i}, \mathbf{f}_{s_j}\right) & d\left(\mathbf{f}_{s_i+1}, \mathbf{f}_{s_j}\right) & \cdots & d\left(\mathbf{f}_{e_i}, \mathbf{f}_{s_j}\right) \\ d\left(\mathbf{f}_{s_i}, \mathbf{f}_{s_j+1}\right) & \ddots & & \vdots \\ \vdots & & & \vdots \\ d\left(\mathbf{f}_{s_i}, \mathbf{f}_{e_j}\right) & \cdots & \cdots & d\left(\mathbf{f}_{e_i}, \mathbf{f}_{e_j}\right) \end{pmatrix}$$

where

$$d\left(\mathbf{f}_k, \mathbf{f}_l\right) = \left|\mathbf{f}_k - \mathbf{f}_l\right|^2$$

If segment *i* and *j* are the same, diagonal elements in $\mathbf{D}_{ij}$ are 0.

If the user passed through the same location at the same speed in segments *i* and *j*, the diagonal elements in $\mathbf{D}_{ij}$ are expected to be small.

If the user passed through the same location in segments *i* and *j*, but the user was slower in *j* than in *i*, $\mathbf{D}_{ij}$ would have the form of Figure 7.

Therefore, the similarity between segments *i* and *j* should be calculated along corresponding elements, which are not necessarily on diagonals. We search a path from the (1,1) element to the $(e_i\text{-}s_i, e_j\text{-}s_j)$ element of D_{ij} under the constraints:

- proceed from (m,n) only to $(m+1,n)$ $(m,n+1)$, or $(m+1,n+1)$.
- along the chosen path, the sum of the elements is minimum.

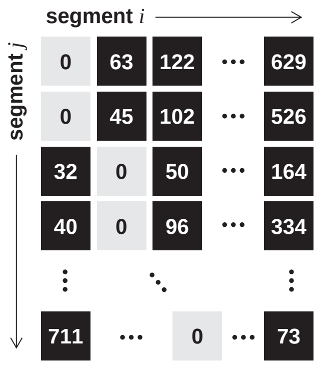

Figure 7 Difference in speed.

The path chosen in the above way is called minimal path, and the sum of elements along the minimal path is denoted as $\mathbf{T}_{ij}$. For example, if $\mathbf{D}_{ij}$ is;

$$\mathbf{D}_{ij} = \begin{pmatrix} 0 & 3 & 2 & 9 \\ 3 & 4 & 5 & 2 \\ 7 & 9 & 2 & 0 \end{pmatrix}$$

Here, the minimal path is (1,1)-(2,2)-(3,3)-(3,4) and $\mathbf{T}_{ij}$ is 0+4+2+0 = 6. (2,2) is chosen even if it is greater than (1,2) or (2,1), since along paths via (1,2) or (2,1), the sums of elements are greater than via (2,2). Dynamic programming frequently used in speech recognition can be used to find the minimal path [16]. Dynamic programming finds the correspondence of frames in segments even if speed is different or not stable.

The proposed system uses 40 frames for each trajectory or video segment. When the location dictionary has 16 trajectories, the dictionary actually holds 40x16 = 640 histograms. When a sequence of histograms for a location *P* is denoted as $\mathbf{F}_{dict(P)}$, and a sequence of histograms for recent frames at time *t* is denoted as $\mathbf{F}_{test(t)}$, we note the following incremental comptation of **D**:

$$\left(\mathbf{D}_{dict(P)test(t)}\right)_{kl} = \left(\mathbf{D}_{dict(P)test(t-1)}\right)_{k,l+1} \qquad \left(l < e_{test}\right)$$

Therefore, only $\left(\mathbf{D}_{dict(P)test(t)}\right)_{k,e(test)}$ is calculated whenever the system receives a new frame. The cost per frame is one histogram calculation, 40x16 = 640 similarity $\left(\mathbf{D}_{dict(1..16)test(t)}\right)_{k,40}$ calculation, and 16 searches for minimal paths.

4. Trajectory Dictionary

Not all locations are suitable for the trajectory dictionary. For example, when a video segment is recorded while the user walks 20 feet in a monotonous hallway, and another segment is recorded of another 20 feet of the same hallway, it is impossible to differentiate the two sequences. Only discriminative segments are suitable to be chosen for the trajectory dictionary. Therefore, a measurement of distinctiveness will be introduced in the following.

We calculate $\mathbf{T}_{ij}$ for all the combinations of segments in the test data with segments partially overlap to each other. Distinctiveness measurement (M) for segment i is introduced as follows:

$$M_i = \sum_{all\_j} \mathbf{T}_{ij} \Big/ (\text{number\_of\_segments})$$

In the training data, the same location can be visited more than once. Ideally, segments from the same location should be eliminated from the calculation of M_i. However, the number of segments to be eliminated is relatively small with respect to all the segments. And it is difficult to eliminate them without having a priori association of location and training video sequence.

After M_i is calculated, the system shows the distinctive locations by using the order of M. This list can contain segments too close to each other. For example, if the segment from frame 301 to 340 is on the top of list, segment from 302 to 341 may be listed on the second or third. Therefore by eliminating overlapping segments from the list, the system creates a trajectory dictionary of the most distinctive locations in the environment.

5. Experiments

A training video sequence and a test sequence are taken in the MIT Media Lab Building. The training sequence is 16 minutes and 30 seconds in duration, and the test sequence has 7 minutes and 30 seconds. Both sequences are taken by walking mainly around the third floor. However, in both cases, the user walked on the other floors too. Sequences are taken at the different days. The training sequence is taken at night and the test sequence is taken during the day. Trajectories for the dictionary are only chosen in the third floor. Figure 8 shows an approximate topological map of the third floor and trajectories chosen for the dictionary. These trajectories are chosen from the top of the list as described in Section 4. The order is 13, 9, 11, 4, 14, 5, 8, 6, 10, 16, 2, 1, 7, 15, 12, and 3. Trajectory 8 and 16 are the same, but taken at different visits in the training sequence. The dictionary is made by picking up 9 to 16 trajectories from the top of list.

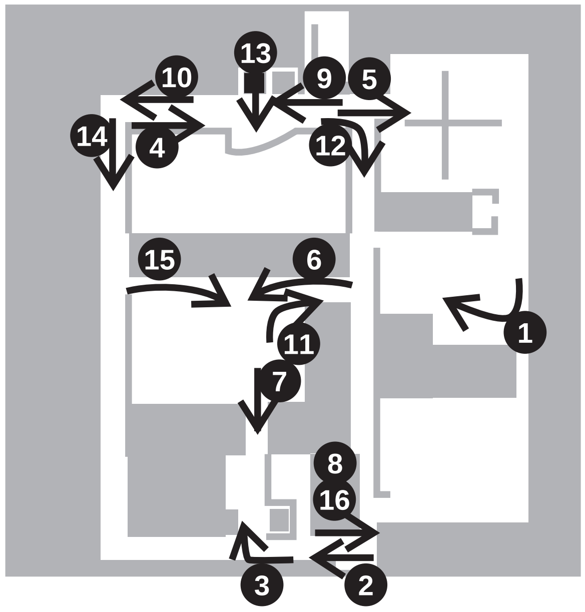

Figure 8 The floor plan and chosen trajectories.

Figure 9 shows the path of the test walk. The solid line indicates the first part of the test walk (3 minutes). The broken line indicates the last part (2 minutes 30 seconds). Between these two parts, the user went down to the ground level by stairs, walked around the ground level, went down to the lower level by stairs and came back to the third floor (2 minutes). These three parts are consecutive. Figure 10 shows detection result of the system. Numbers in normal letter indicate correct detection. Reversed letters indicate substitution error. Blank black box indicate deletion error. Numbers in box indicate insertion error. Blank areas are the gaps between two trajectories in the dictionary. For example, trajectory 6 on the first column is correctly detected at anytime. Trajectory 7 on the second column is correctly detected when the dictionary has 16 trajectories. When the dictionary has 15,14 or 13 trajectories, the system fails to detect (deletion error). The dictionary doesn't contain trajectory 7 when it has 12 or fewer trajectories, therefore, failures of detecting trajectory 7 are not considered as errors any more.

As for the second part of test walk (out of the third floor), nothing is detected as expected.

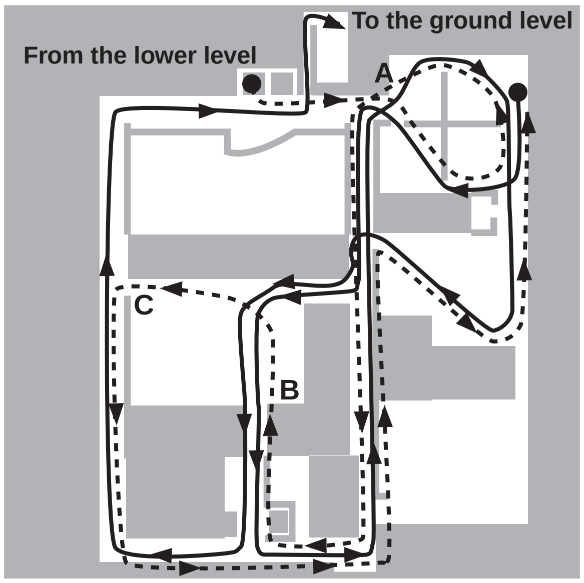

Figure 9 The test walk.

Table 1 shows the numerical evaluation of the system. We use similar evaluation as Starner et al. [7]. *T* and *G* are the total numbers of trajectories and gaps in the test sequence. *G* includes the gap between trajectories 4 and 13 (the second part of test sequence, of the third floor). *D* (deletions) is the number of trajectories not detected, *S* (substitution) is the number of trajectories falsely detected, *I* (insertions) is the count of false detection in gaps, which are not supposed to be detected. Accuracy Acc_T and Acc_G are given as follows:

$$Acc_T = \frac{T - D - S}{T}$$

$$Acc_G = \frac{G - I}{G}$$

#	*T*	*T-D-S*	*D*	*S*	A_{ccT}%
16	12	9	3	0	75.0
15	11	8	3	0	72.7
14	11	8	3	0	72.7
13	11	7	3	1	63.6
12	9	7	2	0	77.8
11	8	6	2	0	75.0
10	7	6	1	0	85.7
9	6	5	1	0	83.3

#	*G*	*G-I*	*I*	A_{ccG}%
16	12	2	10	16.7
15	12	1	11	8.3
14	12	5	7	41.7
13	12	6	6	50.0
12	10	4	6	40.0
11	9	4	5	44.4
10	8	4	4	50.0
9	7	4	3	57.1

Table 1 System performance (#: number of trajectories in the dictionary)

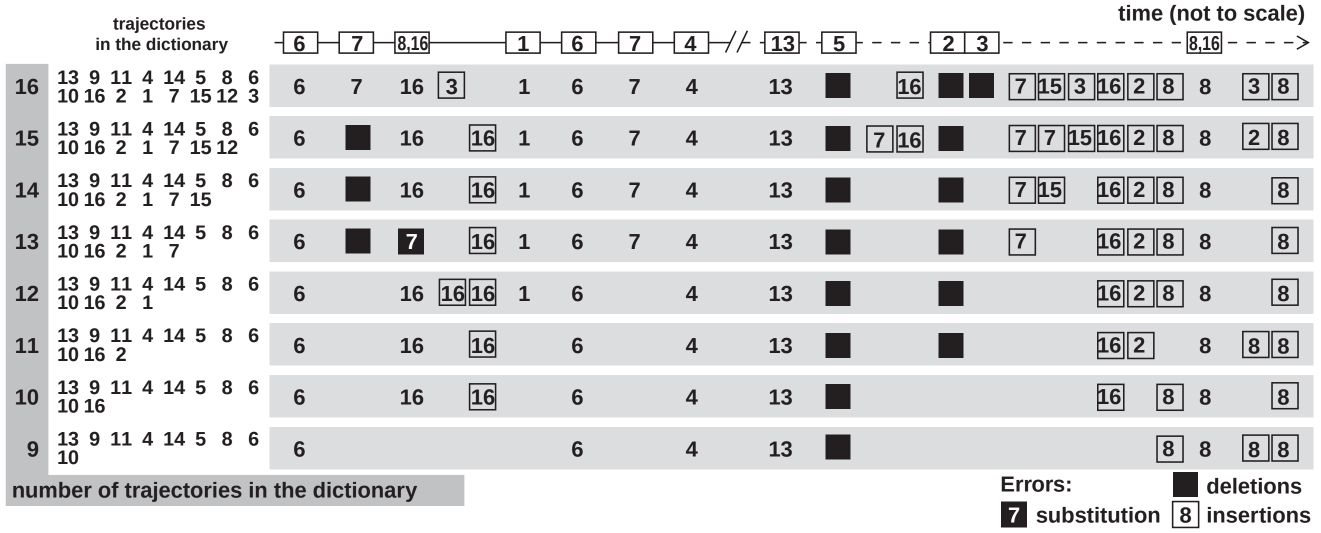

Figure 10 The results.

As described in [7], Acc_G can be negative when many insertions are detected. D and I are related to each other. When the parameters are set to minimize I, D goes up.

Table 1 shows that the system performs best when 9 or 10 trajectories are chosen for the dictionary.

As seen in Figure 10, location 5 has not been detected. This is because door A was open when training data was taken, and it was closed during testing. The system is confused when the user passes through from point B to C. This is because the color of the whole image is stable for 4-5 seconds in this area, and the path from B to C is similar to 6,7 and 15. The other confusion around 2,3,8,16 was caused by the same reason. There was no salient object in this hallway and this caused the mismatch. This means, 6,7,11 and 15 are very salient to any other segments except 6,7,11 and 15. The same holds for 2,3,8, and 16.

6. Results

We proposed a location recognition system which runs on a wearable computer. Recognition is done in realtime on a standalone PC, and the system doesn't require any wireless communication. The experiments showed that the system recognized locations in relatively good accuracy from the previous work. In addition to the proposed method, we suppose that the introduction of a topologic grammar will solve many problems especially during the insertions. Currently, we are also working on different means to automatically choose the location dictionary. We are considering more discriminative frame representations by changing adding frame features and/or using multiple subregions of each frame.

References

[1] M. Lamming and M. Flynn. *Forget-me-not: intimate computing in support of human memory.* In Proceedings of FRIEND21 International Symposium on Next Generation Human Interface. pp. 125-128. 1994.

[2] B. Rhodes and T. Starner. *Remembrance agent: a continuously running automated information retrieval system.* In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology (PAAM '96).

[3] M. Weiser. *The computer of the twenty-first century.* Scientific American, 1991.

[4] S. Kakez, C. Vania, and P. Bisson. *Virtually documented environment.* In Proceedings of the First International Symposium on Wearable Computers (ISWC '97).

[5] J. Rekimoto and K. Nagao. *The world through the computer: computer augmented interaction with real world environment.* ACM UIST'95, 1995.

[6] J. Rekimoto and K. Nagao. *Agent augmented reality: a software agent meets the world.* In Proceedings of Second International Conference on Multiagent Systems (ICMAS-96).

[7] T. Starner, B. Schiele, and A. Pentland. *Visual contextual awareness in wearable computing.* In Proceedings of the Second International Symposium on Wearable Computers (ISWC '98).

[8] R. Want and A. Hopper. *Active badges and personal interactive computing objects.* IEEE Trans. on Comsumer Electronics, 38(1):10-20, Feb. 1992.

[9] J. Orwant *For want of a bit the user was lost: Cheap user modeling.* IBM Systems Journal, 35(3), 1996.

[10] T. Starner, S. Mann, B. Rhodes, J. Healey, D. Kirsh, R.W. Picard, and A. Pentland. *Augmented reality through wearable computing.* Presence 6(4): 386-398, 1997.

[11] B. Schiele and J. Crowley. *Probabilistic object recognition using mutltidimensional receptive field histogram.* International Conference on Pattern Recognition (ICPR '96) B:50-54, 1996.

[12] B. Clarkson and A. Pentland. *Unsupervised clustering of ambulatory audio and video.* In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP '99).

[13] H. Aoki, B. Schiele, and A. Pentland. *Recognizing personal location from video.* In Proceedings of the Perceptual User Interfaces Workshop (PUI '98).

[14] H.Ney. *The use of a one-stage dynamic programming algorithm for connected word recognition.* Readings in Speech Recognition: 188-196. 1990.

[15] H. Aoki, S. Shimotsuji, and O. Hori. *A shot classification method of selecting effective key-frames for video browsing.* In Proceedings of ACM Multimedia 96: 1-10. 1996.

[16] H. Sakoe and S. Chiba. *Dynamic programming algorithm for spoken word recognition.* Readings in Speech Recognition: 159-165. 1990.