2006 IEEE International Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Smart Pantries for Homes

C. F. Hsu, H. Y. M. Liao, P. C. Hsiu, Y. S. Lin, C. S. Shih, *Member, IEEE*
T. W. Kuo, *Senior Member, IEEE* and Jane W. S. Liu, *Fellow, IEEE*

***Abstract*—A smart pantry holds non-perishable household supplies and automates the purchasing and delivery of their replenishments. By relieving its user from the chore of keeping the home stocked of essentials, it provides convenience and peace of mind not only to elderly individuals but also busy people of all ages. This paper describes two alternative smart pantry designs and tradeoffs. Underlying methods and technologies used for their implementation are also described.**

I. Introduction

In recent years, population of developed and developing countries is aging at a rapid rate [1, 2]. There is a growing need for low-cost, easy-to-use, and dependable devices and services designed to help the elderly live independently, improve their quality of life and reduce the cost of their care. Examples of these devices include object locators for finding misplaced household and personal objects and automatic and robotic helpers for enhancing physical dexterity and accessibility [3, 4]. Given the fact that the average percentage of population 65 or older will soon exceed the percentage of population under 15 in these countries, one expects that such devices may someday be as demanded as iPod, game consoles and robotic toys.

Another example of consumer electronics for the elderly is smart pantries for storage of nonperishable supplies. Such a pantry monitors its contents and automates the just-in-time replenishment of objects in it. Thus, it relieves of its user from the chore of keeping objects such as shampoo and detergent on hand. Smart pantries are for convenience of elderly individuals, as well as busy people of all ages who have no time or interest to shop (or place order) for boring but essential household supplies.

This paper describes two alternative smart pantry designs and implementations: the *picture-id version* and the *bar-code version*. The difference between the versions arises from differences in the technologies used for content capture and object identification. The picture-id version, called *PID pantry*, uses an overhead camera to capture its contents. In purchase orders sent by a PID pantry to the suppliers, each object to be delivered is specified by a picture captured by the camera prior to the removal of the object from the pantry. The supplier must process the photo image, either manually or automatically, in order to identify the brand and size of the object. For this reason, the usability of PID pantry is not ideal from the supplier's point of view. It is easy to use from owner's point of view, however. Other than making sure that nothing blocks the view of the camera, the user can treat a PID pantry just like a dumb pantry. The bar-code version, called *BAC pantry*, identifies objects in it by their bar codes. Because every object in every purchase order is unambiguously identified by a bar code, BAC pantries are easy to use from the supplier's point of view. On the other hand, the user must scan the bar code of every kind of objects in the pantry. Unless given a bar code before the supply of the kind runs out, the pantry will not be able to order replenishment automatically.

A natural question is why not RF identifiers (RFID). If every household object were to come with a smart tag, a pantry equipped with a RFID reader can easily maintain inventory as the user moves objects in and out of the pantry. This version would have the advantages of both PID and BAC versions and none of their disadvantages. In fact, this is how smart cabinets used in hospitals for storage of medical supplies work. Unfortunately, the RFID-version of smart pantries for home use is still not economically feasible and is likely to remain so for some time to come [5]. A cost of tens of cents per tag is low enough for tagging expensive medical supplies but is orders of magnitude too high for tagging ordinary household items individually.

Smart pantry is one of a family of appliances that are the research focus of the SISARL project [6, 7]. SISARL devices and appliances are consumer electronics of convenience, personal safety and health maintenance. Targeted users are elderly individuals who may have some functional limitations, but are still in relative good health, live independently and, most likely, in homes of their younger years. Like assistive devices and home care equipments, SISARL devices must be easy to use and highly dependable. However, assistive devices (e.g., [8 – 15]) typically assume a smart operating environment equipped with computer(s), Internet and, often, a variety of smart sensors; their targeted users, being in need of help in daily living, subscribe to assistive services, and so on. These assumptions are unrealistic for SISARL devices. To keep the costs of SISARL devices to a small fraction of the costs of typical assistive devices is another challenge.

C. F. Hsu is with Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu, Taiwan (e-mail: aldarishsu@yahoo.com.tw)

H. Y. M. Liao and J. W. S. Liu are with Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan (e-mail: {liao, janeliu}@iis.sinica.edu.tw)

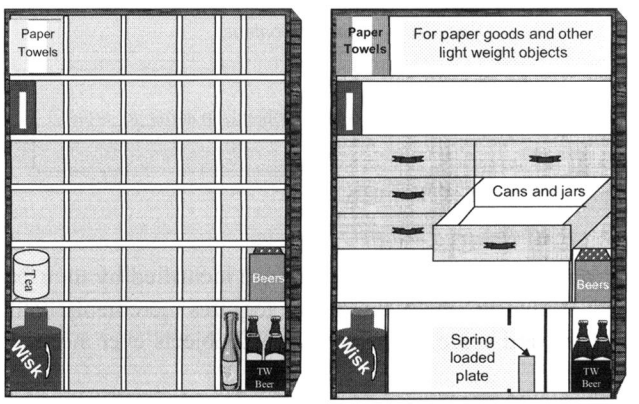
P. C. Hsiu, C. S. Shih and T. W. Kuo are with Department of Computer Science and Information Engineering, National Taiwan University, Taiwan (e-mail: {r91004, cshih, ktw}@csie.ntu.edu.tw)

Y. S. Lin is with Department of Computer Science, National Tsing Hua University, Taiwan (e-mail: yeushian@csie.nthu.edu.tw)

Following this introduction, Section II discusses assumptions and constraints common to both versions of smart pantry. Section III describes of their architectures and implementations. Section IV describes the techniques used by PID pantries for object identification purpose and summarizes preliminary experimental results. Section V describes wireless sensing schemes for BAC pantry. Section VI summarizes the paper and discusses future work.

II. COMMONALITIES AND DIFFERENCES

As stated earlier, a smart pantry is used to hold non-perishable household supplies. The pantry knows its contents and can automate their replenishments. For example, each pantry in Fig. 1 knows that a bag of paper towels is on the top shelf. When the last roll is removed, the pantry contacts a supplier of user's choice and requests the store to deliver a replacement bag.



(a) (b)

Fig. 1 Smart Pantries

An underlying assumption is that one or more grocery and discount stores have agreed to receive and process purchase orders sent by the pantry on the user's behalf and deliver each order by a specified date. (Alternatively, the panty may be provided to the user by a supplier.) The information required for contacting suppliers, placing orders and arranging payments and deliveries were entered into the pantry at initialization time, together with information on user preferences. By default, purchase orders are sent via messages over a dial-up connection, but a user with broadband internet access can configure the pantry to place orders via Internet.

Both versions are designed to accommodate concurrent insertions and retrievals of objects by multiple users. They use a keypad, a microphone, a speaker and a recorder to support user-pantry interaction. The microphone, speaker, and recorder form an audio interface that records voice segments of the user and plays back user voice interleaved with pre-recorded pantry voice. By allowing the pantry to interact with the user, the audio interface makes the pantry friendlier and more tolerant to misuse.

Like ordinary dumb pantries, the storage space in a smart pantry is divided into compartments. Figure 1 shows two configurations. The picture-id version is constrained to use the one in Fig. 1(a). The fact that each compartment is clearly defined by a rectangular boundary simplifies the extraction of pictures of individual compartments from a picture of the entire pantry. The bar-code version is constrained to use the configuration in Fig.1(b). In this configuration, shelves are not necessarily divided vertically. Rather, each compartment on a shelf corresponds to a switch that is in the front of the shelf and a spring-loaded plate that moves perpendicular to the shelf. (This construction is similar to the ones used in many drug and grocery store shelves.) When a plate is at the front of the shelf, as illustrated by the plate on the bottom shelf in the figure, it presses the switch closed, indicating that the corresponding compartment is empty. The compartment is nonempty when the plate is pushed towards the back of the shelf by an object, leaving the switch open. By sensing the states of the switches, a BAC pantry can distinguish empty compartments from non-empty ones.

Both versions require that objects in each compartment are identical. As we will see in the next section, a PID pantry cannot tell whether objects in two or more compartments are identical. By default, it will order replenishment when the last of the objects in any compartment is removed, even when some other compartments may contain more of the same objects. Moreover, it is constrained to order all objects from the same default supplier with the same *replenishment time* (i.e. the length of time-to-delivery interval). The bar-code version does not have these limitations.

III. ARCHITECTURES AND IMPLEMENTATIONS

In our discussions, we refer to compartments by 2-tuples of rows and columns. As an example, (3, 4) refers to the fourth compartment from the left on the third shelf from the top. We use *rows* and *columns* to mean numbers of rows and columns of compartments in the pantry.

A. PID Pantry Design

Fig. 2 and Fig. 3(a) show the physical components of a PID pantry. It consists of a base unit and a remote unit. The base unit contains a digital camera, together with the processing and storage modules that do most of the work. The base unit is mounted overhead so that the camera can capture a front view of the pantry shelves. The remote unit contains all the I/O devices and is within an easy reach of the user. It also provides access to the supplier(s). The units are connected wirelessly.

Fig. 4 describes the operations of the pantry. The pantry is empty during initialization. After capturing *current_picture* of the pantry, the pantry controller processes it to determine the boundaries of the compartments and values of *rows* and *columns* and captures and stores *EMPTY,* a picture of an empty compartment. (For the sake of simplicity, our discussion assumes that all compartments look alike when empty. This restriction can be easily removed.) It then allocates an array, called *picture*[*rows, columns*], to store pictures of individual compartments and initializes every element to *EMPTY.*

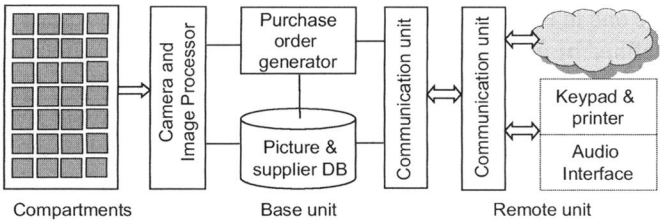


Fig. 2 Components of a PID pantry

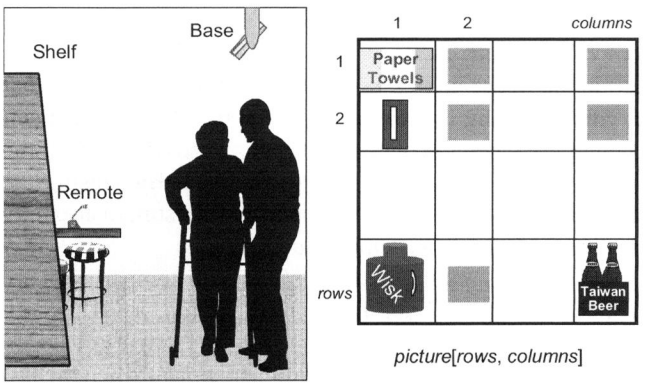


Fig. 3 Arrangement and Contents

During normal operation, the controller examines the contents of the pantry periodically (e.g., every 200 milliseconds) by having the camera take a snapshot of the pantry at the start of the period. When the controller finds that the new snapshot (*current_picture*) is essentially identical to the previous one (*previous_picture*), it does nothing. Otherwise, it extracts the picture *current_content* of every compartment (*i, k*) from *current_picture* and compares it with the one stored at *picture*[*i, k*]. Thus it determines whether the compartment has changed from being empty to non-empty and vice versa. In the former case, it stores the new picture of the compartment in *picture*[*i, k*] for use later. In the latter case, the compartment has just been emptied. The pantry inserts the picture stored at *picture*[*i, k*] into the list of objects to be ordered and then sets the element to *EMPTY*. The list thus generated can be printed and used as a shopping list if the user chooses to shop personally, rather than relying on the pantry, or sent by the pantry to a supplier as a purchase order. Fig. 3(b) shows what *picture* may contain at some time. The solid colored boxes indicate *EMPTY*. To ease the task of automatic object identification, the pantry also sends *EMPTY* in every purchase order.

We note that a basic PID pantry such as one described here merely extracts pictures of the objects in compartments. It cannot identify the objects. As we will see in Section IV, the object identification function for determining the brands and sizes of objects from their pictures requires significantly more processing power than what is needed to segment a big picture into smaller pictures. The function also requires a database of high quality pictures of objects to be identified. By off-loading this function to servers at the supplier side, the pantry is kept as simple as possible. (The base and remote units can be built from commodity digital camera and cordless phone, respectively.) On the other hand, because the pantry cannot distinguish objects from each other, it has the limitations mentioned in Section II.

```
Initialization:
   Initialize supplier information and user preferences;
   Command the camera to take current_picture of the pantry;
   previous_picture = current_picture;
   Process current_picture to identify compartment boundaries,
   Determine numbers of rows (rows) and columns (columns);
   Allocate array picture[rows, columns];
   Initialize elements of picture to EMPTY;
while pantry runs, periodically do {
    Command the camera to take current_picture of the pantry;
    if (current_picture != previous_picture) {
       for every compartment (i, k), do {
          Get current_content of (i, k) from current_picture;
          if (current_content != picture[i, k] ) {
             if (current_content == EMPTY) {
                Insert picture[i, k] to items_to_order list;
                picture[i, k] = EMPTY;
             } else {
                picture[i, k] = current_content;
             }
          }
       }
       Generate a purchase order for objects in items_to_order;
    }
}
```

Fig. 4 Operations of PID pantry

B. BAC Pantry Design

Again, objects in a BAC pantry are identified by their bar codes. With user's help, the pantry acquires incrementally the bar codes and voice descriptions of all objects ever stored in the pantry

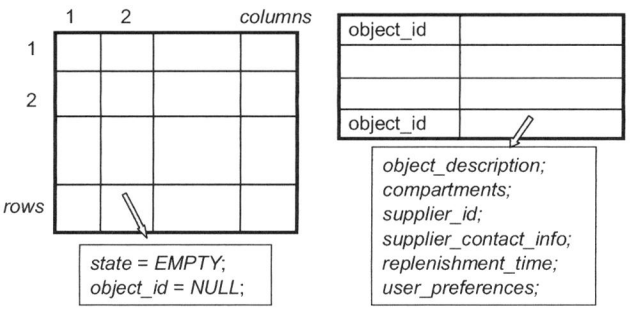


Fig. 5 Compartment and object descriptions

Some of the date structures maintained by the pantry are shown in Fig. 5. Each element of the *compartment array* on the left contains the state of a compartment and a pointer to the description of the object in it. The right half of the figure depicts the information on each object, which includes the bar code and a recorded voice description of the object. The user may choose to keep some kind of objects in multiple compartments and have the pantry order replenishment only when all the compartments holding them become empty. In that case, the *compartments* field gives pointers to structures of compartments holding the same kind of object. The user of a BAC pantry has the flexibility of ordering different kinds of objects from different suppliers and specifying different replenishment times. This is why the pantry maintains

supplier and user preference information on each kind of object individually.

The architecture of BAC version is similar to that of PIC version. A difference is, obviously, that the remote unit of a BAC pantry includes a bar code scanner. Another difference is that BAC version uses an array of switches to monitor the states (i.e., empty or non-empty) of the compartments. Whenever the state of a compartment (*i, k*) changes from empty to non-empty, the pantry controller acquires, with the user's help, the bar code and a voice description of the object just placed in the compartment. When the state of (*i, k*) changes from non-empty to empty, the controller puts the bar code of the object that was in the compartment in the *items_to_order* list, sends a purchase order to the supplier of the user's choice and then marks the compartment empty. Details on the architecture, operations and implementation of an interrupt-driven BAC pantry controller can be found in [4]. That implementation assumes that switches used to monitor compartment states are wired to the pantry controller interface. We will return in Section V to describe a ways to configure an existing dumb pantry into a smart one using wireless sensors.

Like the PIC version, multiple users of a BAC pantry can place objects and remove them in any order. Scenarios of user-pantry interactions putting objects in pantry and removing them can also be found in [4].

Errors during placements and removals are inevitable. When the user does not follow the normal sequence of scan and placement, some objects in the pantry may have no bar codes and some objects may have wrong bar codes. These errors are recoverable. An error of the former type is known to the pantry. It handles the error by asking the user to scan the object at the time of the removal. When an object has a wrong bar, the voice confirmation from the pantry during the removal process provides the user with an opportunity to discover the error and initiate a corrective action. An error is unrecoverable when it causes the pantry to fail in ordering correct replenishment in time. Unfortunately, unrecoverable errors can occur when the user ignores the voice from the pantry verifying with the help of the user the accuracy its knowledge about on the contents of the compartments.

IV. OBJECT IDENTIFICATION MODULE

To make PID pantries easy to use, a supplier add to its order processing server an object identification module (OIM) that processes pictures contained in purchase orders from smart pantries and returns as results the brands, sizes and locations of the objects. Roughly speaking, the module identifies the object in each picture by determining which image among all images of known objects in its database best matches the image it extracts from the input picture.

The OIM works faster and more accurately when the number of candidate images to match is small. Hence, the module maintains for each user a small repository of images of objects that have identifiers and are known to have been purchased by the user. When the user orders no new products, the OIM only needs to search the user's repository when it tries to identify objects in the input pictures. It searches the image database of all objects in the supplier's inventory only on rare occasions when the user orders something new.

A. Approach and Algorithms

The flow chart in Fig. 6 gives an overview of OIM operations. The major steps are background subtraction, low-level image processing, color matching and shape context-based search.

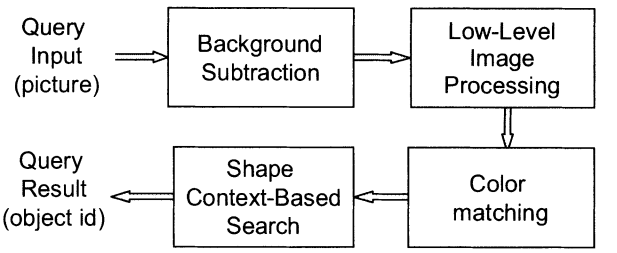


Fig. 6 Object identification operations

Background Subtraction (*BS*) The first step is to extract the foreground image of the object from the input picture of a non-empty compartment. For this work, the OIM uses as background the picture, *EMPTY,* of empty compartments, which the pantry sends in the purchase order. Let $B(x, y)$ and $I(x, y)$ be the values of the background image and the input image at pixel (x, y), respectively. A way to determine the value $f(x, y)$ of the foreground image at (x, y) is to let it be $I(x, y)$ if the absolute difference between $I(x, y)$ and $B(x, y)$ is larger than a preset threshold T; otherwise, $f(x, y)$ is equal to 0. The module uses this simple way to determine the values of most foreground pixels. It chooses the threshold T based on the statistics on past pictures sent by the pantry.

Low-Level Image Processing (*LLIP*) Because of shadow points and noises, the foreground image produced after the BS step may have fragments that originally belong to the same component. A goal of the LLIP step is to fix this problem. In this step, OIM carries out morphological operations (i.e., dilation followed by erosion) several times [16]. It then executes a connected-component-labeling process to label distinct objects in the foreground. It considers objects that are smaller than a threshold size as noises and removes them from the foreground image.

Shadow is another problem that must be dealt with. Some shadows may be of large enough sizes to be retained after the connected-component-labeling process. The object identification module applies a shadow removal algorithm described in [17] to remove them. The algorithm compares the hue, saturation and intensity values between the foreground and the background images and distinguish shadow pixels from the foreground pixels using the equation given below: In the equation, $I_k^V(x,y)$, $I_k^S(x,y)$, and $I_k^H(x,y)$ represent, respectively, the intensity, saturation, and hue values of a foreground pixel at (x, y); B_k^V, B_k^S, B_k^H represent these values of a background pixel at (x, y),

respectively; and α, β, τ_S, and τ_H are thresholds chosen on the basis of lighting condition and pre-determined statistics.

$$SP_k(x,y) = \begin{cases} 1 \;\; if \;\; \alpha \le \dfrac{I_k^V(x,y)}{B_k^V(x,y)} \le \beta \\ \wedge (I_k^s(x,y) - B_k^S(x,y)) \le \tau_S \\ \wedge \left| I_k^H(x,y) - B_k^H(x,y) \right| \le \tau_H \\ 0 \;\; otherwise \end{cases}$$

$SP_k(x, y) = 1$ represents that the pixel under consideration is a shadow pixel and is to be removed from the foreground image. Otherwise, the pixel remains in the image.

Color Matching (*CM*) The foreground object obtained after the LLIP step is a candidate for object recognition. The OIM uses a coarse-to-fine matching mechanism. For coarse level search, it works with two standard descriptors in MPEG-7 [18], dominant color and color layout, to perform object recognition. In this process, it calculates the dominant colors from R, G and B channels, quantizes 256 colors into 32 bins, and then distributes all pixels belonging to an object into these bins. The top 3 bins of each channel are picked to represent the channel for comparison. The comparison metric is Euclidean distance, and the weights assigned to the channels are identical.

Since the dominant color is a global feature, it does not carry any relational information. Therefore, the OIM also includes color layout and uses a quad-tree to express it. This way, a different object that has the same set of dominant colors or an identical object that is placed in different orientations will not be mistakenly chosen in the coarse search stage.

Shape Context-Based Search (*SCBS*) In addition to the dominant color and color layout, a detailed description of the foreground object to be recognized is required for the fine search process. For this purpose, the current version of OIM uses a shape context-based descriptor [19, 20] to characterize the shape of an object.

To compute the descriptor, the OIM first applies the Canny edge detector [21] to extract a silhouette of the target object. (The Canny type detector is one of the best existing edge detectors but cannot guarantee the full extraction of complete silhouette.) The OIM then selects *r* control points from the detected silhouette. The distance between every consecutive control point pair is almost equal except for the broken parts due to incompleteness of the silhouette. Fig. 7 shows an example. The top row shows the picture of an object. The picture in the left of the bottom row shows the almost complete silhouette produced by Canny edge detector. The picture in the right shows the *r* selected control points generated from the silhouette.

Next, the object identification module computes log-polar histograms corresponding to the *r* control points for each object that is to be compared. Each histogram characterizes the relationship among the *r* chosen control points of the object [19, 20]. To derive the histograms, the OIM uses a circle mask to cover every control point of the target object. It first divides the circle into 12 30-degree bins along the circular direction and then divides each bin in the radius direction into 5 bins equally according to the log of the radius. The circle mask after partition is shown in Fig. 8(a). The log-polar histogram of a control point can then be found by putting the center of a circle mask on the control point and calculating its log-polar histogram. Calculation of a log-polar histogram when a circle mask is placed at a control point is illustrated in Fig. 8(b).

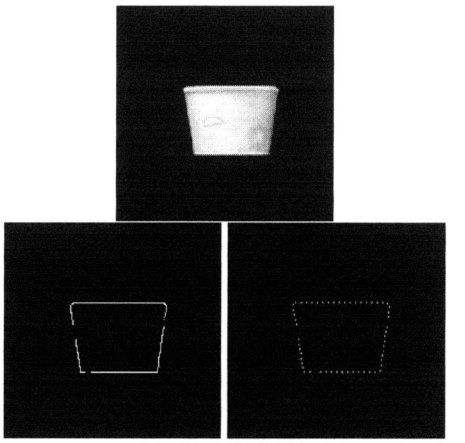


Fig. 7 Example on context-based descriptor

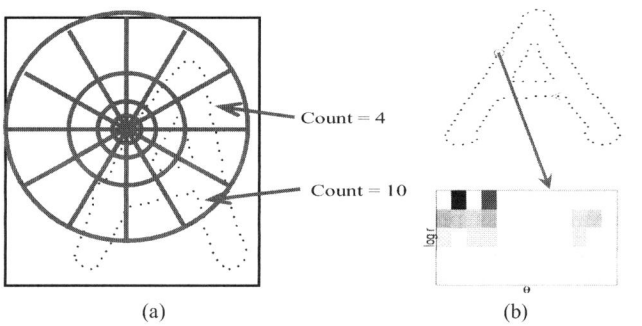


Fig. 8 A circle mask and a log-polar histogram

After computing sets of *r* log-polar histograms of the objects in pictures contained in a purchase order, the OIM puts them and pre-computed sets of r histograms of known objects contained in the user's repository into a bipartite graph. As the final step, the module calculates the degrees of matches according to Hungarian algorithm [22].

B. Preliminary Performance Results

In order to test the efficiency and effectiveness of the OIM, we constructed a database of the forty objects and a small pantry. The objects are shown in Fig. 9. In an experiment, we processed a purchase order containing pictures of five objects in ways described above. The picture on the left in Fig. 10 shows how the pantry looks when empty, and the picture on the right shows how the five objects looked in the pantry before they were removed.

Fig. 9 Objects in database

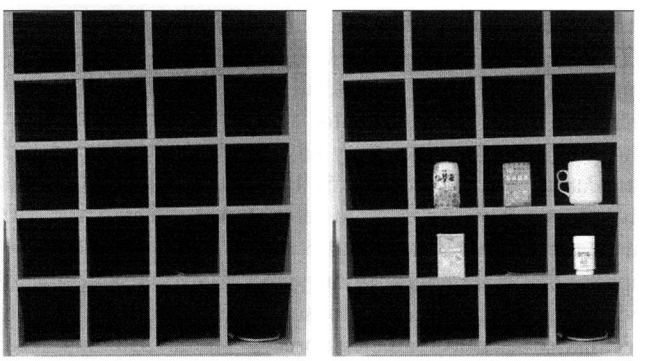


Fig. 10 Empty and loaded pantry

The images extracted by the OIM are in the leftmost column in Fig. 11(a). The remaining six columns in the figure give the top six (or fewer) candidate objects retrieved by the module for each input object after the coarse search (i.e., the CM step). Obviously, a coarse search does not eliminate enough candidates. However, the coarse search process is very fast, allowing the module to quickly screen the database and obtain a significantly smaller set of candidates. In the fine search process, the module applied the shape context features on candidate objects and obtained much more accurate results. Fig. 11(b) shows the results of fine search. The speed of the fine search process is slower than the coarse search because of the computation of shape context, but the time consuming process is applied to only a small number of candidates.

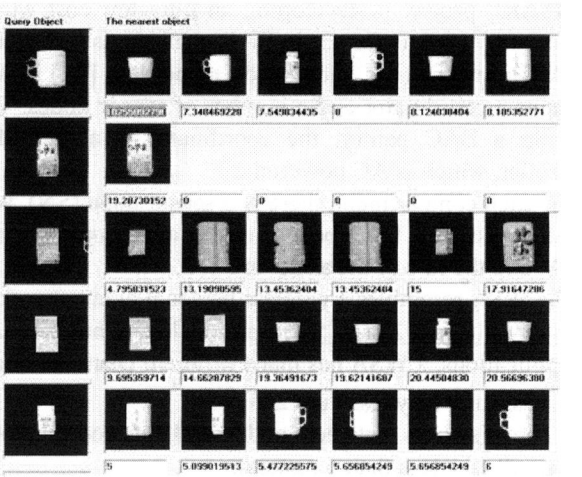


Fig. 11(a) Results of coarse search

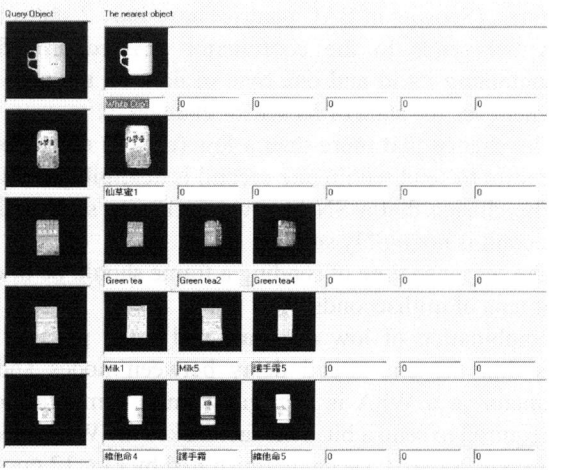


Fig. 11(b) Results of fine search

V. WIRELESS SENSOR ARRAY

In the BAC version described in Section 3 and [4], switches for monitoring compartment states are connected by wires to the sensor interface on the pantry controller I/O bus. This construction is suitable for pantries specially built to be smart. However, wiring up an existing dumb pantry to make it smart would be an unattractive option. A better alternative is to use wireless sensors, allowing the user to configure a dumb pantry (or a part of the pantry) into a smart BAC pantry as easily as a PID pantry.

A wireless binary sensing scheme for this purpose is described in [4]. That scheme uses passive RFID tags, one per compartment. The controller contains a reader. The shelves and spring-loaded boards are constructed so that the tag corresponding to a compartment is shielded from the reader when the compartment is empty but is visible to the reader when the compartment is non-empty. The controller determines the states of the compartments by reading the tags periodically. While the scheme works in principle, it is not ideal. Metal objects in the pantry can shield the reader from some visible tags. Solutions (e.g., use of multiple antennae) to this problem lead to added cost and installation difficulty.

SISARL project is developing an ultra-low-cost wireless sensor array (WSA) for use in BAC pantries, as well as other SISARL devices (e.g. medication dispensers [7].) A WSA contains a coordinator and a number of sensor nodes. When used in a BAC pantry, the coordinator is a part of the controller, which is AC powered.

For each compartment, there is a sensor node (SN), which is battery assisted. Let N be the number of sensor nodes in the WSA, and the id's of the nodes are 1, 2, …, N.

The WSA resembles a wireless personal-area network (see http://www.ieee802.org/15/) in physical size, but because of its applications, a WSA has many different characteristics:

1. N is small (say less than 265)
2. The distances between nodes and the coordinator are small (say less than 5 meters).
3. Each SN contains a sensor that has a small number of states. (Sensors in BAC pantries have only 2 states.)
4. A SN sends to the coordinator a fixed size frame containing its id and one byte of data on the new state whenever the state of its sensor changes.
5. The chance that more than a few (e.g., 3) nodes having frames to send within one second is negligibly small.
6. The chance that a SN has two frames to send within a second is negligibly small.
7. The response time of sending a frame should be in order of tens of milliseconds.

The combination of low data rate and small physical size means that the end-to-end delay between nodes and the coordinator in a WSA is negligibly small compared to the time required to send a bit. This fact makes the WSA medium access control (MAC) scheme illustrated by Fig. 12 possible.

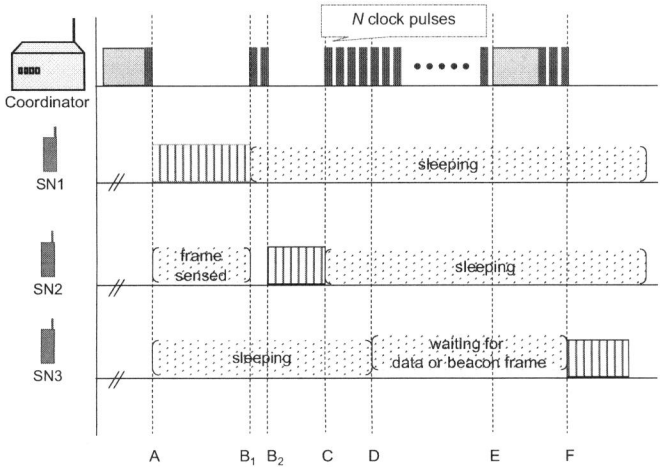


Fig. 12 MAC of WSA

According to the scheme, when no sensor node has frames to send, the coordinator continuously polls the nodes by sending a beacon frame followed by a sequence of clock pulses. (Beacon frames are depicted in Fig. 12 by light-colored square boxes on the top time line, and the clock pulses are depicted by narrow dark boxes.) After sending each clock pulse, the coordinator pauses to listen briefly. Hearing no data frame, it continues to send clock pulses. After it has sent the N-th pulse in the sequence, it repeats a beacon frame followed by clock pulses.

If the coordinator hears a data frame immediately after it sends a clock pulse, it switches to receive the data frame. At the end of the data frame, it sends clock pulses, listens in between pulses, switches to receive if it hears a data frame; otherwise, it sends a beacon frame after it completes the sequence of N clock pulses.

Medium access by the sensor nodes are prioritized according to their ids: the smaller the id, the higher the priority. Specifically, when a node with id k has a data frame to send, it waits until it hears a beacon frame or data frame and then counts the clock pulses after the end of the frame. If it hears a data frame before the kth clock pulse, it waits until the data frame ends and then counts clock pulses again. The node sends its data frame, whenever it hears the kth clock pulse in a pulse sequence after a beacon or data frame.

Fig. 12 illustrates the operations of sensor nodes with ids 1, 2, and 3. At the start, both SN1 and SN2 have frames ready to send. They wait until they hear the beacon frame from the coordinator. At time A, SN1 sends its data frame after it hears the first pulse following the beacon frame. Its transmission causes SN2 to wait. Since SN1 no longer has data frame to send, SN2 gets to send at time B_2 after hearing two clock pulses following the data frame from SN1. At time C, no node has data to send, and the coordinator begins to send a sequence of N clock pulses. Suppose that SN3 awakes at time D in the midst of the sequence. It must wait until it hears the next beacon and then starts to count. In this example, SN3 gets to send at time F after the third clock pulse.

To estimate the worst-case response time in a WSA, suppose that the time to send a data or clock pulse is 10 microseconds. The coordinator takes 20 microseconds to send a clock pulse. The array has 250 sensors. Beacon and data frames are 100 bits and 50 bits long, respectively. The times the coordinator takes to send a beacon frame and 250 pulses are 2 milliseconds and 5 milliseconds, respectively. Suppose that three lowest-priority nodes, SN248, SN249 and SN250, have data frames to send immediately after the coordinator starts to send clock pulses. The SN250 must wait for the time required to send 4 sequences of clock pulses, a beacon frame and two data frames. So, the worst case response time of SN250 is approximately equal to 25 milliseconds.

VI. SUMMARY

This paper describes the PID and BAC versions of smart pantry. One can get a PID pantry by adding a digital camera and pantry electronics to any dumb pantry that has compartments. The pantry owner can use it much like a dumb pantry when placing and removing objects. On the supplier's side, pictures in each purchase order sent by the pantry must be processed to identify the objects in the order and find their inventory control codes for locating the objects to be delivered. The process can be automated by the suppliers. The

paper describes an object identification module designed for this purpose.

From both the technical and usability points of view, the BAC version represents a reasonable compromise. The supplier can rely on the bar codes in purchase orders to identify and locate the objects to be delivered. However, the users must scan the content of each compartment at least once before the last object in it is removed. For users who are willing to follow this rule, a bar-code version is sufficiently user friendly and reliable.

Much work remains to be done to access the merits of the OIM and WSA described here. We are refining the algorithms used in the OIM so that it can accurately identify objects in low resolution pictures. The WSA is being prototyped. An assumption is that the transmission is sufficient error-free as to make error detection and recovery unnecessary. Some form of ARQ will be added if this assumption is found invalid in our evaluation.

ACKNOWLEDGEMENT

This work is partially supported by the Taiwan Academia Sinica thematic project SISARL: Sensor Information Systems for Active Retirees and Assisted Living. The authors wish to thank Professor A. C. Pang for her help in the design of MAC for WSA.

REFERENCE

[1] Japan Assistive Products Association, http://www.jaspa.gr.jp/, April 2003.

[2] "Global Aging," *BusinessWeek*, January 31, 2005.

[3] Jane W. S. Liu, *et al*, "Reference Architecture of Intelligent Appliances for the Elderly," *Proceedings of International Conference on System Engineering*, Las Vegas, NV, August 2005.

[4] Jane W. S. Liu, *et al.*, "User Scenarios and Designs of Smart Pantry, Object Locator and Walker's Buddy: Consumer Electronics for the Elderly," Technical Report No. TR-IIS-O5-OO7, Institute of Information Science, Academia Sinica, Taiwan, July 2005.

[5] R. Glidden, *et al.* "Design of Ultra-low-cost UHF RFID Tags for Supply Chain Applications," *IEEE Communications*, August 2004.

[6] http://sisarl.org. SISARL: Sensor Information Systems (and Services) for Active Retirees and Assisted Living,.

[7] P. H. Tsai, *et. al*, "Compliance Enforcement of Temporal and Dosage Constraints," Technical Report No. TR-IIS-06-006, Institute of Information Science, Academia Sinica, Taiwan, October 2005.

[8] S. Helal W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator Tech Smart House: A Programmable Pervasive Space," *IEEE Computer*, March 2005.

[9] E. Dishman, "Inventing Wellness Systems for Aging in Place," IEEE *Computer,* May 2004.

[10] A. Pentland, "Healthwear: Medical Technology Become Wearable," IEEE *Computer*, May 2004.

[11] D. A. Ross, "Cyber Crumbs for Successful Aging with Vision Loss," *IEEE Pervasive Computing*, April 2004.

[12] http://architecture.mit.edu/house, Changing Places Consortium, MIT.

[13] http://www.cc.gatech.edu/fce/seminar/fa98-info/smart_homes.html, FCE Smart House Research Survey, Georgia Tech.

[14] http://www.futurehealth.rochester.edu/, Center for Future Health, University of Rochester.

[15] http://marc.med.virginia.edu/projects_smarthomemonitor.html, Marc smart home, University of Virginia.

[16] J. C. Russ, *The Image Processing Handbook, Fourth Edition*, CRC Press, July 2002.

[17] A. Prati, I. Mikic, C. Crana, M. M. Trivedi, "Shadow Detection Algorithms for Traffic Flow Analysis: A Comparative Study." *IEEE Conference on Intelligent Transportation System,* Oakland, California, pp.340-345, 2001.

[18] Y. Deng, B. S. Manjunath, C. Kenney, M. S. Moore, H. Shin, "An Efficient Color Representation for Image Retrieval." *IEEE Transactions on Image Processing,* Vol.10, No.1, pp. 140-147, January 2001.

[19] S. Belongie, J. Malik, J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.24, No.4, April 2002.

[20] G. Mori, S. Belongie, J. Malik, "Efficient Shape Matching Using Shape Contexts." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No.11, pp. 1832-1837 November, 2005.

[21] J. Canny, "A Computational Approach Edge Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.8, No.6, pp.679-698, November 1986.

[22] R. C. Gonzales and R. W. Woods. *Digital Image Processing*, Addison-Wesley Publishing Company, 1992.