

# Dietary-Aware Dining Table – Tracking What and How Much You Eat

Keng-hao Chang<sup>a</sup>, Shih-yen Liu<sup>b</sup>, Jr-ben Tian<sup>a</sup>, Hao-hua Chu<sup>a,c</sup>, Cheryl Chen<sup>d</sup>

Department of Computer Science and Information Engineering <sup>a</sup>,  
Department of Information Management <sup>b</sup>,  
Graduate Institute of Networking and Multimedia <sup>c</sup>,  
School and Graduate Institute of Nursing <sup>d</sup>,  
National Taiwan University  
r93018@csie.ntu.edu.tw, b90701219@ntu.edu.tw, {r93045, hchu}@csie.ntu.edu.tw;  
cheryl.chen@ha.mc.ntu.edu.tw

**Abstract.** We are what we eat. Our everyday food choices affect our long-term and short-term health. In this paper, we have designed and implemented a dietary-aware dining table that can track what and how much we eat. To enable automated food tracking, the dining table is augmented with two layers of weighting and RFID sensor surfaces to detect and recognize multiple, concurrent person-object interactions occurring on the table.

## 1 Introduction

We are what we eat. Research continues to confirm that nutrition profoundly influences many chronic illnesses, and it represents one of the most accessible means for prevention and intervention to reduce health risk and promote well-being [5]. Food choice affects our health in many ways. The vast majority of population has chronic illnesses such as heart disease, diabetes, hypertension, dyslipidemia, and obesity in which proper dietary intake and related interventions have been demonstrated to be effective in ameliorating symptoms and improving health.

Unlike the traditional health care in which professionals assess and weigh each individual's dietary intake, and develop a plan for behavior changes, ubiquitous healthcare technologies provide an opportunity for individuals to quantify and acknowledge their dietary intake in the point of living without intensive labor cost. Such technologies provide a mean for individuals to proactively monitor their food as well as water intake and act upon it, leading to a better food selection and sensible eating.

In this paper, we propose a dietary-aware dining table that can automatically track what and how much each individual eats from the dining table over the course of a meal. This is in accordance to the *vision of disappearing computers*, where computing HW & SW are hidden into everyday object (i.e., dining table) and remain *invisible* to human users. There is *no* digital access devices (such as cell phones, PDAs, or PCs) needed for human users to interact with in order to use this digital dietary service. In comparison, the traditional dietary tracking software requires the human users to

recall the amount of foods consumed, and then manually type them into computers. This is less accurate than our automated approach due to mistakes in visual measurement and imperfect memory. More importantly, the traditional method requires explicit human efforts to operate digital devices.

We have augmented a dining table with two layers of *sensor surfaces* underneath the table – the *RFID surface* and the *weighting surface*. As shown in Fig. 2, these sensor surfaces are divided into 3x3 *sensor cells*. Within each sensor cell contains a RFID reader/antenna and a load sensor. The RFID surface serves two functions: (1) it enables identification and label of RFID-tagged tabletop objects; and (2) it can track *cell locations* of these objects through their RFID tags. The weighting surface measures the amount of weight transfers across different tabletop containers located on different weighting cells, as servings of food are transferred between different tabletop containers. By combining the RFID and weighting surfaces, our system can trace the complete *food movement path* from its source tabletop container to other tabletop containers, and eventually into the individual mouth.



**Fig. 1.** Typical dining table setting in a Chinese family.

Our dietary-aware dining table supports many people simultaneously sharing a meal on the same dining table. Fig. 1 shows a typical meal setting in a Chinese family – the family members sit around a circular table with the main dishes placed in the center and individual rice bowls and plates arranged on the table periphery. Each participant first uses shared utensils to transfer servings of food from the main dishes to his/her personal plate or rice bowl, and then eats from there. In this dining scenario, multiple table participants are continuously and concurrently engaging in the food transferring and eating motions. This creates *multiple, concurrent person-object interactions* (objects are tabletop objects such as plates, bowls, etc.) from which a single table surface needs to observe, track, and infer high level interaction semantics. This is the *main technical challenge* addressed in this paper – how to design a sensor-embedded tabletop surface to track food consumption from each of many table participants.

To our knowledge, we have not found any work that attempts to address complex, simultaneous person-object interactions from a load sensing surface. This work is believed to be the first to augment the load sensing surface with a RFID surface to enable tracking of multiple, simultaneous person-object interactions over a tabletop surface. The closest system to our work is the load sensing table [1] from Lancaster University. They have utilized four load cells installed at four corners of a rectangular table to acquire positional information of tabletop objects, and infer interaction events

such as adding or removing an object from the surface, or knocking an object over. They have demonstrated success with these interaction events. However, their main limitations are *complex, simultaneous interactions involving multiple objects*. For example, their positioning algorithm would fail if two or more objects are moved simultaneously on the tabletop surface. In comparison to our work, such complex, simultaneous interactions are expected to be relatively common in our family dining scenario; therefore, they are the target of our work. There are other related but less relevant works that apply load sensing to derive context information. Smart floor [4] has demonstrated that by applying pressure sensors underneath the floors, it is possible to identify users and to track their locations. The posture chair by Selena [3] has deployed two matrices of pressure sensors (called *pressure cells*) on a chair to recognize postural behaviors of a child, and then infer his/her affective interest level.

The remainder of this paper is organized as follows. Section 2 describes our design, implementation, and limitations. Section 3 draws our conclusion and future work.

## 2 Design and Implementation

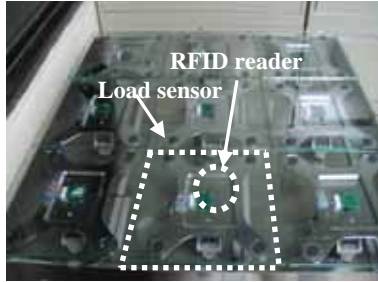
We first describe our system hardware design and implementation, followed by our software proof-of-concept design. Then we propose two new designs by adding new functions to reduce limitations in the first design.

### 2.1 Hardware design and implementation

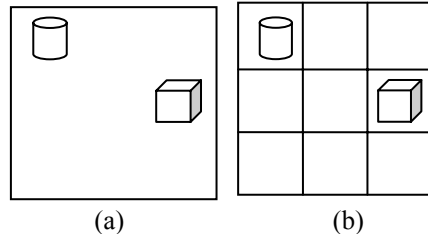
Our current tabletop prototype has a dining surface of  $90 \times 90 \text{ cm}^2$ , which is about the size of a small dining table. To detect multiple, concurrent person-object interactions on the tabletop surface, the tabletop surface is divided into an array of  $3 \times 3$  cells, each with the size of  $30 \times 30 \text{ cm}^2$ . Each cell contains a load sensor and a passive RFID reader/antenna as shown in Fig. 2. The RFID reader is the Skyetek M1 [2] RFID reader with an average read range of  $30 \text{ mm}$ . The load sensor is attached to a weight indicator with a resolution of  $0.5 \text{ gram}$  and can output weight readings through a serial port at a frequency of 8 samples per second.

The reason for splitting the weighting surface into cells can be described in Fig. 3-(a). If users were to put a cylinder object and a cubic object simultaneously on this weighting surface, it could only sense their aggregate weight. In other words, it could not determine the individual weight of the cylinder or the individual weight of the cubic. This is called the *single-cell-concurrent-interactions problem* – it is impossible to distinguish multiple, concurrent person-object interactions on the surface with the weight information from only one load sensor (in the Lancaster’s approach, the scale is made up of four load sensors at four corners of a table). To address this limitation, our solution is to divide the surface into multiple cells shown in Fig. 3-(b). When cells are small enough, it is likely that each tabletop object would occupy a different cell. Therefore, our solution can utilize multiple weight sensors at different cells to distinguish the weight of the cylinder object from the cubic object. This idea can be generalized as follows: the larger the weighting cell relative to the average size of objects, the higher the likelihood that multiple, concurrent person-object interactions

can occur within the same cell; therefore the higher the probability of single-cell-concurrent interactions. To reduce this probability, we divide the weighting surface into cells of an appropriate size that just fits the average size of tabletop food containers, such as plates, bowls, etc.



**Fig. 2.** The setup of the weight surface.



**Fig. 3.** It illustrates why a multi-cells surface can track the multiple objects, but a single-cell surface cannot.

In order to support multiple, simultaneous person-object interactions, we have found that using a weighting surface alone is insufficient, because it is difficult to distinguish a tabletop object from its weight that may change over the course of a meal (i.e., as people transfer weights between food containers). Therefore, we augment the weighting surface with a passive RFID surface to help identify tabletop objects. Within each cell contains a RFID reader/antenna that can read off unique IDs from RFID-tagged tabletop objects put on top of that cell.

## 2.2 Software design and implementation

In the following subsections, we describe our design in three different stages. The 1<sup>st</sup> stage targets a simple proof-of-concept prototype. We make the following two simplifying assumptions. (1) The food movement paths from the food containers to individuals' mouths are pre-determined and static. (2) Each tabletop object cannot cross multiple cells as shown in Fig. 5.

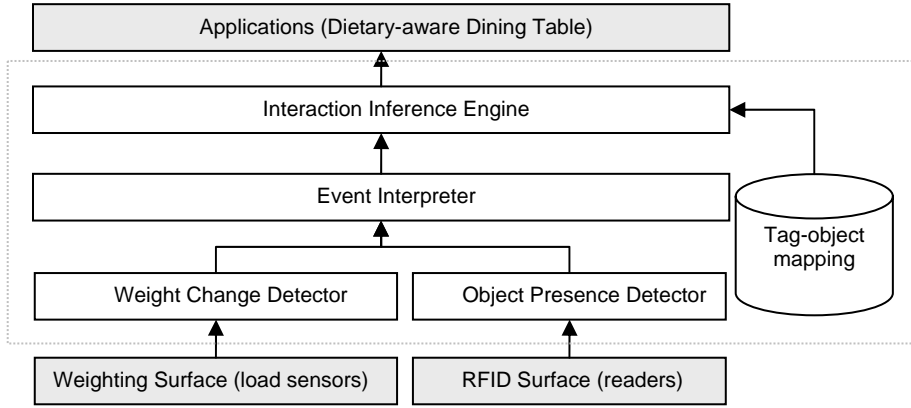
Based on these two assumptions, we have come up with an initial design that applies our multi-cells weighting and RFID surfaces to detect multiple, concurrent person-object interactions. We have further proposed two designs: the first one removes the first assumption to detect different possible food movement paths, and the second one further removes the second assumption to enable the detection of cross-cell person-object interactions.

### 2.2.1 Proof-of-concept prototype : concurrent multiple person-object interactions

The goal of the initial design stage is to create a simple proof-of-concept prototype to show that our system can detect multiple, concurrent person-object interactions under the two simplifying assumptions mentioned in the previous section: *no cross-cell objects* and *predetermined food movement paths*. The target scenario consists of two

people enjoying an afternoon cake and cocktail shown in Fig. 5. The cake plate and the cocktail container are both labeled with unique RFID tags and placed in the center of the table *on separate cells*. Two persons have their own cups and personal plates labeled with unique RFID tags and also placed on separate cells in front of their sittings. The food movement paths for the cake and cocktail are predetermined: the cake’s food movement path is cake-plate  $\rightarrow$  personal-plate  $\rightarrow$  mouth, and the cocktail’s food movement path is cocktail-container  $\rightarrow$  personal-cup  $\rightarrow$  mouth. In our two later designs described in the following subsections, these two assumptions will be relaxed one by one.

To realize our target scenario, we need to detect complex person-object interactions, such as transfer-cake, eat-cake, transfer-cocktail, drink-cocktail, etc. In order to infer the above inter-object interactions, our system collects and interprets low-level sensor data from weighting and RFID surfaces. Fig. 4 shows the overall system architecture. We describe each component in a bottom-up fashion as follows.



**Fig. 4.** System architecture

The *weight change detector* performs the following three tasks: (1) it aggregates weighting samples collected from each of the 9 load sensors; (2) it detects *weight change events* by filtering out noises in the stream of weight samples; and (3) it attaches timestamps to the detected *weight change events* and reports them to the *event interpreter*. The *object presence detector* performs similar tasks: (1) it continuously checks for presence and absence of RFID-tagged tabletop objects within each of the 9 RFID reader cells (2) it infers two object presence events: *put-on-cell* and *remove-from-cell*, and (3) it attaches timestamps to detected events.

The *event interpreter* interprets two *intra-object-weight-change* events, as shown in Table 1. The event interpreter builds internal states and internal events using inputs from the weight change detector and the object presence detector, and then interprets *intra-object-weight-change* events. Table 2 shows the rules to interpret events. For example, the *weight-increase(object<sub>i</sub>, w)* event represents that the *object<sub>i</sub>*’s weight is increased by *w*. This *weight-increase* event can be inferred from two possible internal observations: (1) *cell<sub>j</sub>* gains an additional weight *w* while *object<sub>i</sub>* stays on *cell<sub>j</sub>* and (2)

$object_i$  is put back on the table with a weight gain  $w$  from the last time  $object_i$  was put on the table.

**Table 1.** Intra-object weight-change events, internal events, and internal states.

Events	Descriptions
$Weight-increase(object_i, w)$	$Object_i$ 's weight is increased by $w$
$Weight-decrease(object_i, w)$	$Object_i$ 's weight is decreased by $w$
Internal Events	Descriptions
$Put-on(object_i, cell_j, w)$	$Object_i$ is put on $cell_j$ with weight $w$
$Put-away(object_i, cell_j, w)$	$Object_i$ is away from $cell_j$ with weight $w$
$Weight-gain(object_i, cell_j, w)$	$Object_i$ stays on $cell_j$ with weight gain $w$
$Weight-loss(object_i, cell_j, w)$	$Object_i$ stays on $cell_j$ with weight loss $w$
Internal State	Descriptions
$Weight(object_i, w)$	$Object_i$ has weight $w$

**Table 2.** Sample rules to recognize intra-object weight-change events.

Event Interpretation Rules
$Put-on(object_i, cell_j, w) \cup Put-away(object_i, cell_j, w) \rightarrow Weight(object_i, w)$
$Weight(object_i, w_1) \cap Weight-gain(object_i, cell_j, w_2) \rightarrow Weight(object_i, w_1+w_2)$
$Weight(object_i, w_1) \cap Weight-loss(object_i, cell_j, w_2) \rightarrow Weight(object_i, w_1-w_2)$
$Put-on(object_i, cell_j, w_1) \cap Weight(object_i, w_2) \cap w_1 > w_2 \rightarrow Weight-increase(object_i, w_1-w_2)$
$Weight-gain(object_i, w) \rightarrow Weight-increase(object_i, w)$

The *interaction inference engine* infers *inter-object interactions* initiated by the user  $u$  shown in Table 3. For example,  $transfer-cake(u, w)$  means that a serving of cake with a weight  $w$  has been transferred from the cake-plate to the user  $u$ 's personal plate. This interaction event can be inferred by first observing a weight decrease  $w$  in the *cake-plate*, followed by an equivalent weight increase  $w$  on the user  $u$ 's  $object_{i2}$ . The tag-object mappings provide two relations:  $Is(object, C)$  shows the type ( $C$ ) of the *object*, such as personal-plates or cake-plates, and  $Owner(object, u)$  tracks the owner ( $u$ ) of the *object*.

**Table 3.** Inference rules for inter-object interactions

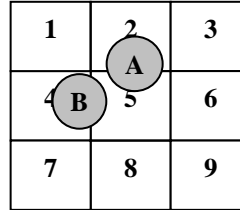
Interactions	Interaction Inference Rules
$Transfer-cake(u, w)$	$Weight-decrease(object_{i1}, w_1) \cap Weight-increase(object_{i2}, w_2) \cap Is(object_{i1}, cake-plate) \cap Is(object_{i2}, personal-plate) \cap Owner(object_{i2}, u) \rightarrow Transfer-cake(u, w_1)$
$Eat-cake(u, w)$	$Weight-decrease(object_i, w) \cap Is(object_i, personal-plate) \cap Owner(object_i, u) \rightarrow Eat-cake(u, w)$
$Transfer-cocktail(u, w)$	$Weight-decrease(object_{i1}, w_1) \cap Weight-increase(object_{i2}, w_2) \cap Is(object_{i1}, cocktail-container) \cap Is(object_{i2}, personal-cup) \cap Owner(object_{i2}, u) \rightarrow Transfer-cocktail(u, w_1)$
$Drink-cocktail(u, w)$	$Weight-decrease(object_i, w) \cap Is(object_i, personal-cup) \cap Owner(object_i, u) \rightarrow Drink-cocktail(u, w)$

Based on our experience with the initial design prototype in Fig. 5, it works well under two limitations: (1) The food movement paths from the food containers to

individuals' mouths are pre-determined and static and (2) no cross-cell person-object interaction (meaning that objects have to be placed strictly in separate cells). Next, we propose a new design to accommodate flexible food paths.



**Fig. 5.** The dining scenario for the proof-of-concept system: two users enjoying a cake and cocktails.



**Fig. 6.** Illustration of cross-cell problem: the container *A* is overlapped on the same cell #5 as container *B*.

### 2.2.2 Design proposal for flexible food path

We extend *interaction inference engine* in Fig. 4 to remove the *predetermined food movement paths* assumption. In our proof-of-concept design, the food source for each personal container is predefined. But in the real world scenarios, there are often different foods on the table, meaning that multiple food sources can be transferred to the same personal container at different times. For example, the weight-increase to a cup may be contributed by pouring of coke, juice, or tea.

To track a food movement path from the food's source containers to personal containers, we design a *weight matching model*. The basic idea is to match a weight-decrease from one container to a complementary weight-increase from another container within a certain time period. This matching process can be thought as a *hop of food transfer* from the source food container in the center of the table, to the personal containers on the table periphery. Based on this idea, we would implement this weight matching model by maintaining a queue of recent *object-weight-change* events. When a new weight-change event is detected, our model applies a matching function to compute a match probability with each existing weight-change event in the queue. This probability is computed based on two factors: (1) *weight*: the difference between the weight-decrease and the weight-increase pairs should be small; and (2) *timestamp*: the elapsed time between the weight-increase and weight-decrease events pairs should be short. We select an existing event on the queue that has the highest matching probability as a possible matching candidate. This highest matching probability must be above the threshold for successful match; otherwise, the new event remains unmatched and waits in the queue.

### 2.2.3 Design proposal for cross-cell interaction

We extend the *event interpreter* in Fig. 4 by removing the *no cross-cell objects* assumption. In a typical dining scenario, a randomly placed food container is likely to

cross multiple cells, such as containers *A* and *B* in Fig. 6. In more complex situations, multiple food containers may overlap across the same cell, such as cell#5 is occupied by both containers *A* and *B*. When a weight change is detected on cell #5, the question is whether this weight change is due to (1) container *A*, (2) container *B*, or (3) both containers *A* and *B*. To answer this question, our approach is to check whether weight changes on adjacent cell #2 or on cell #4. If the weight change is due to container *A*, a weight change event must also occur concurrently in the adjacent cell #2, because container *A* occupies both cells #2 and #5. In this case, the weight change of container *A* is the sum of weight changes in cells #2 and #5. Similarly, if the weight change is due to container *B*, a weight change event must also occur concurrently in cell #5. Moreover, if the weight changing event is due to both containers *A* and *B*, weight change events must also occur concurrently in both cells #2 and #4. We can approximately split the amount of weight changes on cell #5 to each of containers *A* and *B* based on the ratio of weight changes measured on cell #2 and cell #4.

### 3 Conclusion and future work

This paper describes the design and implementation of our dietary-aware dining table and two additional design proposals. We have augmented a regular dining table with two layers of sensor surfaces underneath the table – the RFID surface and the weighting surface. The dietary-aware dining table can automatically track what and how much each individual eats from the dining table over the course of a meal. For our future work, we are planning to implement these two design proposals and provide a thorough evaluation in real dining scenarios.

### Acknowledgements

We would like to thank Taiwan NSC and Quanta Computer for their general supports under NSC grants #93-2213-E-002-088, #93-2218-E-002-146, and #93-2622-E-002-033.

### Reference

1. A. Schmidt, M. Strohbach, K. Van Laerhoven, A. Friday and H.-W. Gellersen. "Context Acquisition based on Load Sensing". In Proceedings of Ubicomp 2002, G. Boriello and L.E. Holmquist (Eds). Lecture Notes in Computer Science, Vol 2498, ISBN 3-540-44267-7; Springer Verlag, Gothenburg, Sweden, September 2002
2. Skyetech RFID engineering, <http://www.skyetek.com/index.php>
3. S. Mota and R. W. Picard (2003), "Automated Posture Analysis for Detecting Learner's Interest Level." Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction, CVPR HCI, June, 2003.
4. R. J. Orr and G. D. Abowd. The Smart Floor: A Mechanism for Natural User Identification and Tracking. Gvu Technical Report GIT-GVU-00-02, 2000.
5. Rosenberg, I. H. (1996). Nutrition research: an investment in the nation's health. Nutrition Review, 54, s5-s6.