García Sanjuan, F.; Jaén Martínez, FJ.; Catalá Bolós, A. (2013). Evaluating heuristics for tabletop user segmentation based on simultaneous interaction. Expert Systems with Applications. 40(14):5578-5587. doi:10.1016/j.eswa.2013.04.011.

Evaluating Heuristics for Tabletop User Segmentation Based on Simultaneous Interaction

Fernando Garcia-Sanjuan, Javier Jaen, Alejandro Catala

{fegarcia, fjaen, acatala}@dsic.upv.es

Grupo ISSI, Departamento de Sistemas Informáticos y Computación (DSIC)
Universitat Politècnica de València
46022 Valencia (SPAIN)
Phone: +34 96 387 35 69

Abstract

Differentiating between users that interact on a tabletop could be beneficial for collaborative tasks to support territoriality-oriented features such as a more efficient space management or a better presentation of the contents. In this paper, we design a novel algorithm for the user differentiation or segmentation based on the simultaneous manipulation of controls. This is a potential differentiating factor that has remained unexplored so far, and in combination with other factors may become relevant to successfully accomplish such differentiation task. Basically it relies on the idea that users manipulate digital elements with a single hand, and hence, if two controls are being used at the same time, they most likely belong to different users. On the generic algorithm, three different versions have been implemented that include several heuristics to address the problem. The comparison under a simulated experiment shows that the heuristic involving more knowledge on distances on user controls performed better according to different goodness functions. This shows promising to further development and refinement of the approach by expanding it with other potential factors to eventually build a robust user differentiation subsystem.

Keywords

User differentiation, Heuristics, Tabletop, Collaboration

1 Introduction

Interactive tabletops provide a natural way of working collectively and several studies have shown their potential for collaborative activities (Dillenbourg & Evans, 2011) (Hornecker, Marshall, Dalton, & Rogers, 2008) (Morris, Paepcke, & Winograd, 2006). As stated in (Decouchant, Mendoza, Sánchez, & Rodríguez, 2013) systems intended to support collaboration should adapt themselves, for instance, in terms of the state of the activities, available resources and collaborators' location. This context-awareness interaction is relevant to provide a more user-oriented interface by considering the context of use and the context of user interfaces (Seo & Lee, 2013a) (Seo & Lee, 2013b). However, although multiple users are

involved in many of the tabletop-based tasks, applications are not usually aware of who is interacting or which controls belong to which users. Therefore, they do not take into account this valuable information that could be useful to mitigate some typical issues that sometimes arise in these interfaces or provide some enhanced features. For example, a specific issue is the surface clutter when there are many controls on a limited area. Some refined strategies could be devised to reduce the clutter by taking into account the ownership of controls. An option could be rearranging the interface elements around their owner in order to obtain a better layout of the elements and make more space available (for instance, moving away the elements that have not been used in a while). This rearrangement would be motivated by two main observations. Firstly, as reported in (Scott, Sheelagh, Carpendale, & Inkpen, 2004) and (Tse, Histon, Scott, & Greenberg, 2004) users tend to territorialize their working space and almost exclusively use the area in front of them. Secondly, users prefer working with their own UI elements, fact that was described in (Morris, Paepcke, Winograd, & Stamberger, 2006), showing that even if users can share the controls, they choose to replicate them. To successfully involve all these ideas, it is essential to differentiate the users owning the controls.

In general, being aware of who is using the surface would allow the development of user interfaces that automatically adapt to the way users interact in terms of content visualization and space management. Therefore, studying the way to differentiate users is interesting because it could open new design considerations on how the content can be better delivered on these multi-user tabletop interfaces and eventually improve the quality of the designed tabletop-based applications.

Researchers like Harrison et al. (Harrison, Sato, & Poupyrev, 2012) and Zhang et al. (Zhang, Yang, Ens, Liang, Boulanger, & Irani, 2012) have specified some desirable qualities that user differentiation systems should have. According to these authors, they should avoid requiring wearable elements and minimize or suppress the peripheral hardware. They should also be fast, robust (allowing the correct distinction between a variable number of users interacting both sequentially and simultaneously), transparent (not distracting the users from their tasks) and, finally, cheap, easily deployable, compact and with low power requirements.

In this paper, we focus on describing a factor that has not yet been considered: the simultaneous use of different controls, which in combination with other factors may increase the robustness of the ownership of controls assignment process. When interacting with digital media, users tend to use one single hand even though multi-touch surfaces allow bimanual interactions (Terrenghi, Kirk, Sellen, & Izadi, 2007). Therefore, for some situations it could be thought that, if two UI elements are being manipulated at the same time, they most likely belong to different users. In addition, some co-located collaborative activities have shown a certain trend to parallelization of tasks while users interact orally with one another (Morris, Lombardo, & Wigdor, 2010). Therefore, it could be possible to effectively use this factor to differentiate between users, because many controls would be manipulated at the same time, restricting their number of possible owners. Hence, we propose in this paper an algorithm to effectively segment (or partition) the controls on a surface according to their most likely owner based on collisions arising as a result of simultaneous manipulations of the controls in the shared multi-touch surface. We present several goodness functions that may be applied during

the segmentation process and we perform an experimental analysis to be able to identify the most effective one(s) in this respect.

The rest of the paper is organized as follows. First, related work is described in Section 2, distinguishing between approaches relying on hardware setups versus those focused only on software processing, with no additional hardware instrumentation. In Section 3, we discuss three different approaches for our partitioning algorithm. Section 4 shows an experiment for evaluating these approaches according to several goodness functions. Finally, in Section 5, the conclusions and future works are described.

2 Related work

Several works have been conducted in order to provide the user-aware assignment of controls to owners in a shared multi-touch surface. Among the existing works we distinguish two types of approaches: hardware-based and software-based. Hardware solutions rely on either enhancing the tabletops with external devices or equipping the user with some wearables, whereas most software approaches explore a single factor for the user differentiation task, usually the position and orientation of finger contacts. According to the desirable qualities proposed by (Harrison, Sato, & Poupyrev, 2012) (Zhang, Yang, Ens, Liang, Boulanger, & Irani, 2012) that user differentiation systems should have, software-based approaches seem to be more desirable, except for they are less robust than the hardware-based ones.

Regarding hardware solutions, the DiamondTouch tabletop (Dietz & Leigh, 2001) uses antennas beneath the surface that form a circuit with some receivers situated under the seats of the users. When a finger touches the surface, the circuit closes producing a unique electrical path, which allows the system to know which user has produced the contact. Harrison et al. (Harrison, Sato, & Poupyrev, 2012) present a promising method for user differentiation based on the electrical properties of users' bodies, using an external sensor. Although it is designed for tablets, the principles they use could be extended to tabletops in the near future.

Dohse et al. (Dohse, Dohse, Still, & Parkhurst, 2008) propose the use of an external camera situated above the surface to infer the position of the users capturing their hands and arms. Schmidt et al. present HandsDown (Schmidt, Chong, & Gellersen, 2010), a way of user identification based on the contour of the hands and the posterior tracking of the user using also an external camera. Proposals described in (Annett, Grossman, Wigdor, & Fitzmaurice, 2011) (Tănase, Vatabu, Pentiuc, & Graur, 2008) use proximity sensors to detect users and track them even if they move around the table. However, if they move away, the system forgets about them and they are considered new users when they return. Other works use gloves with coded tags (Marquardt, Kiemer, Ledo, Boring, & Greenberg, 2011), wristbands with LED transmitting an identification code (Meyer & Schmidt, 2010) or rings transmitting infrared light pulses (Roth, Schmidt, & Güldenring, 2010), which are recognized by the vision systems of some tabletops (e.g. Microsoft PixelSense).

All the previous proposals need of additional hardware besides the working tabletop, sometimes embedded in it and sometimes external. Some drawbacks are identified by their respective authors, such as that the users are tied to a fixed location (Dietz & Leigh, 2001) (Dohse, Dohse, Still, & Parkhurst, 2008), the recognition is not very accurate when several

users are working on the same side of the table (Dohse, Dohse, Still, & Parkhurst, 2008), and the number of users is limited (Dietz & Leigh, 2001) (Harrison, Sato, & Poupyrev, 2012) (Marquardt, Kiemer, Ledo, Boring, & Greenberg, 2011) (Meyer & Schmidt, 2010) (Roth, Schmidt, & Güldenring, 2010). Also, the use of wearable hardware (Marquardt, Kiemer, Ledo, Boring, & Greenberg, 2011) (Meyer & Schmidt, 2010) (Roth, Schmidt, & Güldenring, 2010) may result uncomfortable for some people, and it usually requires a previous registration from the users (also observed in (Schmidt, Chong, & Gellersen, 2010)), which is not a very natural way of starting the interactions.

With respect to software solutions, Wang et al. (Wang, Cao, Ren, & Irani, 2009) present an approach to the differentiation of users by using the orientation of fingers to infer their position. However, it presents some problems when several users are working on the same side of the surface and, if they move, the system cannot track them. In this line, Dang et al. (Dang, Straub, & André, 2009) also present a way of finding the hand which some contacts belong to, given the position and orientation of the fingers. Zhang et al. (Zhang, Yang, Ens, Liang, Boulanger, & Irani, 2012) also present a differentiation of users relying on the orientation of fingers and a machine learning approach to predict the position of users. Although their solution provides high accuracy, they have only studied interactions done with the index finger and with at most three users standing at very specific places. In this work, the authors remark the potentiality of combining their distinction method with other approaches. When working with controls where the contents are represented in a fixed direction, such as text elements, the controls are faced towards the owner (Kruger, Carpendale, Scott, & Greenberg, 2003) (Morris, Lombardo, & Wigdor, 2010). Morris et al. (Morris, Lombardo, & Wigdor, 2010) rely on a technique based on the pointing direction of controls in order to detect users. Nevertheless, their solution works only with this type of controls. This would not be suitable when working with 360º controls, which try to mitigate the problem of accessing the elements from any position around the table (e.g. (Catala, Garcia-Sanjuan, Jaen, & Mocholi, 2012)). Besides, their proposal considers only four directions (North, South, East, West) and, therefore, the number of users that can be interacting with the surface is limited to four.

3 Tabletop User Segmentation Based on Simultaneous Interactions

In order to allow the system to differentiate between users when several ones are interacting simultaneously with the controls of an application, we have designed an algorithm that exploits the “simultaneous manipulation of controls” factor described in Section 1. The rationale behind the proposed algorithm is the classification of the existing controls in several groups in order to avoid two controls to be in the same one if they have been manipulated at the same time previously. We henceforward refer to this time overlapping as “usage collision”. Thus, a given group is a collision-free container of controls. The ideal situation will be having all the controls belonging to a given user in the same segment, and as many as users interacting on the tabletop. The case of specific controls that require the simultaneous manipulation of contacts is not a threat for our approach since these manipulations are known in advance, i.e.

the gestures involving more than one contact are known at design time and may be filtered out not to consider them as usage collisions.

Let us define a function H (*Hit*) that indicates whether two controls have collided with each other in the past:

$$H: (Control \subset (\mathbb{N} \cup \{0\})) \times (Control \subset (\mathbb{N} \cup \{0\})) \longrightarrow \{True, False\}$$
$$H(a,b) = \begin{cases} True & \text{, if } a \text{ and } b \text{ have collided} \\ False & \text{, otherwise} \end{cases}$$

At the beginning of the execution, all the controls are inside a single group, and any new control entering the system is automatically classified in it. As soon as two elements collide, the segmentation algorithm is triggered in order to obtain a new collision-free separation (a new collection of parts) of the existing controls. This is accomplished by extracting one of the two colliding controls from its group and moving it to another where there are no controls that have been collided previously with it (*migration*). If there is no such a collision-free container then a new one is created (*split*). Moreover, if, after migration takes place, two groups contain controls that have not collided with one another, the algorithm will merge the two segments into one (*merging*). This will avoid the number of groups to increase disproportionately. The main steps of the algorithm, executed after a new collision is detected, are shown in Algorithm 1. In sum, this algorithm is designed to support two properties or states:

- If two groups exist is because there exists at least one control in one of them that has collided with at least another control of the other container. In mathematical terms:
$$\forall k_i k_j \; Group \mid k_i \neq k_j \big(\exists c_j \in k_j \exists c_i \in k_i \mid H(c_i, c_j) = True\big),$$
where $Group \subset (\mathbb{N} \cup \{0\})$
- None of the controls within a group have collided with one another, which can be mathematically expressed as:
$$\forall k \; Group \; \forall c_i c_j \in k \; \big(H(c_i, c_j) = False\big)$$

Input: c_i, c_j colliding controls
Precondition: c_i and c_j have not collided in the past and they are in the same group when the collision takes place. Otherwise, the states before and after the execution of the algorithm will be the same.
Begin
Step 1: Select control to be moved m: $select(c_i, c_j)$
Step 2 (split): If m does not have any destination collision-free group, create a new one and establish it as the destination container for m. Then go to step 4.
Step 3 (migration): Select the destination group within the ones m can be moved to: $destination(m)$
Step 4 (merging): Try to merge any groups containing only controls that have not collided with one another.
End

Algorithm 1. Algorithm used to move controls between groups when a collision occurs.

In the optimal situation, if every control of each user collides with all the controls of the others, the algorithm arrives to the desired situation where there exists one group per user and every control within a group belong to the same person, although this situation may be reached before all the collisions take place. In most cases the segmentation algorithm will

reach suboptimal solutions which are still better than following a completely uninformed approach in terms of territoriality management.

To deal with steps 1 and 3, different versions of the algorithm have been created. A naive one or baseline (defined as a basis for comparison), where the selected control and the destination group are chosen arbitrarily, and several alternative heuristic-based ones to make the classification more accurate since the movements of controls are made to containers where they are more likely to pertain. This is important because in a real application, where people are manipulating controls, it would be difficult that all the collisions necessary to achieve the optimal classification take place.

3.1 Heuristic approaches

Users tend to maintain their controls grouped (Scott, Sheelagh, Carpendale, & Inkpen, 2004). Under this hypothesis, we propose several more informed versions of the algorithm which are based on the distances between controls. A group can be seen as a set of points representing the positions of its contained controls. Therefore, the centroid of a given group k with n controls inside (which positions are labeled from p_1 to p_n) is calculated as follows:

$$centroid: Group \longrightarrow \mathbb{Z}^2$$
$$centroid(k) = \frac{p_1 + p_2 + \cdots + p_n}{n}$$

Under the previous assumption, the distance from each control to the centroid of its group (the one containing the controls owned by the same user) should be smaller than the distance to the centroids of the rest.

Let us redefine the argmin and argmax functions so they return a single element instead of a set:

$$\underset{\substack{x_i \in X \\ i \in \mathbb{N}}}{\operatorname{argmin}'} f(X) = \begin{cases} x_a & , if\ \{x_a\} = \underset{X}{\operatorname{argmin}} f(X) \\ x_a & , if\ \{x_a, x_b, \cdots\} = \underset{X}{\operatorname{argmin}} f(X) \end{cases}$$

And argmax' is defined analogously.

Let us define now some functions that will be needed for the formal specification of the different heuristic approaches. These functions are $availables$, which returns the number of available (collision-free) containers for a given control; and $isAvailable$, which indicates whether a control can be moved to a specific group (if it is collision-free for the given control). Also, let k_c be the current group of a surface control c.

$$isAvailable: Control \times Group \longrightarrow \{True, False\}$$
$$isAvailable(c, k) = \begin{cases} True & , \text{if } k \neq k_c \wedge \forall c' Control \in k \left(\neg H(c, c')\right) \\ False & , \text{otherwise} \end{cases}$$

$$availables: Control \longrightarrow \wp(Group)$$
$$availables(c) = \{k \mid isAvailable(c, k) = True\}$$

Given the $isAvailable$ function, the step 3 of our algorithm is defined in a straightforward way by the function $destination$. Note that c_{pos} represents the position of a surface control c.

$$destination(c): Control \longrightarrow Group$$
$$destination(c) = \underset{k \in availables(c)}{\text{argmin}'} \left\| c_{pos} - centroid(k) \right\|$$

Subsections 3.1.1 - 3.1.3 explain, for each considered approach, how the step 1 of the algorithm is defined.

3.1.1 Closest Remote Centroid Migration Approach (CRCM)

When a collision occurs between two controls, this approach chooses the one with the closest available group (i.e. closest to the centroid) as the candidate control to be migrated. In case of tie, the selected control will be that with a greater number of available destinations. This is to prevent the excessive creation of new segments, although the case where two controls are at the exact same distance from a centroid is highly unlikely.

Let d_c^{Avail} be the distance of a control c to the centroid of its closest available group:

$$d_c^{Avail} = \begin{cases} \min\limits_{k \mid isAvailable(c,k)=True} \left\| c_{pos} - centroid(k) \right\| & , \text{if } \exists k\ isAvailable(c,k) = True \\ undefined & , \text{otherwise} \end{cases}$$

Then, the selection function for two colliding controls c_i, c_j could be defined as follows:

$$select: Control \times Control \longrightarrow Control$$
$$select(c_i, c_j) = \begin{cases} \underset{c \in \{c_i, c_j\}}{\text{argmin}'}\ d_c^{Avail} & , \text{if } d_{c_i}^{Avail} \neq d_{c_j}^{Avail} \wedge \\ & \exists k_1, k_2 Group(isAvailable(c_i, k_1) \wedge isAvailable(c_j, k_2)) \\ \underset{c \in \{c_i, c_j\}}{\text{argmax}'} |availables(c)| & , \text{otherwise} \end{cases}$$

3.1.2 Furthest Local Centroid Migration Approach (FLCM)

This is a different approach where, given the two colliding controls, the one which is furthest away from the centroid of its current group is chosen to be moved. In case of tie, the behavior is the same as for CRCM. The rationale behind this heuristic is that the further the control is to the centroid of its current segment the more likely it is it belongs to a different one.

Let $d_c^{Current}$ be the distance between the control c and the centroid of its current group:

$$d_c^{Current} = \left\| c_{pos} - centroid(k) \right\|, k\ Group \wedge c \in k$$

The selection function for two colliding controls c_i, c_j in this case would be defined as follows:

$$select(c_i, c_j) = \begin{cases} \underset{c \in \{c_i, c_j\}}{\text{argmax}'}\ d_c^{Current} & , \text{if } d_{c_i}^{Current} \neq d_{c_j}^{Current} \\ \underset{c \in \{c_i, c_j\}}{\text{argmax}'} |availables(c)| & , \text{otherwise} \end{cases}$$

In the unlikely case in which both controls are at the same distance to the centroid of their current group, the one with more available destinations is selected.

3.1.3 Remote-Local Centroids Difference Approach (RLCD)

In both CRCM and FLCM approaches, it is possible to come to a situation where the state reached after moving a control is worse than the previous one. For example, in CRCM, if the selected control to be moved c_i has a distance to its destination centroid which is greater than the distance to its current group, i.e. in mathematical terms if $\text{argmin}'_{c\in\{c_i,c_j\}} d_c^{Avail} = c_i$, but $\mathrm{d}_{\mathrm{c_i}}^{Avail} > \left\| c_{i_{pos}} - centroid(k_{c_i}) \right\|$ where $k_{c_i} = \{k\ Group \mid c_i \in k\}$, then the best option for c_i might be stay in its current group, which it is the closest for the control.

Let us specify an example of the situation described above with some sample numerical values as shown in Figure 1. In the picture, c_i and c_j are represented as circles; the centroid of their current container is symbolized as a square; and the centroids of their respective closest available groups, as triangles. In this example, if only CRCM was taken into account, the selected control to be moved would be c_i, as it presents a lower value of distance to the centroid of the remote closest segment. From the perspective of FLCM, this decision would be erroneous, since c_i is closer to the centroid of its current group, and c_j would be the candidate control that would have to be moved instead. The same situation occurs if FLCM is applied because the selected control (c_j) leads to a non-optimal state of the system in terms of the CRCM approach.

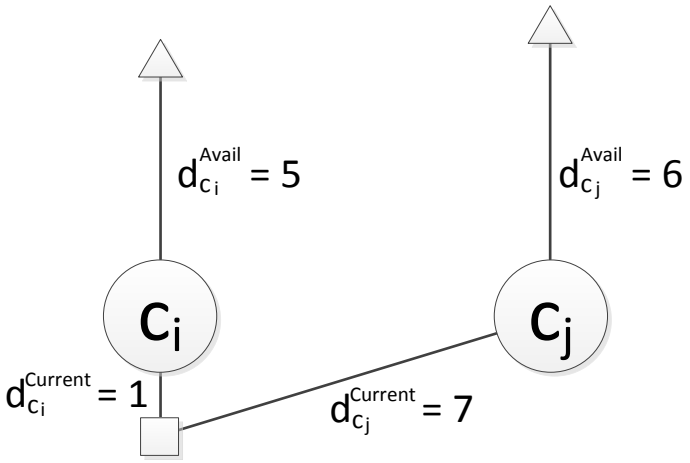

Figure 1. Example of RLCD measured distances for two colliding controls $\boldsymbol{c_i}$, $\boldsymbol{c_j}$. The centroids of their closest available destination containers are represented by triangles, and the centroid of their own current group is represented by a square.

Therefore, this example suggests that a clever combination of CRCM and FLCM would result in a better heuristic in terms of effectively migrating colliding controls.

Given two colliding controls c_i, c_j , let us define two error measures $\Delta^{Avail} = \left| d_{c_i}^{Avail} - d_{c_j}^{Avail} \right|$ and $\Delta^{Current} = \left| d_{c_i}^{Current} - d_{c_j}^{Current} \right|$ where Δ^{Avail} represents the overall distance difference with respect to the respective closest remote available destination and $\Delta^{Current}$ represents the overall distance difference with respect to the centroid of the current group of c_i, c_j. If Δ^{Avail} is greater than $\Delta^{Current}$ then we will migrate the control which is closest to its remote available destination ($\text{argmin}'_{c\in\{c_i,c_j\}} d_c^{Avail}$) because this decision will contribute to reduce the difference which is greater in magnitude (Δ^{Avail}). Otherwise, the control to be migrated

will be the one which is furthest away from its current segment ($\text{argmax}_{c\in\{c_i,c_j\}}{}' d_c^{Current}$) because this decision will contribute to reduce the greater difference ($\Delta^{Current}$).

RLCD is defined in mathematical terms as follows:

$$select(c_i, c_j) = \begin{cases} \underset{c\in\{c_i,c_j\}}{\text{argmin}}{}' d_c^{Avail} \\ \underset{c\in\{c_i,c_j\}}{\text{argmax}}{}' d_c^{Current} \end{cases}, \quad \begin{array}{c} \text{if } availables(c_i) \neq \emptyset \wedge availables(c_j) \neq \emptyset \wedge \\ (\Delta^{Avail} > \Delta^{Current}) \\ \text{, otherwise} \end{array}$$

4 Experimental Analysis

The performance analysis of the proposed approaches can be done in terms of their theoretical asymptotic complexity and in terms of its experimental computational effectiveness, i.e. not only according to the time it takes to perform the computation after a collision occurs but also analyzing the suitability of the state reached after a collision has been processed using each of the proposed approaches. Taking into account the asymptotic time complexity, the choice of the approach is irrelevant. All of them have an asymptotic cost $O(K^4)$ considering K the number of current segments or groups. As K tends to the number of controls n (in the situation where each control is in a different segment), the complexity of the worst case in which there would be just one control per group would be $O(n^4)$. Although this is a high theoretical cost, empirically the time is not a critical factor due to the physical restrictions of the applications of this algorithm, where the number of users and the number of controls per user involved are limited.

However, the most important issue is to find the approach that provides a better distribution of the controls among groups. This comparative study has been carried out by implementing a simulator of collisions so that a great number of experimental repetitions may be generated for each configuration of the experimental factors without the intervention of real users.

4.1 Apparatus

The implemented simulator allows the specification of the simulation factors: the dimensions of the working surface, the number of users involved and their respective number of controls. In our experiments, the value for the “dimensions of the surface” factor has been fixed to 1024 x 768 pixels, to consider a surface of real dimensions such as the Microsoft Pixelsense 1.0. The simulator places a certain number of controls on the surface randomly, depending on two parameters: the number of users U that could be interacting at the same time with the application, and the number of controls C that each user owns. The distribution of the controls is done under the territoriality hypothesis explained in Section 3.1, i.e. grouped in front of their owner. The simulator divides the surface as if the users were situated around it (simulating a real situation such as the one shown in Figure 2), assigning the same size of working space to each user, and places the appropriate number of controls randomly in each section according to C, as depicted in Figure 3 with an example for $U = 4$ and $C = 3$. On this configuration all the possible collisions between the controls from different users are randomly generated. All these sequences of simulated interactions will be part of the datasets used as input in the study.

Figure 2. Several users interacting with a tabletop.

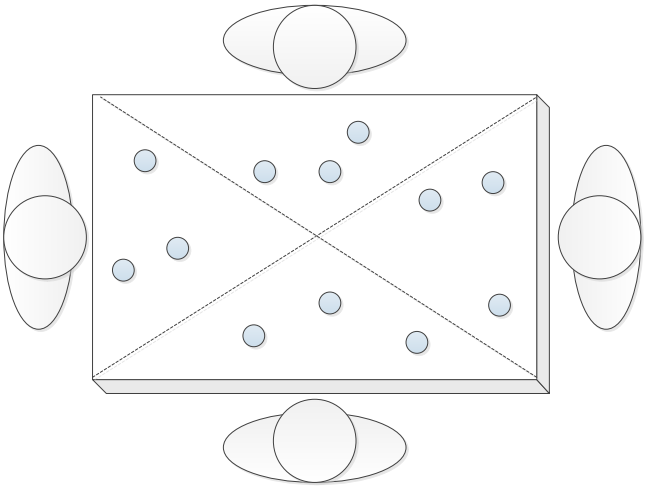

Figure 3. Distribution of the controls by the simulator in a hypothetical situation with $\boldsymbol{U = 4}$ and $\boldsymbol{C = 3}$.

4.2 Goodness Functions

To study the goodness of the different approaches, several functions have been implemented. For each collision observed, these functions represent how far the state reached after the collision is from the ideal situation in which all the controls have been successfully classified. In terms of how these functions have been defined, the closer their value is to zero, the better the approach, because it drives the state of the groups containing controls towards the ideal classification in which all of them contain only the controls owned by a single user.

Let u_k be the number of different users who have controls inside the group k; K, the number of current existing groups; n_k, the number of controls within k; U, the number of participants; and C, the number of controls that each user owns. The measured functions are defined as follows:

- Function F1 (*Excess of users*): Ideally, there should be one segment for each user. This function measures the difference between the current number of users in a group with respect to the desired situation.

$$F_1 = \frac{\frac{\sum_{k\ Group}(u_k - 1)^2}{U}}{K}$$

- Function F2 (*Excess of controls*): Ideally, there should be C controls per user in each group. This function computes the difference between the current number of controls within a container with respect to the desired situation.

$$F_2 = \frac{\sum_{k\ Group}(n_k - C)^2}{K}$$

- Function F3 (*Spread of controls*): The controls inside a group should belong to the same user and, therefore, they should be physically close to one another. This function measures the distance between the controls and the centroid of their respective container. The more grouped they are, the closer to 0 the value will be.

$$F_3 = \frac{\sum_{k\ Group}\sum_{c\ Control\ in\ k}\frac{\left\|centroid(k) - c_{pos}\right\|^2}{n_k}}{K}$$

- Function F4 (*Lack or excess of groups*): On the final (ideal) situation, the number of containers K should be equal to the number of users U, since each one of them should contain the controls belonging to a single user. Hence, this function measures the lack (negative values) or excess (positive values) of groups with respect to this ideal situation.

$$F_4 = K - U$$

In addition to the previous functions, several refinements of function F1 have been considered given that the heterogeneity of each group in terms of the users included is a key quality factor. In this sense we want to pay special attention to the degree or "density" of this heterogeneity in terms of the proportion of controls of a given user in a group with respect to the proportion of controls in that same container belonging to other users. This heterogeneity density can be measured in different ways given the proportion of controls of a given user in a group. Let us define:

n_k^u as the number of controls in group k owned by user u.

n_k as the total number of controls in group k.

$\rho_k^u = \frac{n_k^u}{n_k}$ as the density of controls in group k owned by user u.

- Function F5 (*Proportion of controls: max*): It takes the maximum value of the densities in each container and obtains an average value for all the segments. In other words, it measures the goodness of a given approach as how effective it is in terms of increasing the maximum average density of each group.

$$F_5 = 1 - \frac{\sum_{k\ Group}\max_{u\ user\ with\ controls\ in\ k}\rho_k^u}{K}$$

- Function F6 (*Proportion of controls: max-min*): It takes the difference between the maximum and the minimum values of the density in each group and obtains an average value. It measures how effective is a given approach in increasing the difference between the maximum and minimum values of density within each group.

$F_6 = 1 - \frac{\sum_{k\ Group} x}{K}$, where

$$x = \begin{cases} \left(\max_{u\ user\ with\ controls\ in\ k} \rho_k^u\right) - \left(\min_{u\ user\ with\ controls\ in\ k} \rho_k^u\right) & \text{, if } u_k > 1 \\ \rho_k^u & \text{, otherwise} \end{cases}$$

- Function F7 (*Proportion of controls: max1-max2*): It takes the difference between the two maximum values of density in each container and then obtains the average value of those calculated differences. This function measures the effectiveness of a given heuristic in reducing the number of maximum densities within a group. Groups with high values of densities for different users involve a potentially high number of control migrations in the future to reach the ideal situation in which all the controls in a given segment belong to the same user.

$$F_7 = 1 - \frac{\sum_{k\ Group} x}{K}, \text{where}$$
$$x = \begin{cases} \rho_k^u & \text{, if } u_k = 1 \\ max1 - max2 & \text{, otherwise} \end{cases}$$
$$max1 = \max_{u\ user\ with\ controls\ in\ k} \rho_k^u$$
$$max2 = \max_{u\ user\ with\ controls\ in\ k\ except\ umax} \rho_k^u$$
$$umax = \operatorname{argmax}'_{u\ user\ with\ controls\ in\ k} \rho_k^u$$

- Function F8 (*Proportion of controls: max-rest*): This is a more elaborated version of functions F6 and F7 in which not just the differences between max-min densities or max1-max2 densities are calculated within each group, but rather the difference between the highest and the average of the rest density values.

$$F_8 = 1 - \frac{\sum_{k\ Group} x}{K}, \text{where}$$
$$x = \begin{cases} \rho_k^u & \text{, if } u_k = 1 \\ max - rest & \text{, otherwise} \end{cases}$$
$$max = \max_{u\ user\ with\ controls\ in\ k} \rho_k^u$$
$$rest = \frac{\sum_{u\ user\ with\ controls\ in\ k\ except\ umax} \rho_k^u}{u_k - 1}$$
$$umax = \operatorname{argmax}'_{u\ user\ with\ controls\ in\ k} \rho_k^u$$

4.3 Procedure

A set of experiments have been conducted to know whether there are any significant differences among the presented heuristic approaches when the number of users U and the number of controls C for each user vary. The comparative analysis of the proposed heuristics has been performed in terms of the goodness functions described above.

In the experimental design, the number of users U has been varied from 2 to 9 and the number of controls per user C, from 1 to 10. In order to simplify the analysis, these values have been arranged in groups of density (low, medium and high) as it is depicted in Table 1. Among all possible combinations of values for C and U, a subset of 3 has been randomly selected as representative candidates for the different densities. With these combinations, the corresponding datasets have been generated according to the conditions described in section 4.1 changing both the position of the controls and the order in which the collisions take place, resulting in 50 trials per combination and a total of 450 different trials.

Hence, a trial refers to a specific combination of U and C along with a specific sequence order of collision events among the controls. Every collision event in each trial's sequence will be processed by all the approaches. After a collision is observed and processed using a given approach, each goodness function is computed and its value stored for a posterior analysis, so that a complete trace of the evolution of this function is obtained.

U	Low	2, 3
	Medium	4, 5, 6
	High	7, 8, 9
C	Low	1, 2, 3
	Medium	4, 5, 6 , 7
	High	8, 9, 10

Table 1. Classification of $\boldsymbol{U}$ and $\boldsymbol{C}$ by densities.

4.4 Results and discussion

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	3201	U7C3	3 - 2 - 10
		U3C1	0123	U5C3	32 - 10	U8C2	3210
		U3C3	3 - 201	U6C2	32 - 1 - 0	U9C1	23 - 01
	Medium	U2C5	312 - 0	U4C6	3 - 2 - 1 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	312 - 0	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	3 - 21 - 0	U6C7	3 - 2 - 1 - 0	U9C5	3 - 2 - 1 - 0
	High	U2C8	3 - 120	U4C10	3 - 2 - 1 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	312 - 0	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 21 - 0	U6C9	3 - 2 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 2. Classification of the approaches from the best to the worst for the goodness function F1.

		U					
		Low		Medium		High	
C	Low	U2C2	1302	U4C1	3201	U7C3	2301
		U3C1	0123	U5C3	3201	U8C2	2310
		U3C3	0132	U6C2	2 - 310	U9C1	23 - 01
	Medium	U2C5	1320	U4C6	3210	U7C4	23 - 01
		U2C6	13 - 20	U5C5	32 - 10	U8C6	23 - 0 - 1
		U3C7	3120	U6C7	32 - 01	U9C5	23 - 01
	High	U2C8	13 - 02	U4C10	321 - 0	U7C10	32 - 1 - 0
		U2C9	13 - 20	U5C8	32 - 1 - 0	U8C8	32 - 01
		U3C10	321 - 0	U6C9	32 - 10	U9C9	32 - 01

Table 3. Classification of the approaches from the best to the worst for the goodness function F2.

		U					
		Low		Medium		High	
C	Low	U2C2	3210	U4C1	3201	U7C3	3 - 2 - 1 - 0
		U3C1	3210	U5C3	32 - 1 - 0	U8C2	3 - 21 - 0
		U3C3	3210	U6C2	32 - 1 - 0	U9C1	3210
	Medium	U2C5	3210	U4C6	3 - 21 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	3120	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	321 - 0	U6C7	3 - 2 - 1 - 0	U9C5	3 - 2 - 1 - 0
	High	U2C8	3120	U4C10	3 - 21 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	3120	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 21 - 0	U6C9	3 - 2 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 4. Classification of the approaches from the best to the worst for the goodness function F3.

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	1023	U7C3	1302
		U3C1	0123	U5C3	1320	U8C2	10 - 32
		U3C3	1032	U6C2	1 - 302	U9C1	01 - 23
	Medium	U2C5	1320	U4C6	3 - 12 - 0	U7C4	3120
		U2C6	3120	U5C5	321 - 0	U8C6	31 - 2 - 0
		U3C7	3 - 12 - 0	U6C7	321 - 0	U9C5	132 - 0
	High	U2C8	31 - 02	U4C10	3 - 21 - 0	U7C10	3 - 21 - 0
		U2C9	31 - 20	U5C8	3 - 21 - 0	U8C8	3 - 21 - 0
		U3C10	321 - 0	U6C9	32 - 1 - 0	U9C9	3 - 21 - 0

Table 5. Classification of the approaches from the best to the worst for the goodness function F4.

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	3021	U7C3	3 - 2 - 1 - 0
		U3C1	0123	U5C3	32 - 1 - 0	U8C2	32 - 1 - 0
		U3C3	3 - 210	U6C2	32 - 1 - 0	U9C1	32 - 10
	Medium	U2C5	312 - 0	U4C6	3 - 2 - 1 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	3120	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	3 - 21 - 0	U6C7	3 - 2 - 1 - 0	U9C5	3 - 2 - 1 - 0
	High	U2C8	31 - 20	U4C10	3 - 2 - 1 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	31 - 20	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 2 - 1 - 0	U6C9	3 - 2 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 6. Classification of the approaches from the best to the worst for the goodness function F5.

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	3021	U7C3	3 - 2 - 1 - 0
		U3C1	0123	U5C3	32 - 1 - 0	U8C2	32 - 1 - 0
		U3C3	3 - 210	U6C2	32 - 1 - 0	U9C1	32 - 10
	Medium	U2C5	312 - 0	U4C6	3 - 2 - 1 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	3120	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	3 - 21 - 0	U6C7	3 - 2 - 1 - 0	U9C5	32 - 1 - 0
	High	U2C8	31 - 20	U4C10	3 - 2 - 1 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	31 - 20	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 2 - 1 - 0	U6C9	32 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 7. Classification of the approaches from the best to the worst for the goodness function F6.

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	3021	U7C3	3 - 2 - 1 - 0
		U3C1	0123	U5C3	32 - 1 - 0	U8C2	32 - 1 - 0
		U3C3	3 - 210	U6C2	32 - 1 - 0	U9C1	32 - 10
	Medium	U2C5	312 - 0	U4C6	3 - 2 - 1 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	3120	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	3 - 21 - 0	U6C7	3 - 2 - 1 - 0	U9C5	32 - 1 - 0
	High	U2C8	31 - 20	U4C10	3 - 2 - 1 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	31 - 20	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 2 - 1 - 0	U6C9	32 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 8. Classification of the approaches from the best to the worst for the goodness function F7.

		U					
		Low		Medium		High	
C	Low	U2C2	3120	U4C1	3021	U7C3	3 - 2 - 1 - 0
		U3C1	0123	U5C3	32 - 1 - 0	U8C2	32 - 1 - 0
		U3C3	3 - 210	U6C2	32 - 1 - 0	U9C1	32 - 10
	Medium	U2C5	312 - 0	U4C6	3 - 2 - 1 - 0	U7C4	3 - 2 - 1 - 0
		U2C6	3120	U5C5	3 - 2 - 1 - 0	U8C6	3 - 2 - 1 - 0
		U3C7	3 - 21 - 0	U6C7	3 - 2 - 1 - 0	U9C5	32 - 1 - 0
	High	U2C8	31 - 20	U4C10	3 - 2 - 1 - 0	U7C10	3 - 2 - 1 - 0
		U2C9	31 - 20	U5C8	3 - 2 - 1 - 0	U8C8	3 - 2 - 1 - 0
		U3C10	3 - 1 - 2 - 0	U6C9	32 - 1 - 0	U9C9	3 - 2 - 1 - 0

Table 9. Classification of the approaches from the best to the worst for the goodness function F8.

Each experiment has been named UxCy, where *x* is the number of users involved and *y*, the number of controls per user. Table 2 - Table 9 show a sorting of approaches (from the best to the worst) based on each goodness function. The approaches are represented by a number instead of their names to increase the readability of the tables. The approach 0 represents the naive one (the baseline), approaches 1, 2 and 3 represent CRCM, FLCM and RLCD respectively.

Within the tables, the different approaches form groups if there are no significant differences between them. Inside these groups, they are also sorted according to their mean value (from the lowest to the greatest). For instance, in Table 2 the entry for experiment U6C2 is "32-1-0" which means that there are three statistically significant different groups of approaches being 3 (RLCD) and 2 (FLCM) the best ones followed by approach 1 (CRCM) and approach 0 (baseline) which is the worst one in terms of the goodness function F1. However, despite there are no significant differences between 3 and 2 the average F1 goodness value for approach 3 is lower than the one for approach 2 and this is why it is listed first within the group "32".

In order to determine the groups of approaches, the differences in terms of performance were statistically tested. When data normality could be assumed based on Shapiro-Wilk tests, an ANOVA was performed and post-hoc pairwise comparisons using Bonferroni's method were applied to check which approaches are significantly different from one another. Nevertheless, when the normality assumption cannot be accepted, a Kruskal-Wallis test is done with its respective post-hoc pairwise comparisons. In one way or another, the idea is to be able to decide whether the performance of each approach for a given goodness function significantly differs from each other, and eventually decide about the groups.

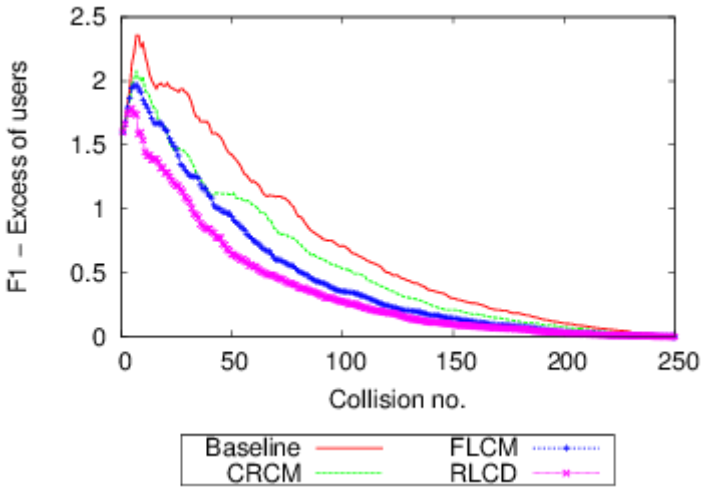

Figure 4. Evolution of F1 with the collisions produced for the experiment U5C5.

The analysis of Table 2 (F1, excess of users) reveals that there is no clear separation between approaches for low densities of either U or C, even though sometimes the naive approach appears to be separated from the other ones. This indicates that, when the number of users or the number of controls per user is small, the use of an informed approach tends to separate quicker the controls of different users than the naive approach. As the densities of U and C become higher, the difference between approaches is clearer. The naive approach presents the worst results, followed by the approach 1 (CRCM) which only takes into account the distance to the available destinations in order to choose the control to be moved from a group in case of collision. Approach 2 (FLCM), which chooses always the control which is furthest to its own segment, seems to be slightly better, but the combination of approaches CRCM and FLCM into approach RLCD is the one which outperforms the others, keeping, mostly, controls from the same user grouped. Figure 4 shows the measured values of F1 after each collision

produced for an experiment with medium densities for both U and C. As it can be seen in this plot, the use of heuristics clearly reduces the excess of users within groups.

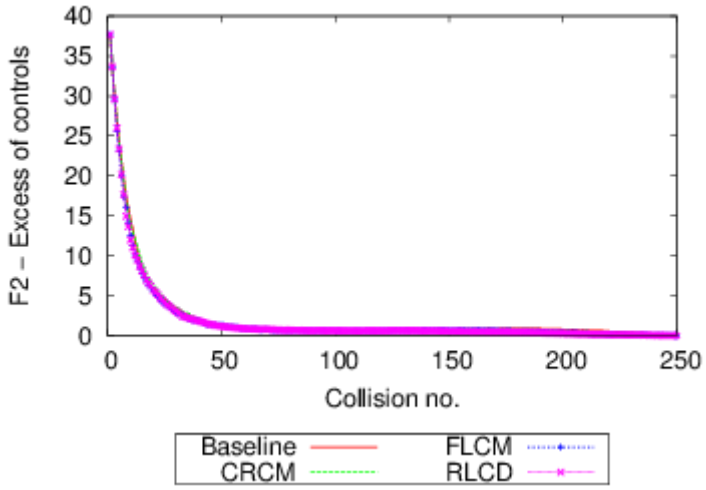

Figure 5. Evolution of F2 with the collisions produced for the experiment U5C5.

Regarding F2, the excess of controls (see Table 3), there are no significant differences between approaches 2 (FLCM) and 3 (RLCD), which are better than the other two for combinations of medium and high densities of U and C. This means that, when trying to have C controls in each group (which would be the ideal situation), the right decision to choose is moving the control that is furthest to the centroid of its own container. For low densities of controls per user, there are no clear differences between approaches. Additionally, some combinations of medium size of users and controls reveal a varying set of winning approaches. These varying results make us conclude that F2 is not an effective discrimination factor to measure the goodness of the approaches as it is shown in Figure 4.

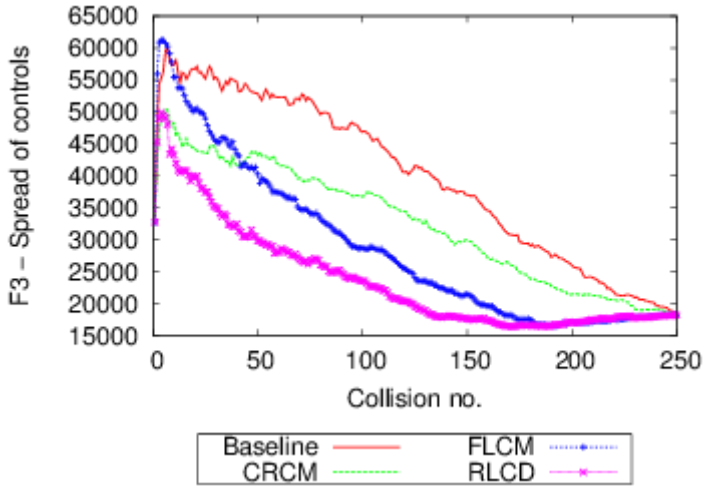

Figure 6. Evolution of F3 with the collisions produced for the experiment U5C5.

Function F3 (see Table 4) measures the average distance between the controls classified in the same group and their centroid. Since this function takes into account the distances between controls, it is expected that the heuristic versions outperform the baseline (which chooses the

control to be moved randomly). Indeed, the results reveal that for any combinations of medium and high densities there is a clear separation of approaches. RLCD approach (number 3) is the winner heuristic followed by approaches 2 (FLCM) and 1 (CRCM) (with FLCM outperforming CRCM), keeping together the controls that are closest to each other (see Figure 6). The separation is not that clear for low densities of C and medium-high densities of U, but the tendency is the same. However, for a low density of number of users there are no significant differences between the different approaches, although RLCD shows smaller mean values of F3.

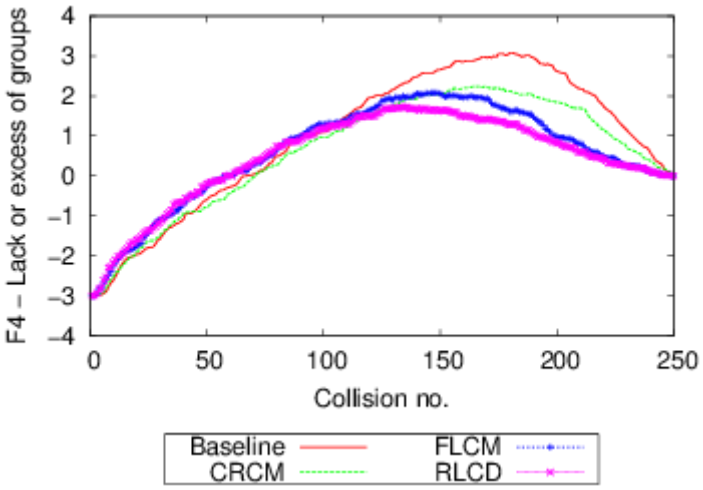

Figure 7. Evolution of F4 with the collisions produced for the experiment U5C5.

The differences between approaches are not very strong considering F4 (the lack or excess of groups), as it can be seen in Figure 7 as well as in Table 5. Although the RLCD approach seems to outperform the baseline for combinations of medium and high densities of U and C, high densities of C are needed in order to differentiate between the heuristic approaches. In this case, the approach that creates a number of segments closest to the ideal case is the approach 3 (RLCD), followed by the other two heuristic approaches and, finally, the naive one. Besides being a metric to measure how close is an approach to the ideal case, this function is important to measure the goodness of an heuristic because a great excess of the number of groups could be detrimental to the asymptotic computational cost, that is a function of the number of this number K: $O(K^4)$. Figure 7 shows the evolution of this function with respect to the number of collisions processed. At first, there are fewer groups than desired, and after around the sixtieth collision, there are an excessive number of them. However, this excess is quite small and, therefore, this is not a critical issue for the computational cost.

Functions F5 (Table 6), F6 (Table 7), F7 (Table 8) and F8 (Table 9) measure the heterogeneity inside a group in terms of the proportion of controls belonging to each user who has controls in that group. Even though these functions differ from one another, essentially they measure a similar factor, i.e. density heterogeneity, which should in theory result in a similar capability for discriminating among approaches. This is in fact confirmed by the results obtained and, as it happens with the other goodness functions explained before, the differences between approaches become very clear when dealing with combinations of medium and high densities of both U and C. In this case, as it can be seen in Figure 8, the RLCD approach outperforms the

others, followed by approaches 2 (FLCM), 1 (CRCM) and 0 (the baseline). Besides, when either low densities of U or C are involved, approaches CRCM and FLCM degrade and become as effective as the naive one, but still in this situation RLCD emerges as the best one in most cases.

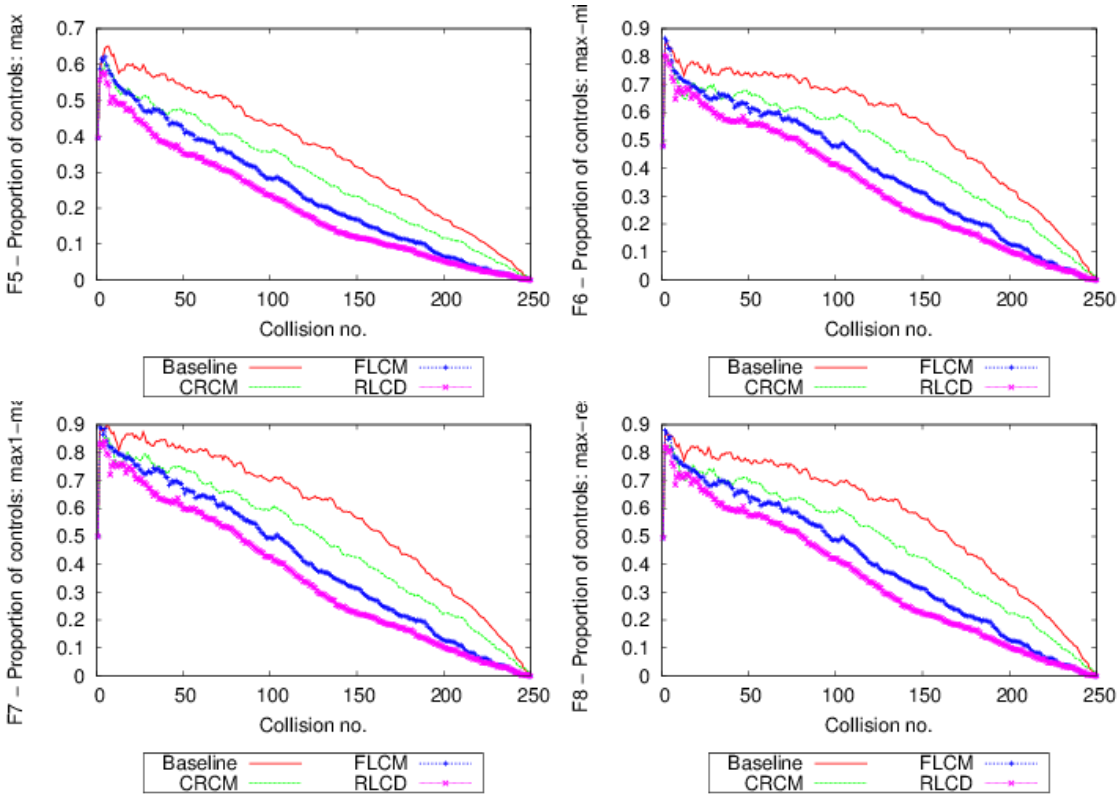

Figure 8. Evolution of F5, F6, F7 and F8 with the collisions produced for the experiment U5C5.

In conclusion, taking into account the different goodness functions measured, the use of heuristics presents better results than choosing arbitrarily the control to be moved. This behavior is clearer as the number of users U and the number of controls per user C increases. Nevertheless, it is important to note that, even with small sizes of U and C, the distribution of controls made with some heuristic approaches of the algorithm is usually better than the one made with the naive approach. In addition, although high sizes of both U and C could be considered unrealistic because of the limited dimensions of the interactive surfaces, the combination of medium values for U and C still support these conclusions, and therefore this research opens the direction towards future scenarios in which larger surfaces will be involved. The results also show that the RLCD approach, which is the most elaborated one in terms of measuring heterogeneity within each group, is the winner under all circumstances except when low densities of users and controls are present. However, in these situations where the surface is not highly populated, territoriality is not an issue and the application of these algorithms could be avoided.

5 Conclusions and future work

In this paper we have proposed a solution for the problem of user differentiation without using any hardware besides the tabletop, by focusing on a factor that has not yet been explored in

detail: the simultaneous use of different controls. A segmentation algorithm has been implemented which maintains controls in different groups if they are assumed to belong to the same user, achieving in this way the differentiation of the users manipulating those controls. The algorithm follows a generic design so that the basic operations of splitting, migrating and merging that have been considered can be implemented in different ways leading to different approaches. This will be particularly useful to consider and easily accommodate other factors together within the algorithm in the future, by simply expanding or re-implementing these functions with the new elements. In this work, three approaches have been implemented for control migration based on different heuristics. These approaches have been named CRCM (Closest Remote Centroid Migration), FLCM (Furthest Local Centroid Migration) and RLCD (Remote-Local Centroids Difference) and they include the benefits of another factor: the distance between controls. Whereas CRCM selects for migration the closest control to another free group, FLCM chooses the one furthest to its current container. RLCD is a hybrid version of the previous ones.

In a simulated environment, the three versions of the algorithm have been compared between one another and also against a baseline (which arbitrarily selects the control to be moved). The comparison has been performed according to eight goodness functions, and the results have shown that, in general, the best approach is the RLCD one. This implies that not always it is preferable to move the control which is closest to a collision free destination if it is very close to the other controls in its current group, and vice versa.

As a future work, we intend to further test this algorithm with the heuristic of the RLCD approach by performing some experiments with real users in order to test its actual performance in a task that requires user differentiation. Also, we will combine this factor with others already explored by other authors, such as the position and orientation of fingers, and see whether the user differentiation system becomes more robust and accurate when combining several factors.

References

Annett, M., Grossman, T., Wigdor, D., & Fitzmaurice, G. (2011). Medusa: a proximity-aware multi-touch tabletop. *Proceedings of the 24th annual ACM symposium on User interface software and technology* (pp. 337-346). New York, NY, USA: ACM.

Catala, A., Garcia-Sanjuan, F., Jaen, J., & Mocholi, J. A. (2012). TangiWheel: A Widget for Manipulating Collections on Tabletop Displays Supporting Hybrid Input Modality. *Journal of Computer Science and Technology*, 27(4), 811-829.

Dang, C. T., Straub, M., & André, E. (2009). Hand distinction for multi-touch tabletop interaction. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (pp. 101-108). New York, NY, USA: ACM.

Decouchant, D., Mendoza, S., Sánchez, G., & Rodríguez, J. (2013). Adapting groupware systems to changes in the collaborator's context of use. *Expert Systems with Applications*, (In Press).

Dietz, P., & Leigh, D. (2001). DiamondTouch: a multi-user touch technology. *Proceedings of the 14th annual ACM symposium on User interface software and technology* (pp. 219-226). New York, NY, USA: ACM.

Dillenbourg, P., & Evans, M. (2011). Interactive tabletops in education. *International Journal of Computer-Supported Collaborative Learning*, 6, 491-514.

Dohse, K. C., Dohse, T., Still, J. D., & Parkhurst, D. J. (2008). Enhancing Multi-user Interaction with Multi-touch Tabletop Displays Using Hand Tracking. *Proceedings of the First International Conference on Advances in Computer-Human Interaction* (pp. 297-302). Washington, DC, USA: IEEE Computer Society.

Harrison, C., Sato, M., & Poupyrev, I. (2012). Capacitive fingerprinting: exploring user differentiation by sensing electrical properties of the human body. *Proceedings of the 25th annual ACM symposium on User interface software and technology* (pp. 537-544). New York, NY, USA: ACM.

Hornecker, E., Marshall, P., Dalton, N. S., & Rogers, Y. (2008). Collaboration and interference: awareness with mice or touch input. *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (pp. 167-176). New York, NY, USA: ACM.

Kruger, R., Carpendale, S., Scott, S. D., & Greenberg, S. (2003). How people use orientation on tables: comprehension, coordination and communication. *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work* (pp. 369-378). New York, NY, USA: ACM.

Lepinski, G. J., Grossman, T., & Fitzmaurice, G. (2010). The design and evaluation of multitouch marking menus. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2233-2242). New York, NY, USA: ACM.

Marquardt, N., Kiemer, J., Ledo, D., Boring, S., & Greenberg, S. (2011). Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (pp. 21-30). New York, NY, USA: ACM.

Meyer, T., & Schmidt, D. (2010). IdWristbands: IR-based user identification on multi-touch surfaces. *ACM International Conference on Interactive Tabletops and Surfaces* (pp. 277-278). New York, NY, USA: ACM.

Morris, M. R., Lombardo, J., & Wigdor, D. (2010). WeSearch: supporting collaborative search and sensemaking on a tabletop display. *Proceedings of the 2010 ACM conference on Computer supported cooperative work* (pp. 401-410). New York, NY, USA: ACM.

Morris, M. R., Paepcke, A., & Winograd, T. (2006). TeamSearch: Comparing Techniques for Co-Present Collaborative Search of Digital Media. *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems* (pp. 97-104). Washington, DC, USA: IEEE Computer Society.

Morris, M. R., Paepcke, A., Winograd, T., & Stamberger, J. (2006). TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1273-1282). New York, NY, USA: ACM.

Roth, V., Schmidt, P., & Güldenring, B. (2010). The IR ring: authenticating users' touches on a multi-touch display. *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (pp. 259-262). New York, NY, USA: ACM.

Schmidt, D., Chong, M. K., & Gellersen, H. (2010). HandsDown: hand-contour-based user identification for interactive surfaces. *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries* (pp. 432-441). New York, NY, USA: ACM.

Scott, S. D., Sheelagh, M., Carpendale, T., & Inkpen, K. M. (2004). Territoriality in collaborative tabletop workspaces. *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (pp. 294-303). New York, NY, USA: ACM.

Seo, D. W., & Lee, J. Y. (2013a). Physical query interface for tangible augmented tagging and interaction. *Expert Systems with Applications*, 40(6), 2032–2042.

Seo, D. W., & Lee, J. Y. (2013b). Direct hand touchable interactions in augmented reality environments for natural and intuitive user experiences. *Expert Systems with Applications*, 40(9), 3784–3793.

Tănase, C. A., Vatabu, R.-D., Pentiuc, Ş.-G., & Graur, A. (2008). Detecting and Tracking Multiple Users in the Proximity of Interactive Tabletops. *Advances in Electrical and Computer Engineering*, 8, 61-64.

Terrenghi, L., Kirk, D., Sellen, A., & Izadi, S. (2007). Affordances for manipulation of physical versus digital media on interactive surfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1157-1166). New York, NY, USA: ACM.

Tse, E., Histon, J., Scott, S. D., & Greenberg, S. (2004). Avoiding interference: how people use spatial separation and partitioning in SDG workspaces. *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (pp. 252-261). New York, NY, USA: ACM.

Wang, F., Cao, X., Ren, X., & Irani, P. (2009). Detecting and leveraging finger orientation for interaction with direct-touch surfaces. *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (pp. 23-32). New York, NY, USA: ACM.

Zhang, H., Yang, X.-D., Ens, B., Liang, H.-N., Boulanger, P., & Irani, P. (2012). See me, see you: a lightweight method for discriminating user touches on tabletop displays. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2327-2336). New York, NY, USA: ACM.