Master on Intelligent and Adaptable Software Systems 2011/2012

Sketch Arm, custom closets rapid prototyping system

Iván Rodríguez Conde

Department of Computer Languages and Systems, University of Vigo, As Lagoas Campus 32004, SPAIN

Abstract

Custom closets manufacturing arises in response to the spatial limitations that nowadays many of the existing housing present, improving the organization and use of available space in living quarters. This type of closet is practical and highly functional, built from the aesthetic taste and specific storage needs of each individual. However, the end customer usually lacks the skills to address the design process autonomously and is supported by the manufacturer to set its configuration, which represents an unnecessary wait time and increases the final manufacturing cost.

While there are already some pieces of software suitable for closet modeling and design, these tools are generally considered as highly complex technical solutions, primarily aimed at professional users. In order to bring the design process closer to potential customers we introduce Sketch Arm, a brand new tool for custom closets rapid prototyping that allows end users to sketch a completely customized design proposal regardless of their technological training. Supported in current multi-touch hardware technology, the application is conceived as a highly interactive system, and it can also assist the user with the prototype creation tasks by autonomously generating desirable layouts. Incorporating a simulated annealing algorithm enables this automatic generation of layouts by optimizing the closet inner space.

Throughout this paper we present the main features and capabilities of Sketch Arm. We demonstrate that our system is able to synthesize suitable closet layouts, creating several design proposals for different user profiles.

Keywords: procedural modeling; closet design; space layout; stochastic optimization; Computer Aided Design; multi-touch devices.

1. Introduction

Custom closets manufacturing arises in response to the spatial limitations that nowadays many of the existing housing present, improving the organization and use of available space in living quarters. This type of closet is practical and highly functional, built from the aesthetic taste and specific storage needs of each individual.

Although the customer is ultimately responsible for deciding the final configuration of his closet, he usually lacks the expertise to deal with the design process independently and demands manufacturer support. This traditional model represents to the client an unnecessary wait time and an increase in the final manufacturing cost of the closet.

Using a software tool that includes the design expert knowledge, will allow the user to sketch out his ideas without the supervision of others.

In the world of architecture there are many computer applications that facilitate the design process and are usually used to make graphical representations of the real world. These Computer Aided Design (CAD) tools are designed to support the work of engineers, architects and in general all kinds of design professionals. Thus, they are
very powerful but technically demanding and generic application in engineering. That’s why these applications do not conform to the closet design industry, a very traditional and specific sub-sector. However, the penetration of these tools among the lay public is low. This is basically due to two causes: firstly, the creative process is based on expert knowledge that the end user usually doesn’t know, and secondly, these tools generally create a representation that aims to be, above all, faithful to reality, sacrificing the user experience. Insofar as we are able to overcome these shortcomings, we will bring the creative process closer to the general public.

In this context we introduce Sketch Arm, a custom closets rapid prototyping tool that, based on industry expert knowledge, provides the potential customer the ability to create a completely customized design proposal. Thus, as a starting point we performed a study of the creative process carried out by practicing designers (through a series of interviews with the experts) in order to extract a set of design criteria or guidelines, adopted as universal by furniture professionals, which define aspects such as standard closet dimensions or the relative position of the different compartments within the closet. Subsequently, and supported in these conventions, we develop Sketch Arm functionalities in search of a solution aimed at simplifying design tasks, improving user-system interaction, and the assistance or orientation of the individual during prototyping. In short, we aim to define and implement a completely new and satisfactory user experience.

2. State of the art

To achieve the goals set in relation to user experience, we use Human-Computer Interaction (HCI) and Artificial Intelligence techniques in order to (i) design a natural interaction system that allows the user a simple and intuitive use of the tool and (ii) implement a functional module for design support, capable of generating possible closet layouts based on specific storage needs. To implement these functions we carry out a preliminary study that allows us to understand and choose the most suitable techniques and/or algorithms in each case.

2.1. Gesture-based interaction for improving the user experience

The proliferation of mobile devices and the democratization of multi-touch technology have meant a jump from Graphic User Interfaces (GUIs) towards gesture-based ones.

By definition, gesture-based interfaces are intuitive because they are based on one of the oldest known human interaction methods, gestures. Meanwhile, touchscreen-based interaction is considered possibly the most straightforward HCI technique [1] because the information visualization and control are performed through a single medium, the display.

The ability to touch and manipulate data directly on the screen, without the intermediation of any additional device, is very attractive for the end user: direct manipulation style favors IT low-skilled users [1, 2], thanks to a smooth learning curve. For its part, gesture-based interaction is an extremely intuitive interface (based on human inherent capabilities) for using applications that involve any artistic work.

However, the use of both interaction methods in human-machine communication does not ensure end user’s satisfaction. We must design a global interaction in which each element has a purpose and adds value to user experience.

2.2. Layout problem

Although the spatial layout problem has been treated in many ways by the authors, it is often identified as an optimization problem, which seeks the best arrangement of a series of elements. In this search process, each element is assigned a position and orientation that minimize the layout cost and meet certain placement constraints. The layout problem formulated in these terms is seen as a generalization of the quadratic assignment problem, and therefore, NP-complete. This implies, firstly, that the time required for calculating the solution is prohibitive when the addressed problem has high complexity and, secondly, that the chances of obtaining the exact solution to the problem in a polynomial time are slim.

In recent years, several papers about furniture layouts generation inside buildings [3, 4] and automatic floor plan design of residential buildings, according to a series of interior design rules, [5] have been published. These papers primarily arise in the field of video games in order to accelerate the creation of urban settings. They are seeking to
enrich or refine previous systems that either required constant user interaction, or did not take into account ergonomic and habitability factors for creating layouts.

However, the layout problem is not restricted exclusively to architecture, it has been studied in multiple areas of knowledge. Known as packing, packaging, container loading, pallets loading or spatial organization, the layout problem is addressed in relation to tasks as diverse as document layout [6, 7], graph nodes layout [8], GUI layout [9] or integrated circuit components layout [10].

To select the most appropriate method in order to optimize the closet inner space, we consulted Jonathan Cagan’s survey about layout problems [11]. In this work, Cagan establishes a classification into 6 main categories of layout algorithms available in the literature: (i) heuristic rule-based algorithms, (ii) traditional optimization algorithms, (iii) genetic algorithms, (iv) simulated annealing algorithms, (v) extended pattern search algorithms and (vi) a last category that encompasses hybrid algorithms. Withal, none of these mentioned approaches is shown in the study as an absolute solution to the layout problem. That said, and since we want to achieve a good balance between performance and quality, we shall have to use a simulated annealing algorithm, able to avoid false optimum. Its computation time will depend on our expertise in configuring the algorithm and on its proper adaptation to the problem to be addressed.

2.3. Simulated Annealing. An overall view

The simulated annealing algorithm was proposed by Scott Kirkpatrick, C. Daniel Gelatt and Mario P. Vecchi in [12] from the work of Metropolis in the field of statistical thermodynamics [13]: based on existing similarities between key concepts in the physical annealing process of solids and factors involved in finding solutions in combinatorial optimization problems.

This metaheuristic approach aims to avoid being trapped in a local minimum. To do that, it allows uphill moves as an escape mechanism. Analogously to the physical process, there is a control factor, called temperature, which guides the search process: the lower the temperature, the lower the frequency of uphill moves and, therefore, the greater the convergence of the system towards a global optimum (only if the algorithm is convergent).

The algorithm seeks to minimize the value of the objective function by following an iterative search through the solution space. For this it is necessary to design an optimal cooling schedule that considers a decrease in temperature slow enough to find the global minimum. This objective function value represents the goodness of the proposed interim solutions and, therefore, determines whether they are accepted or discarded. For each temperature value the algorithm produces a series of movements or disturbances of the existing solution. If the movement leads to a better solution, it is considered desirable and the solution is accepted. In case it leads to a worse solution, the movement is not necessarily ruled out: it is possible to accept it based on a calculated probability.

3. Sketch Arm

Sketch Arm is a graphical application based on OpenGL ES and developed over iOS (Apple’s mobile operating system) that provides the end user a collection of tools for drawing up a full sketch of his closet. The design process begins with the creation of the room or wall where the closet will be installed, and then the different constituent elements of the closet are gradually incorporated.

The structure of the system is based on a Model-View-Controller design pattern (Fig. 1) where:

• The **model** encapsulates the data that define the application state and the structural representation of the closet.
• The **view** is the interface to the user. In this sense, it is responsible for capturing user interaction and rendering screen elements.
• The **controller** represents the design process logic, mapping user interaction to model and view update operations.
Although design logic modules are decoupled, they must maintain two-way communication so that: (i) the user can access the automatic layout design module at any point in the design process, (ii) the system is able to visualize the model created by itself and (iii) the user has the possibility of changing this model.

3.1. Closet structural representation

The use of grammars for problem modeling has been applied in numerous researches, mainly in the field of Computer Aided Architectural Design [14-16] but also for treating problems close to ours [17].

In [17], Manfred Lau introduces a system able to produce furniture components from the grammar-based formalization of a three-dimensional model. Supported by Lau’s work, we also use a grammar to propose a 3D closet definition as a common link between the different functional modules in Sketch Arm. The proposed model includes information about what elements make up the body of the closet and how they are organized, but it does not include information about secondary components like the front and back cover.

The grammar is defined as $(N, \Sigma, P, S)$ where $N$ is the set of non-terminal symbols, $\Sigma$ is the set of terminal symbols for nodes and edges, $P$ is the set of production rules, and $S$ is the non-terminal start symbol. This grammar leads to an undirected graph where each node represents a separate part of the cabinet, and each edge represents a connection between two parts in contact.

We define $N = \{S, B, X, Y, H, M\}$ and $\Sigma = \{h, v, instancia\}$, where $B$ represents a body, $X$ represents a compartment divided horizontally, $Y$ represents a compartment divided vertically, $H$ represents a non-divided empty compartment, $M$ represents a compartment that houses an accessory, $h$ represents a horizontal division, $v$ represents a vertical division, and $instancia$ represents a particular accessory. Below we show the production rules $P$ in Fig. 2.

![Production rules for 3D custom closets](image-url)
Using a grammar specification allows us to address the design process from a procedural modeling paradigm. This paradigm allows: (i) a significant reduction of the amount of data needed to represent the closet, (ii) the autonomous generation of a complete design based on a predefined set of constraints, and (iii) the segmentation and structural analysis of the mesh of the closet.

Additionally, the proposed grammar takes shape in a data structure named Closet that supports the different functionalities that comprise the interactive editing module and enables storage of topological and geometric data necessary to show a complete visual representation of the closet.

Closet structure is conceived as a tree. Each terminal node stores information about a particular storage compartment or Hole, providing also, as appropriate, information about the horizontal and vertical divisions that delimit it (denoted by Shelf and Division respectively). Additionally, terminal nodes are responsible for storing information about the accessories included in the compartments (denoted generically by Instantiable Element).

Similarly, for storing door-related information, we define another tree data structure, virtually identical to Closet, named Door. The specification of this structure comprises essentially two elements: the front panels that make up the door, named Hole, and the horizontal divisions between these panels, denoted by Shelf. Each node includes the geometrical data associated to each panel and, in case of being divided, it also provides references to the resulting panels.

In order to get the data stored for Holes, Shelves, Divisions and Instantiable Elements in both structures, we apply an instantiation of these elements. The instantiation process consists in locating an element defined on a local coordinate system in a particular position of a global coordinate system. To do this, several 3D transformations are applied to each element on the individual axes of the global system. This instantiation paradigm simplifies the specification of Closet and Door, allowing a more efficient generation of a visual representation of the closet.

3.2. Interactive editing module

Having defined the data structures that will be used to store the topological and geometric information of the closet, below we present the creative process carried out by the user to generate these data.

The design process consists of a set of operations that allows the 3D parametric modeling of the closet. This modeling paradigm is common in CAD tools and it involves too much effort and design rigor for a conceptual design tool like Sketch Arm: the aim is not to design a product with the highest level of accuracy, but simply convey a visual representation of the concept.

The design process is developed along four stages. The first three ones deal with the creation and editing of various aspects or elements relevant to the final design, while the fourth one provides the user with a visualization of the resulting design. Each stage results in a functional module that brings the operations required to perform the associated tasks:

- **Room editing module.** As a starting point, a geometric model of the external structure of the closet is generated from a scheme of the room where it will be installed.
- **Closet interior editing module.** Incorporating additional geometric elements, representative of the internal structure of the closet, complements the previously created model.
- **Doors editing module.** Once detailed the structure of the closet, we finish the design by creating the front doors. Initially, a very basic geometry (specifically a solid rectangular prism) is associated with each new created door, and then, optionally, this geometry is segmented into panels and its visual finish is configured.
- **Final design visualization module.** Once the closet editing process is complete, the system provides a visual representation of the resulting geometric model in a 3D scene. The user can interact with the scene to examine the model in detail and evaluate the goodness of the design.

**Interaction system**

As we have seen, Sketch Arm provides the functionality needed to address the design of custom closets holistically. These capabilities are presented through a simple and intuitive graphical interface, which, combined with a direct interaction system that incorporates multi-touch gestures, seeks to facilitate the creative process to all users regardless of their skill in handling tools.
The GUI design aims to provide the final user an immediate access to the different system functionalities. We strip away the superfluous elements to get a clean, clear and minimalist interface that reduces part of its protagonism to give end users an immersive and natural experience.

The tool is designed as a single landscape view that integrates all the interface graphics (Fig. 3).

![Fig. 3. Sketch Arm Graphical User Interface layout.](image)

The interaction between the user and the application is channelled mainly through the central working area. The user uses his finger as a pointing device for creating and editing the different components of the closet:

- **Selecting an object**, either to be modified, or as a reference to introduce new design elements.
- **Activating the tool to be used**. In general, the features associated with tools for creating new objects are implemented as automatic and require no additional interaction beyond the mere activation of the tool.
- **Creating a new element into the scene** by clicking on the working area. There are certain tools that, once activated, demand a subsequent press on the editing panel to create the new element at a specific location in the scene.

The act of touching is a very intuitive metaphor for selecting certain GUI elements. However, the finger contact surface is usually much larger than most of these elements. In addition, touch devices, which support this kind of interaction, generally lack the precision we get with traditional indirect pointing devices. These problems may represent a significant disruption in user experience. To overcome them, three complementary approaches are taken: (i) the graphical user interface is adapted to touch interaction style by including larger selectable items, (ii) each selectable object in the scene is associated with a larger off-screen selectable area, and (iii) the user is allowed to scale certain areas of interest by using gestures.

Beyond the traditional touch interaction, in the last few years, with the growth of multi-touch hardware devices, certain simple and easily assimilable gesture interaction patterns have been developed. These patterns are common to most of applications running on these devices, to the extent that they are considered current standard. Sketch Arm features gesture interaction as support for the design process, maintaining consistency with standard interaction patterns:

- Two-finger pinch-in and pinch-out gestures allow the user to change the scale of the elements in the scene by zooming in and out.
- Two-finger horizontal and vertical scrolling gesture allows scene panning.
- On editing modules, one-finger movement allows the user to move a selected object around the scene.
• On final design visualization module, one-finger movement allows rotating the scene for a thorough visual inspection of the results.

3.3. Automatic layout design module

The use of a natural user interface allows the user to quickly become familiar with the tool and, therefore, to meet the design process in an agile and successful way. However, any creative process begins with a predominantly negative initial stage in which a person is flooded with feelings such as anxiety, depression, insecurity, etc. [18]. It is necessary to break down in some way this initial barrier that prevents the user’s full satisfaction and may cause the rejection of the tool. In this sense, the application must provide a framework for generating new ideas, providing the user proposals for guiding his designs.

With this aim, Sketch Arm includes an automatic layout design module that provides the individual design proposals based on its storage needs. This module uses a simulated annealing algorithm, which is responsible for generating an optimal layout of the usable space of the closet based on a series of spatial and functional system constraints.

As mentioned in section 2.2, the spatial layout problem is NP-complete and, depending on its complexity, the resolution may require a prohibitive computing time. The automatic layout design module is able to provide an almost immediate feedback thanks to an effective customization of the optimization algorithm and a prior simplification of the layout problem.

3.3.1. Problem definition

Sketch Arm tackles closet design as a creative modeling process in 3D space. In this sense, it seems logical to address the layout problem as a geometric arrangement of elements in a three-dimensional space. However, the problem resolution is a complex search process that involves an unaffordable calculation time, and therefore it should be dismissed.

This first approach highlights the need to simplify the problem. It is necessary to reduce the search space so that the simulated annealing algorithm is able to achieve a quality solution within a reasonable time. The implemented simplification process has two sequential and complementary stages:

Using a different level of abstraction for formulating the problem.

Closet interior is conceived as a rectangular prismatic space where we introduce a series of smaller rectangular prismatic subspaces. Solving the layout problem involves finding an optimal organization of these subspaces that maximizes the occupancy of the container space.

It is considered that each subspace is able to accommodate a single clothing category. We identify six categories:

• Long clothing: coats, dresses, parkas, long skirts, jackets, etc.
• Short clothing: trousers, shirts, blouses, skirts, vests, etc.
• Folded clothing: cardigans, pullovers, t-shirts, etc.
• Underwear: underpants, panties, bras, stockings, socks, etc.
• Accessories: scarfs, caps, gloves, belts, etc.
• Footwear: shoes, sneakers, boots, etc.

This classification is derived from the collected expert knowledge. This knowledge allows creating an association between each category and its corresponding subspace-characteristic data set: the standard height range to accommodate a particular clothing category and the most suitable component (shelf or accessory) for organizing it. Table 1 shows the specific data related to each category.
Table 1. Clothing-categories-related characteristic data: height and accommodation accessories of the associated subspaces

<table>
<thead>
<tr>
<th>Clothing category</th>
<th>Height</th>
<th>Accessory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long clothing</td>
<td>120-150 cm</td>
<td>Hanging bar</td>
</tr>
<tr>
<td>Short clothing</td>
<td>80-110 cm</td>
<td>Hanging bar</td>
</tr>
<tr>
<td>Folded clothing</td>
<td>30-40 cm</td>
<td>Shelf</td>
</tr>
<tr>
<td>Underwear</td>
<td>24 cm</td>
<td>Drawer</td>
</tr>
<tr>
<td>Accessories</td>
<td>18 cm</td>
<td>Drawer</td>
</tr>
<tr>
<td>Footwear</td>
<td>18 cm</td>
<td>Shoe shelves</td>
</tr>
</tbody>
</table>

This descriptive information is modeled as pairwise relationships, where Clothing Category is the common link among them. Specifically, we consider three relationships: Subspace-Clothing Category, Clothing Category-Height and Clothing Category-Accessory.

Reducing dimensions of the problem.

Specifying the inner layout of the closet consists of defining the position coordinates in 3D space of the subspaces within. Therefore, we see that in the problem, aside from a lot of variables, these variables can be assigned a large number of values. We are undoubtedly facing an overly large search space.

To reduce the size of the search space, the original layout problem is reformulated as one-dimensional, ignoring X and Z dimensions. In this new abstraction we consider that:

- All the subspaces share the same size and position in Z-axis (Fig. 4).

Fig. 4. Simplifying layout problem, originally conceived as three-dimensional, to a two-dimensional one. Since all the compartments of the closet share the same size (depth) and position (the origin points of all are located in the same XY plane) in Z-axis, depth information is ignored.

- All the subspaces share size and position on the horizontal axis with their container body (Fig. 5).
Fig. 5. Simplifying layout problem to a one-dimensional one. The width is the same for all bodies and it is calculated by dividing the total width of the closet by the amount of existing bodies (three in the example). This value also determines the width of all subspaces that make up the internal layout of the closet.

- Subspaces included in the same body are located in adjacent positions on the vertical axis, so we avoid overlap or gaps between them.

Covered by these conventions, we finally define the closet inner space layout problem as the process of finding the optimal disposition of subspaces, able to satisfy some predefined constraints. This search process consists of generating successive permutations of the subspaces, these being rectangular areas that take up positions in a vertical plane parallel to the front of the cabinet.

3.3.2. Proposed Simulated Annealing

Simulated annealing method, being conceptually simple, is a powerful and flexible approach that, in general, can address a wide variety of combinatorial optimization problems. However, it’s necessary to carry out a preliminary customization in order to apply the algorithm to a specific problem.

Customizing the algorithm implies a decision-making that affects both cooling-schedule-related factors and problem-specific factors. Therefore, specifying the proposed solution means defining clearly the representative elements of the layout problem (the solution representation, the method for calculating the initial solution, the proposed transition mechanism, and the objective function definition), and the elements that guide the temperature evolution during the algorithm execution (the initial temperature, the cooling mechanism and the cooling speed, and the stopping condition).

Solution representation

The solution to the proposed layout problem must provide information about the absolute and relative position of the considered subspaces. The representation used for modeling the arrangement of these subspaces within the closet structure consists of an n x m matrix. Each i subspace meets the following criteria:

1. \( n, m \in \mathbb{Z}_+ \), where \( n \) is the number of bodies and \( m \) is the total amount of subspaces,
2. \( idBody_i \in [1, n] \),
3. \( posBody_i \in [1, m] \),
4. \( i \in [1, m] \),
5. \( [idBody_i, posBody_i] \in [0, m] \)
The \((idBody_i, posBody_i)\) pair defines each matrix cell position, while cell content is denoted by \([idBody_i, posBody_i]\). In addition, \(idBody_i\) and \(posBody_i\) indexes are significant in the problem context: \(idBody_i\) identifies the container body of the \(i\) subspace, and \(posBody_i\) provides information about its position within the body. The integer value stored in each position of the matrix, denoted by \(idSpace_{idBody_i,posBody_i}\), identifies unequivocally the corresponding subspace to each cell.

Fig. 6. The corresponding matrix representation to the inner layout of a closet with three bodies. Each row represents a closet body, and each cell represents a specific compartment.

In Fig. 6, we can see that, except for 0, the value stored in each matrix cell is unique. The assignment of 0 to the id associated with a position aims to maintain the matrix structural integrity and it’s not reflected on the visual representation of the closet.

In addition, from the example shown above we can infer several principles that govern the mapping of the matrix representation to a closet geometric model:

- Mapping is a bijective function: each matrix representation leads to a single geometric model and each geometric model is represented by a single matrix.
- Contiguous matrix positions represent adjacent subspaces.
- Every matrix cell represents a subspace located at the top of the closet if \(posBody_i = 0\).
- Given two adjacent subspaces, \(i\) and \(j\), we say that \(i\) stands on top of \(j\) if it is satisfied that

\[
\begin{align*}
1. & \quad idBody_i = idBody_j, \\
2. & \quad posBody_i = posBody_j - 1, \\
3. & \quad idBody_i \in [1, n], idBody_j \in [1, n], posBody_i \in [1, m - 1], posBody_j \in [2, m],
\end{align*}
\]

- Similarly, we consider that \(i\) is under \(j\) if it is satisfied that

\[
\begin{align*}
1. & \quad idBody_i = idBody_j, \\
2. & \quad posBody_i = posBody_j + 1, \\
3. & \quad idBody_i \in [1, n], idBody_j \in [1, n], posBody_i \in [2, m], posBody_j \in [1, m - 1]
\end{align*}
\]

The matrix representation is a simple topological model, focused on the spatial organization of the closet, which ignores subspace-specific features (its dimensions and the associated clothing category) required to generate the closet geometric model. To bridge this gap, the algorithm must use the information contained in Subspace-Clothing Category and Clothing Category-Height relationships, introduced in section 3.3.1.
Initial solution calculation

In order to generate an initial layout it’s necessary to specify the structure of the solution matrix by defining the number of bodies and subspaces that best fits the user’s reality.

In the initial dialogue with the system, the user provides the width and height of the closet (denoted by $widthw$ and $heightw$ respectively) and the amount of clothing to be stored for each of the categories considered in the problem definition. Each category is assigned an integer value that represents, in percentage terms, the ratio between the amount of clothing in that category and the total amount of available clothing. This commitment facilitates user-system communication and simplifies the calculation of the number of bodies and subspaces that make up the internal structure of the closet.

Based on this information, the established procedure allows calculating both the number of bodies (and its dimensions) and the number of subspaces in each category:

- Assuming the standard body width is 60 cm, the desirable number of bodies is determined by the expression
  \[ n = \left\lfloor \frac{widthw}{60} \right\rfloor \]
- The $n$ closet bodies share the same width and height, denoted by $widthB$ and $heightB$ respectively.
  \[ heightB = heightw \]
  \[ widthB = \frac{widthw}{n} \]
- To calculate the number of subspaces in each category we need to know in advance which portion or fragment of the total area is allocated for each category. Thus, once we know the total available area
  \[ Area_w = widthw \times heightw \]
we calculate the area under each category $cat$ by using the percentages provided by the user, denoted by $percent_{cat}$

  \[ \forall cat \in [1,maxCat] \quad Area_{cat} = \frac{percent_{cat} \times Area_w}{100} \]

where $maxCat$ is the maximum number of clothing categories.

Finally we calculate the number of subspaces resulting from the calculated areas.

  \[ \forall cat \in [1,maxCat] \quad amount_{cat} = \left\lfloor \frac{Area_{cat}}{widthB \times heightMax_{cat}} + 0.5 \right\rfloor \]

where $heightMax_{cat}$ is the maximum allowed height for elements in category $cat$.

Once defined the solution matrix, we create the initial layout by assigning each subspace to a single matrix cell. In each iteration, a single subspace is randomly selected and it’s assigned to the first available position within a random body.

Transition mechanism

Succeed in the optimization process involves defining an appropriate strategy search for the proposed
representation model. This neighbourhood generation mechanism must be able to construct the solution space by successively applying perturbations.

The transition mechanism of the proposed simulated annealing algorithm includes a set of movements that make the subspaces continuously permute. These movements allow an effective exploration of the search space, including both operations that slightly alter the existing solution and operations that change the layout significantly.

**Kicking subspaces out.** This operation involves moving a random subspace from its original position within a container body to a target position in the bottom of another randomly selected body (Fig. 7).

![Fig. 7. Kicking out operation example.](image)

In terms of matrix representation, this movement can be expressed as a relationship

\[(id_{Body_i}, pos_{Body_i}) \rightarrow (id_{Body_j}, pos_{Body_j})\]

satisfying

1. \(\exists j \mid [id_{Body_j}, pos_{Body_j}] = 0, [id_{Body_k}, pos_{Body_k}] \neq 0, id_{Body_j} = id_{Body_k}, pos_{Body_k} \in [1, pos_{Body_j}]\)
2. \(\exists i \neq j \mid pos_{Body_i} \neq pos_{Body_j}, id_{Body_i} \neq id_{Body_j}\)

where \(i\) and \(j\) represent two layout subspaces.

**Shifting subspaces.** Shifting subspaces operation is the translation of a random subspace from its original location to a new randomly selected location within the same body (Fig. 8).

![Fig. 8. Shifting operation example.](image)

Expressed mathematically, as in the kicking out operation, we have a function

\[(id_{Body_i}, pos_{Body_i}) \rightarrow (id_{Body_j}, pos_{Body_j})\]

but in this case it must fulfil that

\(\exists \) pos_{Body_j} \mid [id_{Body_j}, pos_{Body_j}] = 0, [id_{Body_k}, pos_{Body_k}] \neq 0,
Swapping subspaces. Swapping subspaces means interchanging positions of two randomly selected subspaces within the same body (Fig. 9).

\[
\begin{array}{cccccc}
6 & 7 & 8 & 4 & 13 & 0 \\
5 & 9 & 10 & 1 & 2 & 11 \\
3 & 12 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
6 & 13 & 8 & 4 & 7 & 0 \\
5 & 9 & 10 & 1 & 2 & 11 \\
3 & 12 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Fig. 9. Swapping subspaces operation example.

Swapping subspaces between bodies. Like in the previous movement, the positions of two randomly selected subspaces are interchanged. The difference is that both subspaces are included in different bodies (Fig. 10).

\[
\begin{array}{cccccc}
6 & 7 & 8 & 4 & 13 & 0 \\
5 & 9 & 10 & 1 & 2 & 11 \\
3 & 12 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccc}
6 & 12 & 8 & 4 & 13 & 0 \\
5 & 9 & 10 & 1 & 2 & 11 \\
3 & 7 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Fig. 10. Swapping subspaces between bodies operation example.

To determine whether a new state is accepted or not, the objective function value is taken as reference. In case the value for the new state is lower than the value for the existing one, the transition to the new state will be made. If this condition is not met, the acceptance or rejection of the new feasible state will be determined by the value of the probability function

\[
\frac{1}{1 + \exp \left( \frac{\Delta C}{\max (T)} \right)}
\]

where $\Delta C$ is the difference between the objective function value for the candidate solution and the objective function value for the current solution, and $T$ is the temperature value.

Both factors take only positive values. This makes the acceptance probability only take values between 0 and 0.5. The lower the value of $T$ or higher the value of $\Delta C$, the lower the acceptance probability.

Objective function

Optimization consists in finding a solution that minimizes the objective function value. This function must be representative of the problem and include specific defining constraints. Constraints violation causes an objective function value increase. This increase depends on the level of permissiveness of the algorithm about each type of constraint violated.

In order to model constraints hardness we assign a weighting factor to each type of restriction. We use a very high weight for constraints that can never be violated by a feasible solution. By contrast, we use lower weights for constraints whose violation does not necessarily involve the rejection of the solution. At the beginning of the algorithm the weight of each constraint is set, and those values remain unchanged during all iterations.

The proposed function takes into account a number of criteria for guiding the closet internal layout design process. Here we will see which are these criteria and how do we include them to the function formulation.
Non-useful space. We must maximize the use of the available space in each body of the closet. The value of the objective function increases as the non-useful space increases. We define this non-useful space as the available space within any body that is not used to accommodate any clothing category (in terms of matrix representation, a cell that contains value 0).

Being \( h_{i,j} \) the height of a subspace \( j \) within in a body \( i \), we defined the non-useful space cost as

\[
C_o = \sum_i \left( h_B - \sum_j h_{i,j} \right)
\]

\( C_o \) will increase only if the difference between the body height \( (h_B) \) and the overall height of the subspaces within that body is positive.

Space overflow. In any case the subspaces within a given body can represent an area larger than the available space within the body. To encourage that every subspace is completely embedded in its container body, we increase the space overflow cost whenever a subspace exceeds the space defined by the corresponding body.

\[
C_d = \sum_i \left( \sum_j h_{i,j} - h_B \right)
\]

Unlike the non-useful space cost, \( C_d \) will increase if the difference between the overall height of the subspaces within a given body and the body height is positive.

Pairwise constraint. There is a pairwise relationship between compartments of certain clothing categories. Thus, certain compartments hold contiguous positions within the same body: long clothing compartments and footwear storage compartment, short clothing compartments and shelves for folded clothing storage, etc. In this sense, we can say that if there is a pairwise relationship between subspaces \( A \) and \( B \), it is desirable or even mandatory, depending on the importance of the relationship, that \( A \) is located on the top of \( B \).

The pairwise cost will increase if there is separation distance between the associated subspaces. Its value is defined by the expression

\[
C_e = \sum_i \sum_j \sum_k \left( p_{A,B} \cdot w_{A,B} \cdot \sum_{l=j+1}^{k-1} h_{i,l} + p_{B,A} \cdot w_{B,A} \cdot \left( \sum_{l=j+1}^{k-1} h_{i,l} + \sum_{j}^{\max} h_{i,l} \right) \right)
\]

where \( A \) and \( B \) are the categories associated respectively to subspaces \( j \) and \( k \), both located within a body \( i \), \( p_{X,Y} \) is a Boolean value that indicates whether or not there is a relation between two given categories \( X \) and \( Y \), \( w_{X,Y} \) is the weight of the relationship, and \( \max \) is the maximum number of subspaces within the body.

Proximity to ends. It is common to reserve space for certain types of compartments within the internal distribution of the closet, close to (or near) the top of the closet or close to the ground. The clearest example is the footwear storage compartment, usually installed on the floor or on the bottom shelf of the enclosure frame.

In terms of the objective function, we define the proximity to ends cost as

\[
C_p = \sum_i \sum_j \left( \text{dir}(A, \text{Top}) \cdot \sum_{k=1}^{j-1} h_{i,k} + \text{dir}(A, \text{Down}) \cdot \sum_{k=j+1}^{\max} h_{i,k} \right)
\]
where $\text{dir} (cat, \text{end})$ is a Boolean function that indicates whether or not it is desirable that the subspaces belonging to a given category $cat$ are located in positions close to $\text{end}$, and $\text{max}_i$ is the maximum number of subspaces included in the body $i$. The value $C_p$ will depend on the distance between each subspace and the associated target end.

**Subspaces grouping.** Compartments intended to store the same type of clothes tend to be located in adjacent positions within the same body. Thus, for example, drawers are usually grouped into drawer units and removable shelves for footwear storage are often stacked at the bottom of the closet.

For each "source" subspace $j$, we seek a possible "brother" subspace relS. The subspaces grouping cost is determined basically by the separation distance between the two subspaces. Additionally, the location of the subspaces in different bodies is penalized. We define the expression for calculating the cost as

$$C_a = \sum_{i} \sum_{j} \left( \sum_{i=1}^{\text{maxB}} \sum_{k=j+1}^{\text{relS}} \text{subCat}(A) \ast \text{areDifCat}(A, B) \ast h_{i,k} + (\text{maxB} - i) \ast h_B \right)$$

where $A$ and $B$ are the categories associated to the compared subspaces $j$ and $k$ respectively, $\text{maxB}$ is the body where subspace relS is located, $\text{subCat}(cat)$ is a Boolean function that indicates whether or not there are subspaces of category $cat$ in the layout, and $\text{areDifCat}(cat, cat')$ is also a Boolean function that indicates if a pair of categories, $\text{cat}$ and $\text{cat'}$, are different.

Given the above costs, we define the objective function as

$$C = w_o \ast C_o + w_d \ast C_d + w_e \ast C_e + w_p \ast C_p + w_a \ast C_a$$

The weight associated with each term of the objective function is determined by the $w$ coefficients. For the final implementation we set $w_o = 1.0$, $w_d = 1.8$, $w_e = 1.0$, $w_p = 1.0$, and $w_a = 0.0005$.

**Cooling schedule**

The design of the cooling schedule should aim to reach the convergence of the simulated annealing algorithm. We’ll reduce the amount of time required for the algorithm to find an optimal solution insofar as we are able to define an effective cooling schedule.

The definition involves specifying the initial value for temperature and describing the factors that guide its update during the iterations of the algorithm.

Below we detail the configuration of the schedule elements. This configuration is static and remains unchanged throughout the execution of the algorithm.

**Starting temperature $T_0$.** Generally, the starting temperature must be raised because it affects greatly the number of iterations of the algorithm. In our case, the starting value set for the temperature is 80.

**Annealing parameter $k$.** As the algorithm progresses, the temperature decreases gradually according to the exponential function

$$T = T_0 \ast 0.95^k$$

being $k$ the number of iterations already executed until a given time.

**Reannealing interval $R_n$.** Temperature does not decrease steadily: at certain points in the execution of the algorithm, it may be interesting to raise the temperature to prevent the search from getting caught in a local minimum. This temperature increase consists simply of reducing the value of the annealing parameter, once
accepted a concrete number of solutions. The amount of solutions that marks the heating of the system is called reannealing interval and is denoted by $Rn$.

The reannealing interval conditions the behaviour of the algorithm: for high values of $Rn$ the algorithm has stochastic behaviour and for small values its behaviour is deterministic. In the final implementation of our algorithm, we assign the value 4000 to this parameter in order to promote diversity among the solutions.

**Stopping condition.** The optimization process is considered completed when there is no expectation of obtaining a better solution by decreasing the temperature again. At this point, the existing solution is taken as the final solution to the problem.

Our simulated annealing algorithm uses the objective function value as stopping condition. The algorithm stops when the average change in value of the objective function in $It$ iterations is less than the value taken as reference $5\alpha$. Specifically, the value assigned to $5\alpha$ is $10^{-6}$.

### 4. Results

This section presents the results obtained by the simulated annealing algorithm for different kinds of users and the same dimensions of the closet. In particular, the considered profiles are Business, Casual and Sport, being the width and height of the closet 175 cm and 225 cm respectively.

In each simulation, we define a particular user profile by specifying the percentages for the considered clothing categories. We produce 20 feasible layouts for each profile, assigning to the rest of the algorithm parameters, the configuration values given in section 3.3.2.

For testing we used a MacBook Pro with an Intel Core 2 Duo 2.26 GHz processor, 4 GB of 1,066 MHz DDR3 SDRAM and a graphics processor NVIDIA GeForce 9400M with 256 MB of DDR3 SDRAM shared with main memory.

#### 4.1. Business closet

The suit is undoubtedly the main piece of clothing of a businessperson. In general, this kind of people perfectly covers his everyday work with two or three classic suits. Usually, they also have an extensive collection of shirts and ties, with multiple colours and patterns, to match the suits. Their “can’t-miss” pieces of outerwear are a raincoat and a long woollen coat for cold days. With respect to footwear, they usually have one or two pairs (depending on suit colours) of classical laced shoes. For informal events, the common outfit consists of pullover, shirt, chinos or jeans, jacket and any type of footwear that suits the overall outfit.

Most of the closet space should be used for hanging clothes: it will be interesting to have a specific compartment for suits and coats, one for pants, and a third one reserved for shirts and ties. The rest of the layout should be mainly used for folded clothes storage, minimizing the space for accessories and underwear. With regard to footwear, although the stored amount is supposed to be low, it may be interesting to separate the suit shoes from casual shoes.

To generate a layout that fits the closet model we have just described, we assign 35 to short clothing percentage, 15 to long clothing percentage, 37 to folded clothing percentage, 5 to underwear and footwear percentages, and finally 3 to accessories percentage. In Fig. 11 we show some results obtained by the algorithm for these settings.

![Fig. 11. Feasible layout proposals of a closet for a businessperson.](image)
4.2. Casual closet

A basic casual closet consists essentially of modern clothing with a sporty touch. Predominance of T-shirts, casual shirts, polo shirts, sweaters and cardigans in top clothing; and basic jeans (in multiple colours and styles) in bottom clothing. In outerwear, there are a wide variety of clothes: wool duffle coat or feather-filled anoraks for cold days in winter, windbreaks for halftime, leather jackets, denim jackets, etc. Footwear is also very diverse and includes moccasins, sneakers, nautical shoes, boots and, in brief, any comfortable footwear. This outfit is usually completed with accessories such as scarves, hats, caps, gloves or hats.

According to the above ideas, we can say that casual clothing consists basically of folded clothing. The amount of footwear, accessories and underwear increases with respect to the profile discussed in the previous section, and in contrast, the number of clothes that need hanging storage is reduced. In order to create a closet layout suitable for this user profile, we set 35 to folded clothing percentage, 25 to shot clothing percentage, 15 to long clothing percentage, 10 to underwear and footwear percentages, and 5 to accessories percentage. Fig. 12 shows the generated layout solutions.

4.3. Sport closet

Every sport lover clothing is made up of technical (comfortable and durable) but versatile clothes. Shirts, tracksuits, sweatshirts and sweatpants are unavoidable. Outerwear is limited almost exclusively to one or two items of clothing used in sports such as trekking or snowboarding, and boots and running shoes are the usual footwear.

A closet designed for this kind of user should be mainly used to store folded clothing. The storage space reserved for hanging clothes and underwear can be minimal, but it’s desirable to have a considerable space to accommodate footwear. Accessories lose prominence and there is no place for them in the layout. That said we set the folded clothing percentage in 50, the short clothing percentage in 10, the long clothing percentage in 20, the underwear percentage in 5, the footwear percentage in 15 and the accessories percentage to 0. In Fig. 13 we see the different layouts proposed by the algorithm for this profile.

![Fig. 12. Closet layouts suitable for casual clothes storage.](image)

![Fig. 13. Closet designs proposed for a sporty user.](image)
5. Conclusions and future work

As a result of the work carried out we developed Sketch Arm, a non-professional-user oriented software solution that allows a rapid and intuitive prototyping of a custom closet from the user’s preferences and needs.

The developed system is based on multi-touch and gestural interaction and uses metaheuristics algorithms with the aim of making the creative process easily accessible to non-experts by simplifying the tasks involved in the design, improving system-user interaction and providing support.

Below we resume the main contributions of our work:

• Based on Lau’s work [17], we have proposed a formal model for a 3D closet as a common link between the several functional modules.
• We have defined a data structure that represents the structural complexity of a closet. This structure supports closet customization and makes it possible to efficiently edit and display the closet by using the instantiation paradigm.
• We have designed and implemented a set of specific algorithms for interactive editing of the different structural elements of the closet.
• We have developed a 3D visualization core for real-time and interactive rendering of the most relevant parts of the closet at each stage of the design process.
• We have designed an interaction system that consists of using a simple and intuitive GUI with a gesture-based natural user interface.
• We have developed an automatic layout design module responsible for providing the end user a series of design proposals that meet his storage needs. In order to address the closet inner space layout problem and get some variability between the solutions created by the system, we have used a stochastic optimization algorithm, specifically a simulated annealing algorithm. Tests in section 4 yield very positive results and show the system’s ability to generate different user-specific layouts (Fig. 14).

Although our work is extensive, there are some related issues that may be interesting to address in future works:

• The definition of a formal model provides segmentation and structural analysis of the geometric mesh of the closet, supporting a wide range of potential interesting features: (i) the search for market alternatives that fit each proposed closet, (ii) model segmentation for CAD/CAM editing, (iii) assembly instructions generation, and (iv) an estimate of the closet manufacturing materials cost.
• Improving the 3D visualization core by getting a better rendering quality maintaining the overall performance of the application. To do this, we propose the use of shaders.
• Regarding to the proposed layout problem definition, it would be interesting to reformulate some accepted conventions: (i) considering the existence of subspaces narrower than their container body and thereby the possibility of dividing a body vertically, and (ii) increasing the number of considered clothing categories to

Fig. 14. 3D realistic representation of some complete arrangements generated autonomously by Sketch Arm.
generate more detailed and realistic layouts.
• Incorporating to the neighbourhood generation mechanism of our simulated annealing algorithm a movement that changes the height of the subspaces. Additionally, it would be desirable to define a better strategy (smarter) for this mechanism so that movements involving a major reorganization of the space were allowed only at the beginning of the execution of the algorithm, and operations that change slightly the existing solution were allowed only in the final iterations.

Acknowledgments. This work was supported by project 10DPI305002PR (under the name “Deseño e prototipado conceptual de armarios mediante interfaces avanzados en dispositivos táctiles”) of Xunta de Galicia.

References