

EMOTIVE PRODUCT DESIGN BY SWARM INTELLIGENCE MODELLING

Alexander Nikov¹, Arvind Mohais¹, Selahattin Nesil²

¹Department of Mathematics and Computer Science, University of the West Indies, St. Augustine, Trinidad and Tobago (W.I.), E-Mail: {anikov, amohais}@fsa.uwi.tt

²Department of Electronics Engineering, Fatih University, 34500 Istanbul, Turkey, E-Mail: nesil@fatih.edu.tr

Abstract

The purpose of this paper is to apply swarm intelligence in Kansei engineering and explore ways to integrate it into an industrial product development process. For connecting customer emotional needs mathematically to product design properties a particle swarm optimization (PSO)-based approach is proposed. Using PSO, together with the Huffman algorithm, a generic factor was extracted, and used to define product design features. Emotional customer needs are derived bottom-up using Kansei words. PSO approach is illustrated and validated by a case study for pen design. Participants' emotional responses for 13 different pens with 3 design properties were collected for 12 Kansei words measured in 5-point semantic differential scale. Experimental work in implementing the proposed approach was able to suggest credible design attributes of a pen that would be considered optimal by all of those surveyed. The advantages and further developments of approach proposed are discussed.

Key words: emotive design, integrating feelings in design process, swarm intelligence, particle swarm optimization

INTRODUCTION

Tendencies in product development of today make it likely that many future products will be functionally equivalent and therefore hard to distinguish between for customers [15]. They will decide by highly subjective criteria which product to purchase. One goal for product development in this context is to be able to capture the customer's considerations and feelings regarding products and translate these emotional aspects into concrete design characteristics.

Emotive/affective design attempts to define the subjective emotional relationships between consumers and products and to explore the affective properties that products intend to communicate through their physical attributes [1], [2]. It aims to deliver artifacts capable of eliciting maximum psychophysiological pleasure from consumers, targeting all of their senses. Saul Carliner [1] defined the affective design as the third level of a three-part model of information design. The systematic framework [2] conceptualizes customer affective needs in product design.

The aesthetic intelligence [10] acknowledges that we possess an innate, sometimes subconscious, ability to perceive a wide range of qualities in products that shape our responses to them. These qualities can be purposefully discussed and provide engineering designers a way of structuring the complex field of aesthetic response. A process of designing for the senses can be viewed as a means of providing products with which customers can feel a greater degree of empathy.

The approach [5] is based on respect for the user of electronic products. It uses bodily skills to make electronic interaction more tangible, and, as humans are emotional beings, to make interaction a more fun and beautiful experience. It focuses on those neglected aspects of human-product interaction: perceptual-motor and emotional skills. To get to new innovative products, the interaction problem is dealt with on the level of creating a context for experience allowing for rich aesthetics of interaction.

A number of different methods, such as Quality Function Deployment, Semantical Environment Description, Conjoint Analysis and Kansei Engineering are used in emotive product design. Kansei Engineering helps to identify customer needs, their importance and the technical responses as well as

to conduct benchmarking and to connect the customer needs mathematically to the technical responses [4], [15]. It is applied in many studies on a macro level, a micro level and for verifying purpose. Statistical methods like regression analysis, general linear model, qualification theory type 1, as well other methods like genetic algorithms, fuzzy sets and rough sets are used to give a mathematical connection between customer emotions mapped through Kansei words and the product properties [9], [15]. But there are no applications of swarm intelligence methods in Kansei Engineering for emotive product design. Therefore we propose to use swarm intelligence to analyzing customers' emotional responses with regard to products, and to try to propose design properties that embody those responses.

The term *swarm intelligence* is used in artificial intelligence and evolutionary computation research to mean a variety of things. In this paper it will be used in the context of Particle Swarm Optimization, which is an evolutionary algorithm based on observed social interactions in nature that can be mimicked in order to solve optimization problems [6], [7]. We propose to employ swarm intelligence to aggregate emotional design keywords and to suggest product design properties.

DESCRIPTION OF APPROACH FOR SWARM-OPTIMIZATION-BASED EMOTIVE DESIGN

An approach is proposed, which suggests product design properties based on emotive responses. It uses a combination of Kansei techniques and an evolutionary optimization algorithm called Particle Swarm Optimization. The overall process involves designing a survey to collect data on customers' emotional responses to various aspects of a product and then using the swarm intelligence technique to analyze the collected data in such a way as to propose product design suggestions.

Emotive design steps

Figure 1 shows at a high-level, the set of steps that are proposed for determining desirable design properties of a product. An illustration of these steps will be presented in a case study.

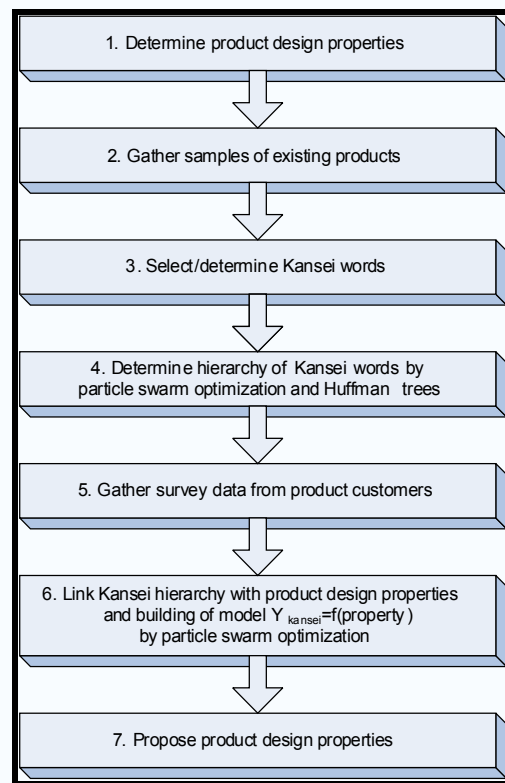


Figure 1. Emotive design steps

Particle swarm optimization algorithm

In this subsection, a general description of the Particle Swarm Optimization (PSO) algorithm will be given using terms of a general optimization function. At a later point in the paper, the PSO algorithm will be used twice, first, in step 4 of Figure 1, where the optimization task will consist of building a binary tree using the Huffman algorithm [5] that minimizes a weight function, and second, in step 6 of Figure 1, where it will consist of fitting a number of binary design parameters to the output of the selected tree from step 4.

Particle swarm optimization (PSO) is an algorithm that is inspired by social interactions of insects, animals, and even human beings [6], [7]. Given the problem of finding a minimiser of the optimization function $f: R^n \rightarrow R$, (where R is the set of real numbers) the PSO algorithm works on a set of candidate solutions called *particles*, “flying” them through the solution space R^n for a certain number of generations until a solution is found, or time constraints force the termination of the algorithm. Each particle $P = (x, v, p_b, f_x, f_{pb})$ consists of a position vector $x \in R^n$, a velocity vector $v \in R^n$, a previous best position $p_b \in R^n$, a fitness value $f_x = f(x)$ and a previous best fitness value $f_{pb} = f(p_b)$.

The set of particles used by the algorithm called population $POP = (P_0, \dots, P_{m-1})$ is represented by a vector of m particles. They exchange information with others in the population, and it is thus that evolution of the candidate solutions takes place. The manner in which information is exchanged between particles is constrained by *neighborhoods*. Each particle P has a *neighbourhood*, which is a subset of the population $N(P) \subseteq POP$ and each element of $N(P)$ is called a *neighbor* of P . The neighbors in $N(P)$ are not determined based on their positions in the search space; they can be completely arbitrary. All of the neighborhoods taken together, i.e. $N = \{N(P_0), \dots, N(P_{m-1})\}$ can be represented by a directed graph $G_N = (V, E)$, where $V = \{0, 1, \dots, m-1\}$ and $E = \{(i, j) | P_j \in N(P_i)\}$. This graph is called a *neighborhood graph* or a *neighborhood structure*.

At each iteration of the PSO algorithm, each particle has its position updated in two steps. First a new velocity vector is computed by among other things, consulting with neighbors to determine their best positions encountered thus far in the run of the algorithm, as well as using the particle's own previous best experience. And then the position vector is updated by adding the newly calculated velocity vector to the current position. These steps are summarized by the following equations, specific formulae for the velocity update equation will be consider below.

$$v' = \text{update-velocity}(x, v, p_b, N(P))$$

$$x' = x + v'$$

A pseudocode description of the basic PSO algorithm is given in Figure 2 and that of the subroutine `find_neighborhood_best_index()` is given in Figure 3. In Figure 2, K_1 and K_2 are called *individual* and *social* constants; they are parameters of the algorithm and represent the degrees of importance allocated to each of the individual and social influences, respectively.

Velocity update equation

The velocity update equation is a very important aspect of the algorithm. The most basic formula used is $v' = v + \phi_1(p_b - x) + \phi_2(p_* - x)$. Modified versions are common. Clerc and Kennedy [2] introduced the *constriction factor* χ making the equation $v' = \chi(v + \phi_1(p_b - x) + \phi_2(p_* - x))$. Regarding this equation, typical values used in experimental work are $\chi = 0.729$ and $\phi_1, \phi_2 \sim U(0, 2.05)$. Mendes altered the equation so that the particle is influenced by all of its neighbors, not just the best one [11], [12], [13]. This scheme, called the *Fully Informed Particle Swarm* (FIPS) operates as follows. For a particle P letting $\phi_k \sim U(0, \frac{\phi_{max}}{|N(P)|})$ for all $k \in N(P)$, and letting $P_{fips} = \frac{\sum_{k \in N} W(k) \phi_k p_{k,b}}{\sum_{k \in N} W(k) \phi_k}$, the velocity update equation becomes $v' = \chi(v + \phi(p_{fips} - x))$. ϕ_{max} is usually

set to 4.1 and χ to 0.729. $W(k)$ is a weighting function that scales the contribution of the neighbor k . It may be desired to weight neighbors' contributions in proportion to their fitnesses, or their distances from the current particle. Alternatively, with $W(k)=1 \forall k \in N(P)$, all neighbors contribute equally to the velocity update equation; in this case, $P_{fips} = \frac{\sum_{k \in N} \phi_k k \cdot p_b}{\phi}$, where $\phi = \sum_{k \in N} \phi_k$.

```

set t=0
initialize_population(m,n,x_max)
initialize_neighborhoods(m)

while ( NOT termination-condition ) {
  for (i=1; i <= m; i++){
    * = find_neighborhood_best_index(N(i))

    for (d=1; d <= n; d++){
       $\phi_1 = K_1 \cdot \text{random}(0,1)$ 
       $\phi_2 = K_2 \cdot \text{random}(0,1)$ 

      for particle  $P_i$ :
         $v_d = \text{update-velocity}(v_d, p_{bd}, p_{*d}, \phi_1, \phi_2)$ 

        enforce_maximum_velocity( $P_i, v_d, v_{max}$ )

         $P_i.x_d = P_i.x_d + P_i.v_d$ 
    }

     $P_i.f_x = f(P_i.x)$ 
    if ( $P_i.f_x < p_i.f_{pb}$ ){
       $P_i.p_b = p_i.x$ 
       $P_i.f_{pb} = P_i.f_x$ 
    }
  }
}

```

Figure 2. Pseudocode for the PSO algorithm

```

find_neighborhood_best_index(N) {
  set idx =  $\alpha$ , for some  $\alpha \in N$ 
  for each  $j \in (N - \{\alpha\})$ 
    if ( $P_j.f_{pb} < P_{idx}.f_{pb}$ )
      idx = j
  return idx
}

```

Figure 3. Pseudocode for finding the neighborhood best index

In [8], it was concluded, counter-intuitively, that weighting neighbors' contributions based on fitness does not help the algorithm. In this paper FIPS with equal weighting is used for all experimental work.

Neighborhood Structures

PSOs typically use one of two neighborhood configurations. The first is called the *ring topology* (cf.

Figure 4a. In this structure, particles are arranged in a circular manner and each is connected to the two particles on its left and right. The second commonly used structure is called the *global topology* (cf.

Figure 4b. Here each particle is connected to all other particles in the population. It is commonly accepted that the global neighborhood leads to too much information being exchanged too quickly, resulting in the premature convergence of the algorithm. Nevertheless, this is a commonly used structure by researchers who are focused on other aspects of the PSO algorithm. In this paper a different type of neighborhood is used: *random* neighborhoods. These are directed, randomly generated graphs. Two parameters affect the nature of the neighborhood graph structure used: the neighborhood size n and the uniform out-degree of the graph k . n is the number of nodes in the graph and hence the number of particles in the swarm. k is the number of neighbors that each particle influences.

Figure 4c shows an example of a graph with 12 nodes and with a uniform out-degree of 2.

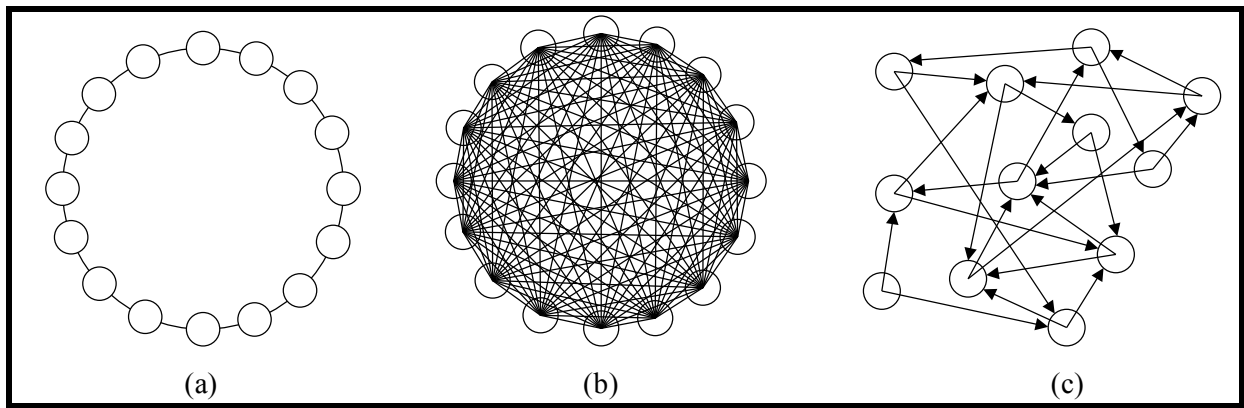


Figure 4. The ring structure, the global structure and a randomly generated structure.

Neighborhood dynamism

Randomized neighborhood re-structuring was introduced in [14]. Under this approach, given values for the n and k graph parameters, the neighborhood graph is subjected to massive perturbations from time to time, according to a *dynamism probability*, p_{dyn} . This drastic change is called *re-structuring* and involves generating an entirely new random graph, but with the same n and k values. At the end of each iteration, this operation is performed with probability p_{dyn} . It is possible to use a fixed value of p_{dyn} for the entire run of the algorithm, or a time varying value may be used. For example, using a linearly decreasing dynamism probability from $p_{dyn} = 1.0$ down to 0.0 over the duration of the run may be used. This means that at the first iteration $p_{dyn} = 1.0$, and at each iteration, it is decremented by $\frac{1.0}{\text{max-iterations}}$, where max-iterations is the maximum number of iterations that would be executed by the algorithm; this is equal to the maximum number of function divided by the population size. In this way, as the algorithm progresses, the probability of a total re-structuring operation becomes less and less.

Specific PSO adaptations for emotive product design

In this subsection, specific models will be given for the types of optimization functions that will be considered for the Emotive Product Design task being considered.

Evolving a binary tree using the Huffman algorithm

Step 4 of Figure 1 represents the creation of a hierarchy of the Kansei words used in the study. This is done by evolving a tree using the Huffman algorithm [5], by means of a PSO. Each particle of the PSO is n -dimensional, where n is the number of Kansei words used in the survey. Each dimension represents a weight assigned to the corresponding questionnaire item.

The weights embodied by each particle are then used to create the tree. Each weight is used to create a leaf node of the eventual tree, and as is usual in such a construction, higher level nodes are created by selecting the two with the smallest weights and aggregating them into a new node whose weight is the sum of the weights of the two children.

In order to evaluate a particle in the PSO, the data vectors (assume there are l of them) of the Kansei survey are introduced as inputs at the leaf level of the Huffman tree corresponding to the particle's weights, and a fitness value is computed using the following formula:

$$fitness(tree_x) = \sum_{i=1}^l \sum_{j=1}^n data_{ij} * pathlength(leafnode_j)$$

The weights of the nodes are not directly used in the computation, instead for each leaf node of the tree, the path length from it to the root is calculated and multiplied by the input value from the survey data corresponding to that node, this computed value is then added to a cumulative sum. After all of the survey vectors have been processed, the cumulative sum serves as the fitness value of the particle.

Fitting binary design parameters to the tree output.

For step 6 in Figure 1, a PSO is again used, this time to fit the product design properties (see step) to the output of the tree from step 4. Here, categorical design properties are converted into binary design properties. For example, if property A can have values of a , b , or c , then this will be converted into three binary properties A_a , A_b , and A_c , each of which can be *true* or *false* (1 or 0). Depending on which value A possesses, the corresponding binary property will be set to *true* and all others to *false*.

For the PSO used here each particle is p -dimensional, where p is the number of binary product design properties. Each dimension of a particle represents a weight associated with one of the binary properties. The fitness of a particle in this phase is calculated using the following formula:

$$fitness(x) = \sum_{i=1}^l \sum_{j=1}^p x_j \cdot bin_data_{ij}$$

where l is the number of data vectors from the Kansei survey and bin_data_{ij} represents the binary properties derived from the categorical product design properties.

CASE STUDY

3.1 Kansei survey

To illustrate and validate the PSO approach for aggregating Kansei words and suggesting product design properties, a case study for design of pen for students of Fatih University, Istanbul was carried out. 13 pens of varying characteristics were presented to 23 students and they were asked to answer a questionnaire based on their feelings about each of the pens. This produced 299 data vectors which will be referred to as $data_{ij}$, where i is a vector index, and j is a questionnaire item index.

The questionnaire consisted of 12 semantic differential Kansei words measured in 5-point scale (cf. Table 1), and 3 pen properties (cf.

Table 2). The first 11 items were used to group the Kansei words by evolving a binary tree using PSO, and the last 3 items (13, 14 and 15) were used for fitting the design properties to the output of this tree. The 13 pens used in the case study are shown in Figure 5.

3.2 PSO parameter values

FIPS-based PSOs with dynamic neighborhood re-structuring for a maximum of 120000 function evaluations were used. As a preliminary step, the neighborhood parameters were kept fixed at $n = 30$ and $k = 4$. These values were shown in other applications to produce high-performance PSO results.

The number of data vectors $l=299$ and the number of questionnaire items considered $n=11$ (corresponding to questionnaire items 1-11). Hence the optimization function for step 4 of Figure 1 becomes:

Table 1 - Scaled questionnaire items.

Questionnaire item number	Questions		
1.	Unimposing	1-2-3-4-5	Imposing
2.	Straight	1-2-3-4-5	Curved
3.	Cool Feel	1-2-3-4-5	Warm Feel
4.	Conservative	1-2-3-4-5	Modern
5.	Ugly	1-2-3-4-5	Beautiful
6.	Unpleasant	1-2-3-4-5	Pleasant
7.	Low quality	1-2-3-4-5	High quality
8.	Feminine	1-2-3-4-5	Masculine
9.	Hard to write	1-2-3-4-5	Easy to write
10.	Bad Design	1-2-3-4-5	Good Design
11.	Thin	1-2-3-4-5	Thick
12.	Unfit for FU students	1-2-3-4-5	Fit for FU students

Table 2 - Categorized product properties.

Questionnaire item number	Product property	Product property values
13.	Pen length	[1] short [2] average [3] long
14.	Pen volume	[1] thin [2] average [3] fat
15.	Pen color	[1] mixed color [1] white [2] blue [3] green [4] yellow



Figure 5. Sample of pens used in the case study

$$fitness(tree_x) = \sum_{i=1}^{299} \sum_{j=1}^{11} data_{ij} * pathlength(leafnode_j)$$

There were 3 categorical design properties (questionnaire items 13, 14 and 15), with 3, 3 and 5 categories respectively, yielding 11 binary design properties. Thus the optimization function for step 6 of Figure 1 becomes:

$$fitness(x) = \sum_{i=1}^{299} \sum_{j=1}^{11} x_j \cdot bin_data_{ij}$$

The desired fitting is illustrated in Figure 6 .

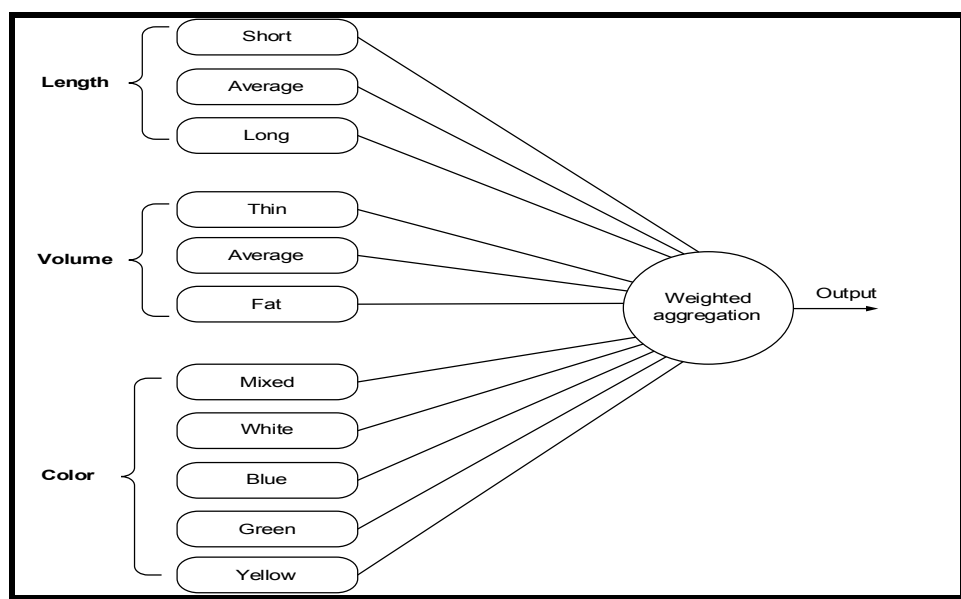


Figure 6. Linking binary design properties to the Huffman tree output

3.3 Case study results

The best result obtained by the PSO algorithm using the specified configuration is the tree shown in Figure 7, where the circled numbers are the weights of the question items q_i .

The fitting of the binary design properties to the output of the tree in Figure 7 yielded the results shown in

Table 3. Based on these results the most important Kanseis for Fatih University students seemed to be long pens with average volume and blue color. This fits also with the logo color of university. Those considered of students to be of less importance is thin pen with average length and white color.

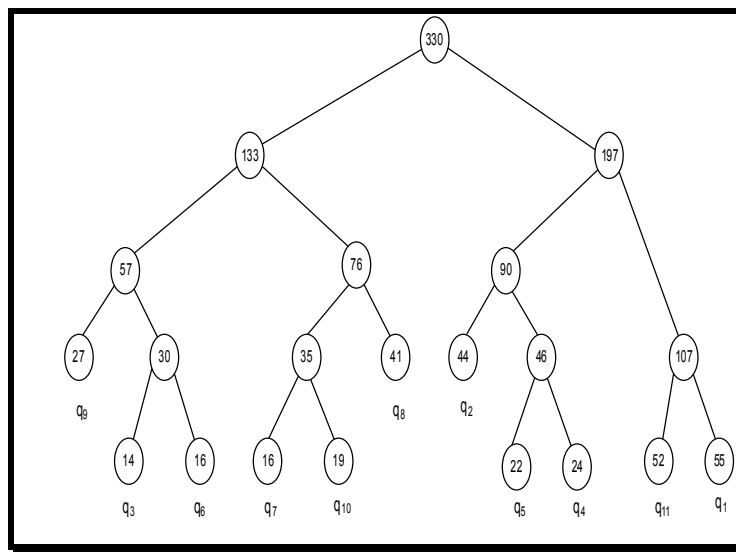


Figure 7. The optimal binary tree

Table 3 Design properties results

Length	Value	Volume	Value	Color	Value	MSE
short	1.4416	thin	0.1451	mixed colors	1.1949	0.0283
average	1.4287	average	0.1581	white	0.9520	
long	1.5257	fat	0.1483	blue	1.1968	
				green	1.1504	
				yellow	1.1074	

CONCLUSIONS

A particle-swarm-optimization-(PSO)-based approach for emotive product design was proposed. The most popular swarm intelligence method, the particle swarm optimization algorithm, was used to model the relationship between Kansei words and product design properties. It is a simple, feasible and versatile approach determining the combination of product properties that could results in designs taking into account customer emotions. A model evolved by the particle swarm predicts the desired values of product design properties. Depending on these values relevant product designs can be proposed.

For illustrating and validating the PSO-based approach a case study of pens for Fatih University was carried out. Relevant Kansei words, three product design properties and sample pens were defined. Based on data gathered from Fatih University students and using the approach, a relevant pen design was determined. The proposed approach can be used for emotive product design.

Some **advantages** of the approach proposed are:

- Particle swarm optimization is a very flexible evolutionary computation method that can be adapted to different optimization tasks and as was done in this paper, it can be re-used in different phases to help solve a problem.
- The use of binary trees built using the Huffman algorithm, provided a standard method of aggregating Kansei words based on numeric weights, without having to consider the semantics of the words themselves.

This work is preliminary and there is room for **further developments**, such as:

- Modified trees using 3, 4 or more children could be considered. Variable numbers of children could also be considered.
- PSO neighborhood and dynamism parameters such as the number of nodes, the uniform-out degree and the dynamism probability could be investigated with a view to improving the optimization of both phases of the approach.
- Attempts could be made to fit the binary design parameters to nodes in the tree, other than the root node.
- Emotional PSOs [3], which presents a modification of the PSO algorithm introducing some psychology factor of emotion into the algorithm may be considered for use in this application.

REFERENCES

- [1] Carliner S. (2000) Physical, Cognitive, and Affective: A Three-Part Framework for Information Design. <http://saulcarliner.home.att.net/id/newmodel.htm> (visited 16 May 2007).
- [2] Clerc M. and Kennedy J. (2002) The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6(1), 58-73.
- [3] Ge Y. and Rubo Z. (2005) An Emotional Particle Swarm Optimization Algorithm. In Proceedings of the International Conference on Advances in Natural Computation (China), Springer Berlin / Heidelberg, pp. 553-561.
- [4] Grimsæth K. (2005) Kansei Engineering. Linking emotions and product features. Undergraduate Thesis, Norwegian University of Science and Technology (NTNU) http://design.ntnu.no/forskning/artikler/2005/artikkel_Kansei_Engineering_K_Grimsath.pdf (visited 16 May 2007).
- [5] Huffman D.A. (1952) A method for the construction of minimum redundancy codes, *Proc. IRE*, vol. 40, pp.1098-1101.
- [6] Kennedy J. and Eberhart R. (1995) Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, IEEE Press, pp.1942-1948.

- [7] Kennedy J., Eberhart R. and Shi Y. (2001) *Swarm Intelligence*, Morgan Kaufmann, San Francisco.
- [8] Khalid H. M. and Helander M. G. (2004) A framework for affective customer needs in product design. *Theoretical Issues in Ergonomics Science*, 5(1), 27-42.
- [9] Lee S. H., Harada A. and Stappers P. J. (2000) Pleasure with Products: Design based on Kansei. In Chapter 16 of *Pleasure with Products: beyond usability*, by W. S. Green (Ed.), Taylor and Francis, pp. 219-230.
- [10] MacDonald A. S. (2001) Aesthetic intelligence: optimizing user-centred design. *Journal of Engineering Design*, 12(1), 37-45.
- [11] Mendes R., Kennedy J. and Neves J. (2003) Watch thy neighbor or how the swarm can learn from its environment. In *Proceedings of the Swarm Intelligence Symposium*, IEEE Press, pp.88-94.
- [12] Mendes R. (2004) *Population Topologies and Their Influence in Particle Swarm Performance*, PhD thesis, Universidade do Minho, Braga, Portugal.
- [13] Mendes R., Kennedy J. and Neves J. (2004) The fully informed particle swarm: Simpler, maybe better. *Transactions on Evolutionary Computation*, 8(3), 204-210.
- [14] Mohais A., Mendes R., Ward C. and Posthoff C. (2005) Neighborhood Re-structuring in Particle Swarm Optimization. In *Proceedings of the Australian Conference on Artificial Intelligence*, Springer, Berlin, pp.776-785.
- [15] Schütte S. (2002) *Designing Feelings into Products: Integrating Kansei Engineering Methodology in Product Development*, PhD Thesis, Linköping University.