An Agent-Based Multimodal Interface for Sketch Interpretation

Sleiman Azar*, Laurent Couvreur†, Vincent Delfosse*, Benoit Jaspart‡ and Christelle Boulanger*
*Laboratory of User Cognition and Innovative Design, B-4000 Liege, Belgium
Email: azar@lema.ulg.ac.be
†Signal Processing Lab, Faculté Polytechnique de Mons, B-7000 Mons, Belgium
Email: couvreur@tcts.fpms.ac.be
‡Montefiore Institut, University of Liege, B-4000 Liege, Belgium
Email: bjaspart@ulg.ac.be

***Abstract*— We present a multimodal interface for sketch interpretation that relies on a multi-agent architecture. The design of the interpretation engine and the different agents are based on a user-centered approach where efficiency measure is defined as user satisfaction. So far, several graphical agents have been implemented for recognizing basic graphical objects (*e.g.*, lines, circles, etc) as well as more complex (*e.g.*, hatches, stairs, captions, etc) in architectural design. Besides, vocal agents have been developed for recognizing spoken annotations (*e.g.*, dimensions) and interface commands. Realistic evaluations with professional users have demonstrated the potential interest of the proposed system.**

I. Introduction

Sketching is an efficient way to express ideas, especially in the early stages of design (*e.g*, in architectural design). Everyone has) already seen someone trying to explain his bright new idea by commenting a sketch on a napkin while sitting at a table in a restaurant. Although hardware for computer drawing exists, Computer-Aided Design (CAD) software classically rely on more constraining interfaces with limited interaction (*e.g.*, keyboard strokes, mouse moves and clicks, etc). Consequently, designers often sketch ideas on paper and transfer them to the computer later in the design process, irremediably impeding their creativeness. Sketching is definitively a more natural design mode and is also an efficient way to annotate and modify existing CAD files.

Most existing solutions for sketch interpretation result from a tradeoff between ease of recognition and freedom of drawing: the more constrained the user drawing style, the more tractable the sketch interpretation. Nowadays, there are attempts to develop more general sketch interpretation systems that can be tailored to specific domains [1], [2] and to possibly combine drawing with other modalities [3], [4].

We present here a 2-D sketch interpretation system that relies on a flexible multimodal multi-agent architecture [5], [6]. It makes the key assumption that the complex design process builds up over simpler behaviors of the designer. For instance, the designer uses low-level graphical objects to draw higher-level objects. The proposed system is composed of task-oriented agents that are designed to recognize these simple behaviors. The user inputs, which can be of different types (*e.g.*, graphical, spoken, etc), are continuously monitored and interpreted by agents with respect to their capabilities. Once a scenario has emerged, it is considered as the original user intention, the recognized object is released and used for updating the application accordingly. The proposed system allows triggering suitable agents depending on the interface context, thereby minimizing the computational burden and maximizing the interpretation robustness. Although the research has targeted architectural design as the domain of interest, the proposed system can be extended to other design fields.

In next section, we first describe the agent-based system architecture. Next, we present the agents that have been already implemented, namely graphical agents and vocal agents. Discussions on the integrated interface are given in section III. Conclusions are drawn in section IV.

II. Proposed Method

A. System Architecture

The system architecture is represented in Figure 1. It is fed with user inputs via various sensors, for example a graphical tablet for capturing pen-based trajectories, a microphone for recording speech utterances, a file access for reading CAD documents, etc. All the inputs are processed by front-end modules to extract low-level data objects. For example, pen strokes are extracted from the pen trajectories on the graphical tablet, acoustic coefficients are computed from the speech signals, graphical objects are read from the CAD files.

The outputs of the front-end modules are then sent on a data bus to which the core interpretation system is connected. It is composed of several agents that are grouped into squads with respect to their nature. When initialized, every agent declares its domain of interest by registration of a list of object types that it can use as inputs. Agents are able to interpret low-level objects or handle them in order to infer higher-level objects, which in turn can be interpreted by other agents. Every agent provides a confidence score for the claimed object. A conflict between agents interpreting the same objects is solved by the negotiation module based on the confidence scores. For example, consider that the user is drawing neighboring vertical strokes that are posted on the data bus. The hatch-agent identifies a hatched zone while the text-agent reads “I” characters. After a few strokes though, the latter will produce a very low confidence score to the recognized text because it corresponds to very unlikely words. Hence, the set of strokes will be labelled as hatches and a hatch object will be posted on the data bus.

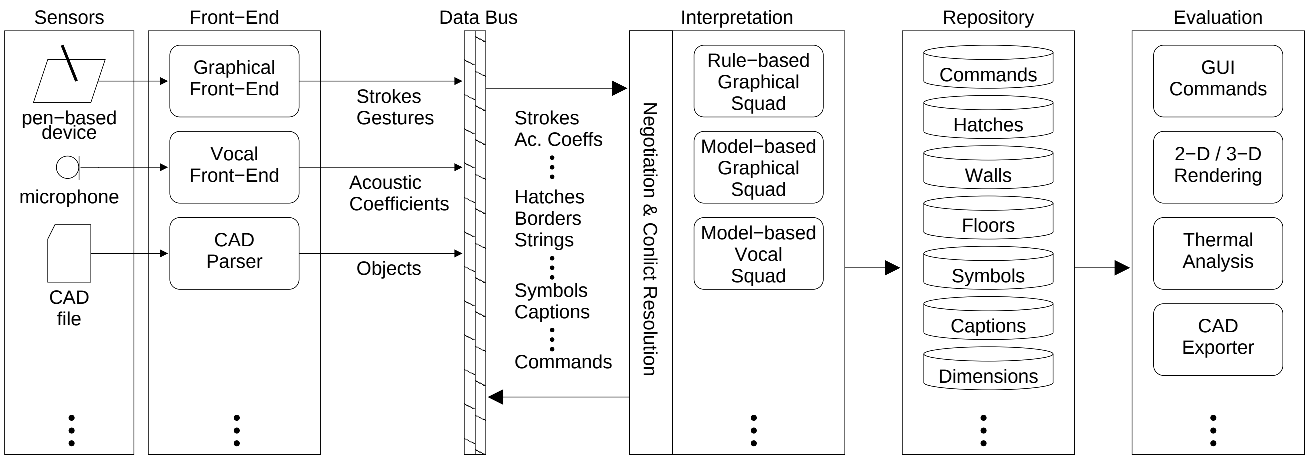


Fig. 1. The architecture of the multimodal multi-agent system for 2-D sketch interpretation.

The life cycle of a data object terminates once no more agent is interested in it. The data object leaves the data bus and is stored in a repository. This object database is used by far-end modules that evaluate data object to provide the user with various services. For example, utterances recognized as commands can be applied by the graphical user interface, graphical objects recognized as floors and walls can be used in a 3-D rendering environment, or to estimate building thermal insulation, etc.

Note that the system architecture includes a socket mechanism to operate the data bus over a communication network and defines common interfaces for running heterogeneous agents written in various languages on remote machines.

In next section, we describe the main squads of agents that have been already implemented. Note that the system architecture is opened to new modality/agent. For example, one can imagine a vision squad that will include a gesture recognition agent, a gaze tracking agent, etc.

B. Rule-based Graphical Squad

Sketching in architectural design can be viewed as a hierarchical process where the designer relies on low-level graphical objects to build higher-level objects. For example, stairs are typically represented by hatches with a crossing perpendicular arrow. Hatches are commonly sketched as parallel straight lines while an arrow is composed of a single straight line ended with a direction mark.

We rely on this sketching ontology to develop the squad of rule-based graphical agents. Every agent is interested by some graphical objects and characterized by a set of rules. A rule can be either a property of an object (*e.g.*, form, size, etc) or a topological relation between objects (*e.g.*, relative position, alignment, etc). As an example, here are the rules for the agent recognizing the “stairs” object in architectural design:

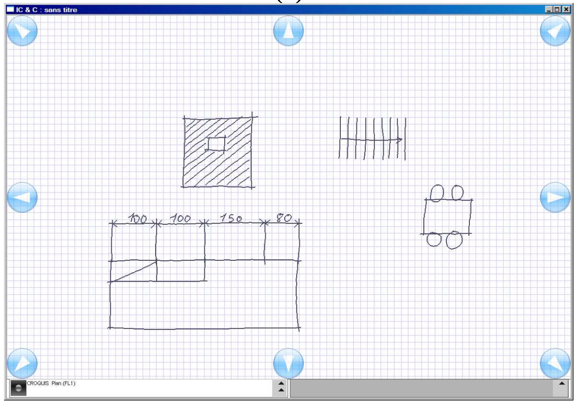
```
(STAIR  (:id 65542)
        ((BLOCK :type ARROW) (BLOCK :type HATCH))
        ((cross_p 0 1) (perpendicular_p 0 1)))
```

When the agent finds an object of interest on the data bus (*e.g.*, either `ARROW` or `HATCH`), it applies its set of rules to verify the object properties and check their possible relations with already locally stacked objects. If the object matches, it is stamped and pushed in a local stack. Relations with stacked objects are checked by testing predicates (*e.g.*, is the first stacked object crossing the second stacked object and are they orthogonal ?). Note that the predicates are specialized functions in the sense of object-oriented programming, that is, their implementation depends on the nature of their arguments. Once all the rules are matched, a higher-level object (*e.g.*, `STAIRS`) is sent to the data bus by the agent and objects are deleted from the stack. Otherwise, the objects are popped out of the stack and released to be interpreted by other agents.

Confidence scores are calculated on the low-level geometrical rules used in the composition of higher objects such as form, alignment, position, etc. These rules use threshold values to decide if the stroke can be included in the symbol. If the test fails, the agent stops its current composition and evaluates the possibility of building a symbol using previously collected strokes. Thus, the global confidence score of a high-level symbol can be seen as the minimum of the set of all values generated by low-level rules during the composition phase. The type of a newly composed object will never be called into question except in case of conflict with other agents. For example, the hatches agent uses the straight-stroke (property) and the parallel (relation) rules which are reasonably flexible but, if the received stroke fails for one of these rules, the composition is stopped. In the stairs symbol example, the agent receives a hatch and an arrow. The confidence score will be the minimum of the values returned by the predicate rules, without re-applying the composition rules of the hatch or the arrow objects.

Figure 2 shows some examples of sketched objects and their interpretation by the rule-based graphical squad. Hatches are recognized by composition of straight parallel lines, and each line can be composed of several aligned lines. Stairs are recognized by composition of an arrow crossing perpendicularly hatches. A table is recognized by composition a rectangle surrounded by small circles. A dimension line is found when straight parallel lines are perpendicular to multi-headed arrow lines, and numbers are located near the arrow lines and between the parallel lines. Many other rule-based graphical agents can be defined by providing a description of their rules.

(a)



(b)

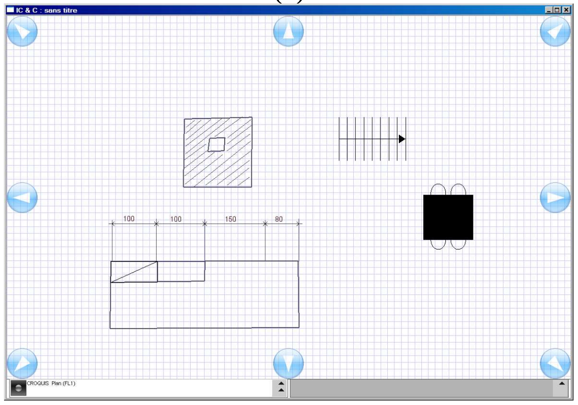


Fig. 2. Examples of (a) sketched objects and (b) their interpretation by the rule-based graphical squad.

It is important to stress that graphical objects are time-stamped. This chronological information allows solving many problems. For instance, because of the time persistance of the drawing, symbols may seem to overlap graphically yet they can actually be resolved by taking into account the time information.

C. Model-based Graphical Squad

As an alternative to rule-based graphical agents, we also propose model-based agents that happen to be very efficient for recognizing multi-stroke isolated symbols (*e.g.*, handwritten characters). Unlike the rule-based approach that describes the graphical objects to be recognized, the model-based approach is based on models of these objects that are estimated from training material.

Figure 3 shows a model-based graphical agent as defined in the approach that we adopt in this work [7]. When the agent is presented with strokes as defined by their sequence of point coordinates, they are first preprocessed. They are centered, normalized, resampled and finally, to reduce variations in the manner of drawing the symbols, the strokes are redirected and reordered in a canonical direction and order [8]. In the current system, there is no further feature-extraction step. Other similar approaches suggest to derive higher-level features from the raw point coordinates. Here, the preprocessed stokes are directly fed into a classification module that uses directly the

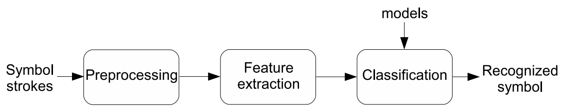


Fig. 3. General scheme of a model-based graphical agent (*e.g.*, for recognition of handwritten characters).

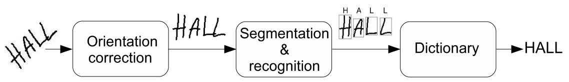


Fig. 4. Overall architecture of the legend and dimension recognition system

coordinates of stroke points to make its decision. The classifier is based on the k Nearest-Neighbor rule (k-NN). The strokes to be recognized are compared with a set of models. The k closest models are returned. The dissimilarity measure used is based on Dynamic Time Warping (DTW), a nonlinear curve matching method, between the sequence of coordinates for the test strokes and the reference strokes. The local cost between two stroke points is their squared Euclidean distance. The system is initialized with a set of models of symbols coming from multiple users. The system is thus initially generic and not adapted to a specific user. New samples of symbols coming from a particular user can also be added to this set of models to increase recognition performance for this particular user. Models that are very close to each other are merged by Learning Vector Quantization (LVQ) in order to limit their number.

To obtain a confidence measure for a recognized symbol, we estimate the probability $P(t|d,c)$ that the symbol has been correctly recognized given its distance d to the model of class c. To do so, for each symbol class c, we keep during the training procedure the distances obtained by a set of symbols belonging to this class in a sorted list L_t and the ones that do not belong to this class in a second list L_f. For a given distance d and a given symbol class c, we can compute an approximation to the probability:

$$P(t|d,c) \simeq \frac{k}{k + |L_f([d-l, d+l])|} \tag{1}$$

where $2l$ is the width of the smallest interval centered at d and containing a given number k of elements in L_t.

As an example, Figure 4 shows the various stages of the legend and dimension recognition process. First, since legends and dimensions may be drawn in any orientation, the strokes are rotated to make them horizontal. The strokes are then simultaneously segmented and recognized by the isolated symbol recognition system. Sequences of successive strokes are grouped. The group that generates the best recognition score is regarded as the correct segmentation. Finally, to improve recognition performance, the recognized word is checked against a dictionary. If the recognized word is not present, the closest word is chosen. Distances between strings are measured using the Levenshtein distance (LD) [9] to allow for substitutions, insertions and deletions of characters.

D. Model-based Vocal Squad

The interpretation of speech utterances relies on the Automatic Speech Recognition (ASR) process that is depicted

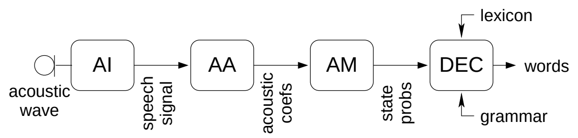


Fig. 5. A typical ASR system: microphone, audio interface (AI), acoustic analysis (AA), acoustic model (MA) and word decoder (DEC).

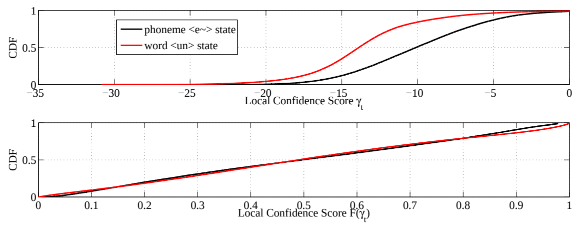


Fig. 6. Cumulative distribution function (CDF) function (PDF) of local confidence score γ (upper) and its normalized version $F(\gamma)$ (lower) for two states of different acoustic models, namely a state of French phoneme /ẽ/ (SAMPA notation) state in phoneme and a state of French word "un".

in Figure 5. It consists of four main blocks, which are all implemented with a homebrewed software [10]. First, the audio interface converts the acoustic wave that is measured by a microphone into a digital speech signal. Second, the acoustic analysis chops the speech signal into frames and computes for each frame a set of acoustic coefficients that capture the essential shape of the power spectrum. In this work, the acoustic coefficients are obtained via the ETSI standardized algorithm for distributed speech recognition [11]. Next, the acoustic coefficient vectors are fed into the acoustic model. Here, the acoustic model is based on the Multi Layer Perceptron (MLP) / Hidden Markov Models (HMM) paradigm [12]. Such model has to be trained beforehand on large speech databases. The outputs of the MLP estimate the *a posteriori* probabilities of all HMM states for the acoustic coefficient vectors. Finally, the word decoder searches the resulting probability lattice for the most likely word sequence. To do so, every word is represented by a HMM and authorized word sequences are defined in a context-free grammar. The search consists in finding the model sequence, thereby the word sequence, that best fits the acoustic observations within the grammar. It is implemented as a one-pass time-synchronous Viterbi algorithm with pruning [9].

The first two blocks of Figure 5 are part of the front-end layer of the system and deliver acoustic coefficient objects on the data bus, while the two last blocks belong to the interpretation layers. The vocal squad consists of several ASR agents that read the acoustic coefficient objects and infer word-level objects. Each agent is operated for a specific task, that is, it has its own lexicon, grammar and acoustic model, which has been trained on specific speech material. For example, there are ASR agents for recognizing specific words to command the interface, numbers to mark dimensions or letters to lay down spelled captions.

When an utterance is spoken, all the active ASR agents produce a recognition result according to their models, and a confidence score. Then, the negotiation module can select the best recognition by maximizing the confidence measure across the competing agents. This approach performs better than an *all-in-one* ASR system that uses a general-purpose acoustic model together with a complete lexicon and a global grammar encompassing all agent lexicons and grammars, respectively. The main reason is that the vocal squad approach allows using specific acoustic models. For some agent, it becomes possible to use word-based HMMs trained on specific speech material instead of composing subword HMMs (*e.g.*, context-independent phonemes). Besides, the set of active agents is dependent on the interface context what allows keeping the search space as small as possible by turning on only required agents, thereby reducing the risk of errors.

The confidence in the recognition result is naturally measured as the accumulated acoustic score along the decoded path [13], [14], that is,

$$\Gamma(W) = \Gamma(s_1^T) = \frac{1}{T}\sum_{t=1}^{T} I(s_t)\gamma_t, \tag{2}$$

$$\gamma_t = \log\left(P(s_t|o_t)/\max_{s_t} P(s_t|o_t)\right). \tag{3}$$

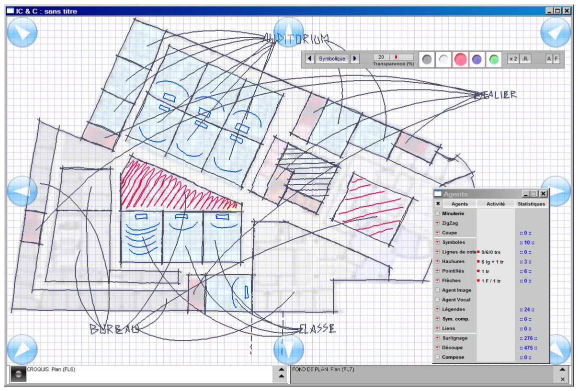
where $s_1^T = \{s_1, \ldots, s_T\}$ denotes the HMM state sequence estimated by the decoder and corresponding to the recognized sentence W, $P(s_t|o_t)$ stands for the *a posteriori* probability of being in state s_t given the observed acoustic vector o_t at t-th frame and is estimated by the MLP, and the indicator function $I(\cdot)$ is equal to 0 if the state s_t at the t-th frame is the silence state, and 1 otherwise.

In order to improve the selection procedure among the vocal squad, the confidence measures are further normalized. Define $F_i(\gamma)$ as the cumulative distribution function (CDF) of the local confidence score γ for the i-th state of the acoustic model. These functions can be estimated during the acoustic model training procedure where the state segmentation of the speech training database is known. The confidence measure is now computed as the average of the cumulative distribution function outputs $F_{s_t}(\gamma_t)$ instead of the local confidence scores γ_t,

$$\Gamma'(W) = \Gamma'(s_1^T) = \frac{1}{T}\sum_{t=1}^{T} I(s_t)F_{s_t}(\gamma_t). \tag{4}$$

Figure 6 shows the cumulative distribution function of the local confidence score γ and its normalized version $F(\gamma)$, respectively, for states of two different acoustic models. We clearly see that the functions are significantly different in the former case, while they tend to be identical and uniform in the latter case. Therefore, CDF outputs can be viewed as normalized local confidence scores that are better candidates for computing a more homogeneous and less sensitive confidence measure.

In this work, phoneme-based and word-based acoustic models are trained on the BREF80 database [15] and on the EUROM1 [16], BDSONS [17] and self-collected databases, respectively. The former models allow designing ASR agents for recognizing any word defined by its phonetic transcription. The latter models are dedicated to ASR agents for recognizing numbers or alphanumeric words. All ASR agents are speaker-independent.



Agents	Activité	Statistiques
Minuterie		
ZigZag		
Coupe		:: 0 ::
Symboles		:: 10 ::
Lignes de cote	0/6/0 trs	:: 0 ::
Hachures	6 lg + 1 tr	:: 3 ::
Pointillés	1 tr	:: 6 ::
Flèches	1 F / 1 tr	:: 0 ::
Agent Image		
Agent Vocal		
Légendes		:: 24 ::
Sym. comp.		:: 0 ::
Liens		:: 0 ::
Surlignage		:: 276 ::
Découpe		:: 475 ::
Compose		:: 0 ::

Fig. 7. Screenshot of the sketch interpretation interface.

III. Discussions

Eventually, the system for multimodal sketch interpretation that is described in the previous sections has been embedded in a graphical user interface that is shown in Figure 7. The design and the evaluation of this interface have been performed in the framework of the Activity Theory (AT) [18]. This user-centered development approach consists of the following steps. First, a field of activity and applications should be defined. Then, the actual activity of users of the field is analyzed and initial design requirements are provided to the interface developers. Next, the developments are performed and evaluated against user objectives. Feedbacks are returned to the developers for possible improvements. The last two steps are repeated until the interface reaches sufficient performances.

Here, we are primarily interested in architectural design and potential applications are design sketching (*e.g.*, original designing of a building), scale drawing (*e.g.*, detail drawing of a worksite including real dimension) and plan retouching (*e.g.*, alterations of existing plan on worksite).

The analysis of user requirements leaded to the following conclusions. The system should be able to recognize basic graphical objects like lines, hatches, circles, etc, as well as as more complex objects like stairs, doors, etc. Besides, it should be able to identify dimension lines and captions. Other multimodal approaches to sketch interpretation [3] generally use the different modalities in a redundant way instead of a complementary way, that is, they assume that the user says what he is drawing. We rather observed that the user prefer dedicating each modality to a specific task. For instance, the user typically pronounces dimension values but writes text captions, and pronounces some interface commands only if they requires more than a few pen/mouse clicks.

All these considerations were taken into account to develop the interface and evaluation in real conditions was performed. They showed that every squad has satisfying performances and the interface allows already functional interpretation of sketches.

IV. Conclusions and Future Work

In this paper, we presented a multimodal sketch interpretation system that relies on a multi-agent architecture. The proposed system allows various input modalities and uses a competition mechanimx between squads of dedicated agents to interpret them. More especially, we considered two modalities, namely graphical and vocal. The graphical inputs are interpreted by either rule-based agents or model-based agents, while the spoken inputs are interpreted by model-based vocal agents.

Main directions for future work include improvement of interpretation robustness (*e.g.*, by specializing the model-based agents to user drawing style and speech) and reinforcement of the negotiation mechanism by defining more meaning-full confidence scores.

Acknowledgment

This work is supported by the Direction Générale des Technologies, de la Recherche et de l'Energie (DGTRE) of the région Wallonne (RW) in Belgium, under project RW-WIST-IC&C.

References

[1] G. Costagliola, V. Deufemia, G. Polese, and M. Risi, "A Parsing Technique for Sketch Recognition Systems", *Proc. of IEEE Symposium on Visual Languages – Human Centric Computing*, Roma, Italy, Sep. 2004.

[2] C. Alvarado and R. Davis, "Dynamically Constructed Bayes Nets for Multi-Domain Sketch", *Proc. of Internaltional Joint Conferences on Artificial Intelligence*, Edinburgh, Scotland, Jul. 2005.

[3] L. Wu, S. L. Oviatt, and P. R. Cohen, "Multimodal Integration – a Statistical View", *IEEE Trans. on Multimedia*, vol. 1, no. 4, pp. 334–341, Dec. 1999.

[4] A. Adler and R. Davis, "Speech and Sketching for Multimodal Design", *Proc. of International Conference on Intelligent User Interface*, Madeira, Portugal, Jan. 2004.

[5] P. Leclercq, "Interpretative Tool for Architectural Sketches", *Proc. International Conference on Visual and Spatial Reasoning in Design: Computational and Cognitive Approaches*, Cambridge, USA, Jun. 1999.

[6] R. Juchmes, P. Leclercq, and S. Azar, "A Freehand-Sketch Environment for Architectural Design Supported by a Multi-Agent System", *Computers and Graphics*, vol. 9, no. 6, pp. 905–915, Dec. 2005.

[7] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas, "On-line Adaptation in Recognition of Handwritten Alphanumeric Characters", *Proc. of International Conference on Document Analysis and Recognition*, Bangalore, India, Sep. 1999.

[8] N. Matsakis, "Recognition of Handwritten Mathematical Expressions", *M. Sc. thesis*, MIT, May 1999.

[9] X. Huang, A. Acero, and H.-W. Hon, "Spoken Language Processing: A Guide to Theory, Algorithm, and System Development", *Prentice Hall*, pp. 522–525, 2001.

[10] J.-M. Boite, L. Couvreur, S. Dupont and C. Ris, "Speech Training and Recognition Unified Tool (STRUT)", `http://tcts.fpms.ac.be/asr`.

[11] ETSI ES 202050, "Speech Processing, Transmission and Quality Aspects (STQ), Distributed Speech Recognition, Advanced Front-End Feature Extraction Algorithm, Compression Algorithm", *ETSI Standard*, Jul. 2002.

[12] H. Bourlard and N. Morgan, "Connectionist Speech Recognition – A Hybrid Approach", *Kluwer Academic Publisher*, 1994.

[13] G. Williams and S. Renals, "Confidence Measures from Local Posterior Probability Estimates", *Computer Speech and Language*, vol. 13, no. 4, pp. 395–411, Oct. 1999.

[14] E. Mengusoglu and C. Ris, "Use of Acoustic Prior Information for Confidence Measure in ASR Applications", *Proc. EUROSPEECH*, pp. 2557–2560, Aalbord, Denmark, Sep. 2001.

[15] L. F. Lamel, J. L. Gauvain and M. Eskenazi, "BREF, a Large Vocabulary Spoken Corpus for French", *Proc. EUROSPEECH*, pp. 505–508, Geneva, Italy, Sep. 1991.

[16] J. Zeiliger, J.-F. Serignat, D. Autesserre and J.-M. Dolmazon, "EUROM1: une Base de Donnees Parole Multilingue", *Proc. JEP*, pp. 303–306, Bruxelles, Belgique, May. 1992.

[17] R. Carre, R. Descout, M. Eskenazi, J. Mariani and M. Rossi, "The French Language Database: Defining, Planning and Recording a Large Database", *Proc. ICASSP*, pp. 324–327, San Diego, California, Mar. 1984.

[18] C. Boulanger, F. Descortis and S. Safin, "Portable Tool for Finalizing Freehand Drawings: Activity Analysis and Design Requirements", *Proc. Annual Conference of the European Association of Cognitive Ergonomics*, Chania, Crete, Sep. 2005.