Autonomous Agents as Competent Assistants: Better Interpretation of Pen Gestures

K. A. Mohamed, Th. Ottmann

Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Georges-Kohler-Allee, 79110 Freiburg, Germany
{khaireel, ottmann}@informatik.uni-freiburg.de

Abstract. Interaction between users and synthetic hardware systems have provided opportunities to utilise an important sensing modality through the use of a digital pen (stylus). Ambient learning environments insist on recognising and reacting to users' input sketches and gestures for expediting the responses needed to improve user-interface (UI) supports for more natural communications. This paper focuses on the techniques and methodologies used in ciphering information obtained from *digital ink* compositions that gave way to the conceptions of elaborate and intelligent programs. We present our classification of sketch-objects for identification purposes while maintaining their native attributes, and discuss our approaches for interpreting them as valid *intended* gestures through autonomous agents.

Keywords: Digital pen gestures, recognition, interpretation, strokes
CR Classification: G.1.10, H.3.3, I.7.5

1 Introduction

In a bid to keep up with contemporary technology, we are seeping from conventional classrooms to embrace hi-tech and modern teaching environments that come equipped with elaborate presentation devices. Albeit the intricacies of these devices to ensure a better and more comfortable atmosphere for its users, achieving that same "natural" user-interface (UI) feeling we get in conventional classrooms actually requires more background work – which are needed to integrate between the various presentation mediums for a smooth, easy, and "natural" delivery of materials.

Efforts to merge these multiple mediums resulted in products such as Panasonic's Interactive Board [14], Numonics' Digital Presentation Appliance^{TM} [3], and Wedgwood Group's Hitachi-Cambridge interactive whiteboards [8]. These digital boards allow presenters to interact with them as they would with typical PCs or laptop computers, but through the use of a stylus for annotating handwriting and sketches, tapping program menus, dragging applications, etc. More often than not, dedicated toolkits are developed to help with these interactions, and it seems a trend that the toolkits interpret their users' writings and sketches

for further manipulations [9, 11, 16, 15, 13]; storing and retrieving them as information sources and attaching recognisability-routines for specialised purposes.

Studies indicate that users' experiences with the interaction technology between the styluses and the digital boards must not be overly obtrusive and awkward. This interaction becomes a burden and an obstacle for a natural UI environment, if it draws too much attention to the technology or imposes a high cognitive load on the users than the tasks at hand [4]. But using only the stylus for sketching inputs to the UI is not enough to satisfy demanding scenarios – such as to quickly change a pre-edited document, or calling up menus for issuing commands to the UI while delivering a presentation. Inevitably then, a unique way to achieve this anomalously, without pressing the corresponding key on a standard keyboard, is to stroke a *gesture*. Such inputs must be received and processed accordingly, depending on the specificity of the user's domain, by some sort of *agents* running in the background; communicating between themselves through predefined protocols.

Competent ubiquitous agents utilise their task specific 'expertise' to guide users through complex UIs for better *board-and-user* interactions. These assistants do not attempt to be experts at everything, but rather, they are familiar with all the various processes that make up their tasks [7]. This would make it possible for users to personalise these assistants by pre-programming their own special gestures to coincide with multiple-menu interactions. As such, the agents must be able to understand the input sketches and gestures so as to pave the way for the development of adaptive goals necessary for contriving and appreciating new objectives.

In this paper, we first review the mechanics behind and the processes involved in gesture recognition. We then report the notion of our conception of autonomous agents as a solution towards building a basis for our *gesture protocols*. Embedded into these agents are the conventional methodologies adapted from Rubine's algorithm [17] for extracting features from *ink*-objects. This is to supplement the heuristics and calculations of the algorithm as part of our recogniser engine. As our experiments form the foundation for our future work on "protocolling GestureObjects", such that we need to include some elements of dynamism when recognising gestures, supplementary techniques to extract *invariant* features are also augmented.

These experiments are described and detailed in the Results section. A section then follows discussing the results to complement our claims that the implementation of our agents are not only successful and efficient, but also accurate in giving correct interpretations of *ink*-objects as intended recognised GestureObjects, suitable for a dynamic and natural UI environment.

2 Gesticulations and the Gesture Continuum

The power of gestures lies in their ability to convey expressive and meaningful strokes as commands to the system without having to leave the vicinity of the digital screen. Cutler and Turk [4], Barrientos and Canny [1], and Wexel-

brat [21] noted in their reports the following functional roles of human gestures; to communicate meaningful information (semiotic), to manipulate the environment (ergotic), and to discover the environment through substantial experience (epistemic).

Cutler and Turk further expanded their definitions to include *communication tasks* when interpreting the gestural inputs from the stylus. These tasks specify the commands and parameters for:

- Navigating through 2D space;
- Specifying items of interest;
- Manipulating objects in the sketching environment;
- Changing object values; and
- Issuing task specific commands.

Kendon [12] added that an important part of 'kinetics' research shows that gesture phrases can be organized in relation to speech phrases. We can parallel his arguments and reasonings to relatively coincide with pen gestures (as natural human gestures) and the instantiated sketch-objects (as dictionised speech contents). He also stated that there is a consistent patterning in how gesture phrases are formed in relation to the phrases of speech – just as, in a continuous discourse, speakers group tone units into higher order groupings resembling a hierarchy, so gesture phrases may be similarly organized.

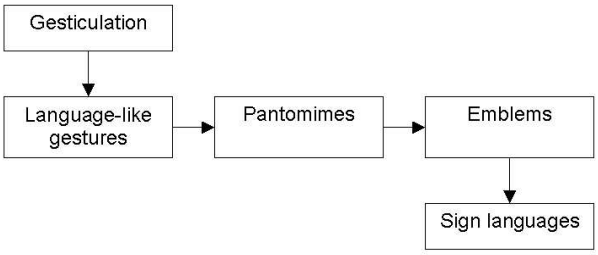

Fig. 1. Kendon's description of the gesture continuum defining five different kinds of gestures.

Gestures that are put together to form phrases of bodily actions have the characteristics that permit them to be 'recognized' as components of willing communicative action (Figure 1). There are five kinds of gestures that make up the gesture continuum:

- *Gesticulation* – spontaneous movements of the hands and arms that accompany speech.
- *Language-like gestures* – gesticulation that is integrated into a spoken utterance, replacing a particular spoken word or phrase.

- *Pantomimes* – gestures that depicts objects or actions, with or without an accompanying speech.
- *Emblems* – familiar gestures accepted as a standard.
- *Sign languages* – referring to the complete linguistic system, such as the American Sign Language.

As the list progresses from top-left to bottom-right in Figure 1, association with speech declines, language properties increases, spontaneity decreases, and social regulation increases. This gesture continuum clearly states a useful background support for deriving systems that are able to receive continuous pen gestures as inputs, along with a string of referenced objects.

3 Autonomous Agents

Our approach uses primary agents that perceive the sketching environment and act accordingly to decipher the digital *ink*-objects as valid (or invalid) gestures (*GestureObject*s). These agents are observed and studied to determine their viabilities as background assistants through their object-interpretation capabilities. Their rationality and intelligence lies on them giving correct inferences, with an attached value of confidence when defining *ink*-objects as recognised GestureObjects.

3.1 Interpreting with Agents

Agents are best used in environments where there exist ample opportunities for full expositions of knowledge representation and reasoning [18]. A knowledge-based agent is described at three levels as follows.

- *Epistemological level* or knowledge level, which is the most abstract level, outlines the agent by saying what it knows. For example, an autocratic agent might be said to know that there is a strong correlation between recognised gestures a and b, from the newly scribed *ink*-object c.
- *Logical level* is the level at which the knowledge is encoded into sentences. For example, the agent might be described as having the logical sentence $\mathbf{StrongCorrelation}(a, b, c)$.
- *Implementation level* is the level that runs on the agent's architecture, which is relative to the system that the agent has been designed to operate in.

Our focus of implementation requires the agents to begin their collaborative interpretation from the input *ink*-data received through the UI (*SketchObjects*), that a possible GestureObject has been scribed. Depending on the hardware system [19], or the human factors associated with this scribing [20], these SketchObjects may appear noisy and render it difficult for any algorithms to efficiently give good diagnoses. Hence the need to filter out the noises and to rationalise the SketchObjects appropriately.

Once a SketchObject is rationalised and becomes observable, our agents can then begin their processes of extracting features, unique only to a special class of recognised GestureObjects. Their analyses will refer to their epistemological and logical levels, which are instilled in their recogniser algorithms and knowledge engines.

3.2 Concerted Architecture

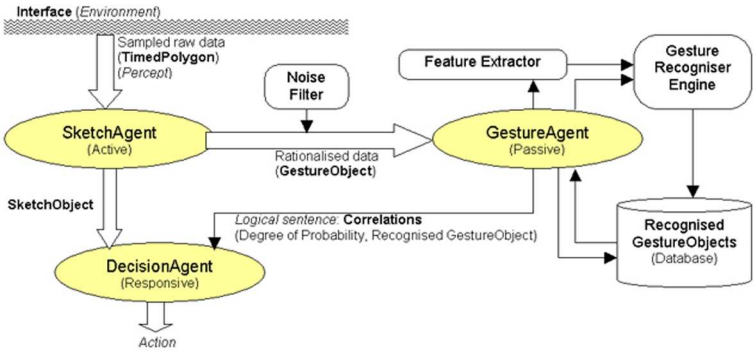

Fig. 2. Deploying autonomous agents into the system's architecture for percepting, interpreting, and deciding whether a SketchObject is strong enough to be classified as a GestureObject.

We structured three specialised agents to receive the raw SketchObjects, perceive the clarity of their information, and act accordingly by invoking necessary methods for identifying GestureObjects. The *SketchAgent* is an active listener, while the *GestureAgent* is a passive processor. The latter determines and associates a degree of probability to each rationalised SketchObject as it is compared to a set of recognised GestureObjects from the agent's knowledge database. This information is then passed to a *DecisionAgent*, which accepts or rejects the logical hypothesis.

Figure 2 illustrates the combined architecture of the three agents. This structure is used for both the "Training"and "Application" phases when interacting with users through the UI. These phases have different goals that complement each other; the former builds the database, while the latter retrieves and reacts from the information contained.

3.3 The Feature Extractor

In generative linguistics, features are defined as any of various abstract entities that combine to specify the underlying phonological, morphological, semantic, and syntactic properties of linguistic forms and act as targets of linguistic rules

and operations. Similarly, the features extracted from newly scribed GestureObjects relay useful information that can be used as parameters which uniquely identify a particular *RecognisedGesture*.

These features tell the angular information, velocity readings, and measurement of lengths. There should also be enough features to provide differentiation between all gestures that might reasonably be expected.

Standard Features Part of our extraction routine includes Rubine's [17] set of 13 features, chosen such that each feature is incrementally computable in constant time per input point. This allows arbitrarily large gestures to be handled as efficiently as small ones. Each standard feature is made in such a way that it is meaningful enough to be used in gesture semantics as well as for recognition.

Advanced Features UC Berkeley's GUIR [13] defined a set of advanced features and associated them as part of the "Human Perception of Gesture Similarity". Their application, *Quill*, tries to predict when it is that people will perceive gestures as very similar, and this prediction is based on some specific geometrical features of a GestureObject. We used these features for similarity prediction, rather than for recognition.

Invariant Features As long as the elementary shapes of the input GestureObjects are relatively similar to the ones contained in their RecognisedGestures database, our agents should be able to identify these with strong degrees of confidence. We ensured that we can interpret intended gestures correctly, irrespective of their sizes, positions, and their disorientation (rather than orientations when gesturing on a large screen) by exploiting moments of the 0^{th}, 1^{st}, 2^{nd}, and 3^{rd} orders of the input SketchObjects – the first three of which corresponds respectively to the area, mean, and variance of a 2-D binary image. These manipulations of the moments are the accorded for the length, median, and variance features of a GestureObject for use in Hu's [10] equations of invariant moments.

3.4 The Gesture Agent

Since Rubine's algorithm makes it necessary to have good example sets (obtained during Training sessions) for making reliable inferences when in application phases, our GestureAgent must be intelligent enough to set its goals accordingly when placed in the two different scenarios. Each goal invokes a different part of the algorithm; it either builds up a good set of example libraries, or refers to and retrieves relevant information from these libraries to perform the necessary calculations.

Figure 3 illustrates the internal works of our GestureAgent in its processing of information and communicating with its environment. The GestureAgent's goal is determined at start-up time by the system, and can be deployed in both Training and Application modes.

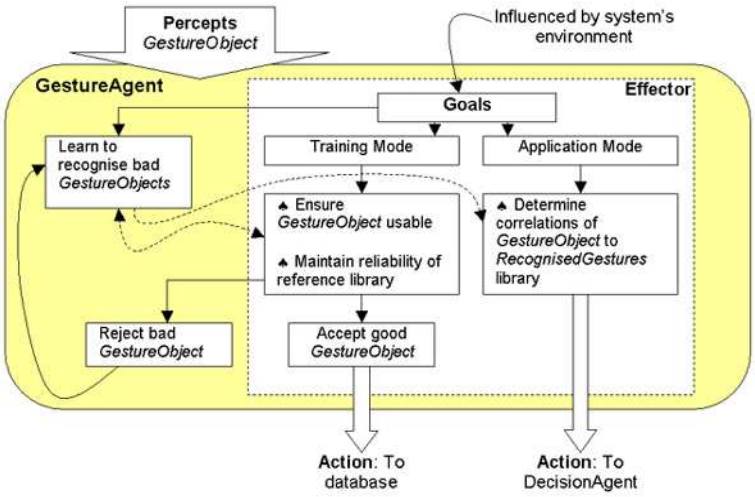

Fig. 3. GestureAgent interacting with its environment.

Referring to our discussions earlier, our implementation of Rubine's algorithm makes provisions for taking necessary precautions to rationalise feature values before performing any computations on them. But there are some unpredictable situations when even this precaution is overlooked, and is only discovered at the end of the entire evaluation process – a rather costly rejection method. As such, our GestureAgent is programmed to learn the feature values that cause its calculations to break down, and we built mechanisms to intercept bad GestureObjects before processing them in both Training and Application modes.

4 Results

We tailored our experiments to test the robustness of the agents in receiving and reacting to the information provided from the UI. We also tested the correctness of the implementation of Rubine's algorithm, within the GestureAgent, in making correct inferences and deciphering the uniqueness between the various classes of the recognised GestureObjects.

Altogether, we structured nine sets of gestures for the experiment as illustrated in Figure 4; the first five of which are straightforward and anomalous gestures – 4(a), (b), (c), (d) and (g). The other four are made to evaluate the strength of the recognition algorithm as it is put to interpret gestures that are ambiguous in their orientations and directions of which they are scribed.

Here, we mention the graphic details of our prototype's "ubiquitous" canvas-interface. Through it, users are able to view programs running in the background as they continue to use their stylus to write naturally on the semi-transparent canvas. With our application intended for use in "teaching" domains, this would

serve as a useful property that, for example, does not block an open PDF document or a Powerpoint presentation.

To relate how well the recogniser engine is being managed by the GestureAgent to form reliable logical hypotheses from its naive epistemological levels, Table 1 shows the average top three identified GestureObjects with respect to each intended gesture class. The table indicates that all *firstly*-identified GestureObjects in the logical sentences for all RecognisedGesture classes are indeed that of the actual intended gestures of the users.

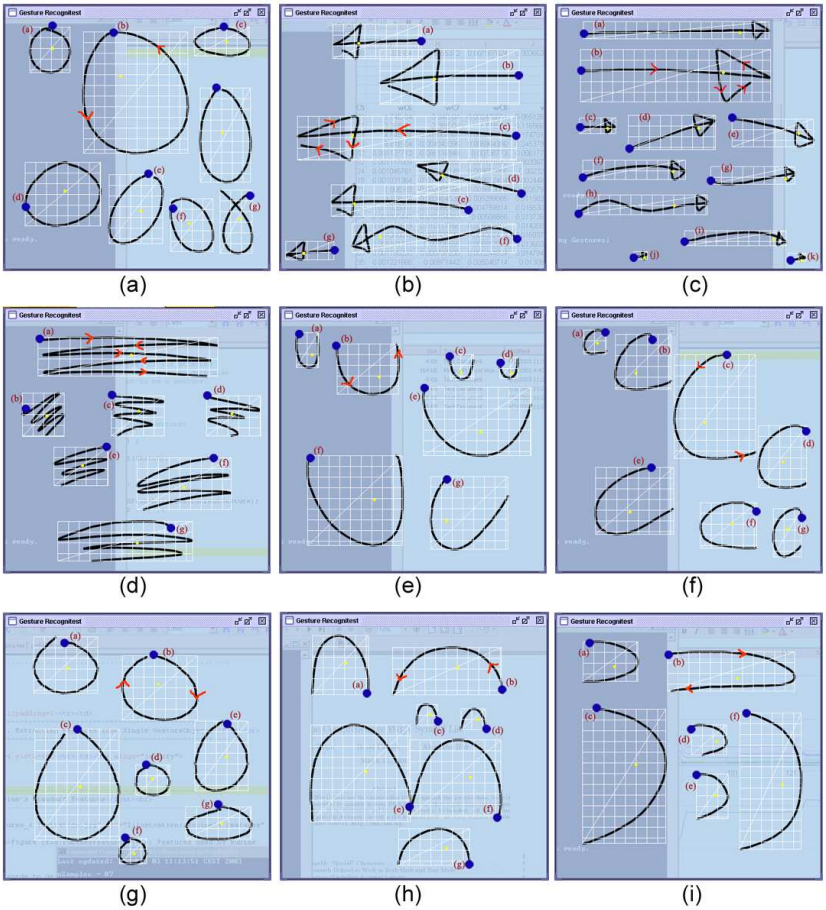

Fig. 4. Gestures identified as (a)Circle-anticlockwise, (b)Left-arrow, (c)Right-arrow, (d)Zig-zag, (e)Union, (f)The-C, (g)Circle-clockwise, (h)Inverted-union, and (i)Inverted-C.

No.	Intended Gesture	Average top three identified
(a)	Circle-anticlockwise	Circle-anticlockwise, The-C, Union
(b)	Left-arrow	Left-arrow, Zig-zag, Right-arrow
(c)	Right-arrow	Right-arrow, Left-arrow, Zig-zag
(d)	Zig-zag	Zig-zag, Right-arrow, The-C
(e)	Union	Union, Circle-anticlockwise, The-C
(f)	The-C	The-C, Inverted-union, Union
(g)	Circle-clockwise	Circle-clockwise, Circle-anticlockwise, The-C
(h)	Inverted-union	Inverted-union, Circle-anticlockwise, The-C
(i)	Inverted-C	Inverted-C, Inverted-union, The-C

Table 1. Top three average correlations for the intended gesture classes.

5 Discussion of Results

The experiments were conducted in a controlled environment with the main objectives set to assess the stability of the autonomous agents running in the background, as well as to ensure that the intended gestures scribed by various users are correctly interpreted. Here, badly scribed gestures or gestures scribed that are not part of our library of RecognisedGestures are not taken into account.

We measure the performance of our agents through their efficiencies, as a whole, and their time complexity analysis via the big '**O**' notation, and are elaborated in the following sub-sections.

5.1 Correctness and Completeness

The results we recorded and tabulated in the previous sections all show that the strongest probability value for each input gesture, correspond correctly to that of the user's actual intended gesture.

The algorithm (through our agents) is still able to recognise the anomalous gesture classes with strong confidence, and the ambiguous gesture classes with a comfortable margin that separates the next highest interpreted RecognisedGesture. The diversity in the styles possessed by seven different people when scribing our experimental gestures also resulted in the algorithm deciphering correct and *near*-correct (those interpretations that came in second in the list of probability strengths) judgements of the intended gestures.

As each information set is communicated between the SketchAgent and the GestureAgent, their epistemological and logical levels are purportedly observed to be correct and thus, complete. That is, it is ensured the recognition process is seamlessly integrated into the system. These epistemological levels refer to the agents knowing that there exist a correlation of likeliness between the newly scribed GestureObject to its recognised library contents, and that the formation of logical sentences generated for each input GestureObject as the list of probability strengths, passed on to the DecisionAgent to accept or reject the hypotheses, is valid.

5.2 Complexity Analysis

Generally analysing the time complexity of the whole agent-system implementation is done in three distinct steps; feature extraction (T_f), training (T_t), and identifying (T_i).

The features in a GestureObject are extracted sequentially, and some of these sequences depend on the number of points sampled, n, in the GestureObject. 18 features are extracted in our experiments, all of which involve straightforward calculations ($\mathbf{O}(1)$). Hence, it takes $T_f(n) = 18n\mathbf{O}(1)$ amount of time to extract the features of an input GestureObject.

Again, the number of sampled points in any particular GestureObject play an important role – this time, in determining the complexity of the training time. The sample estimate of the mean feature vector per class gives a complexity of $E\mathbf{O}(n)$, where E is the number of training examples for the class. We implemented this to build the common covariance matrix M simultaneously. Inverting M costs $\frac{1}{2} \times \mathbf{O}(nE^4)$, as proven by Blum *et. al.* [2]. Solving the linear equations are easy after the inversion of M, and gives a complexity related to the number of features extracted. Thus, it takes $T_t(n) = T_f + E\mathbf{O}(n) + \frac{1}{2} \times \mathbf{O}(nE4)$ amount of time to train one gesture example.

The identification process, at the moment, looks at all c RecognisedGestures in the system's library to calculate the evaluated-vector-values ($c \times \mathbf{O}(n)$) and the probability values ($c \times \mathbf{O}(n)$) for comparisons. This gives a time complexity of $T_i(n) = T_f + 2c\mathbf{O}(n)$.

6 Flexible Domain-Specific Applications

An important thing to note in the analyses from the previous section is that we are able to identify gestures almost instantaneously. The collaborative routines by the three agents in our architecture (Figure 2) passes *ink* information flawlessly between themselves in their attempt to convert the input gesticulations into understandable emblems of RecognisedGestures. This is done while at the same time maintaining the original *ink* information – should there be a need for other external program-routines requiring the raw SketchObject data.

We stated earlier that our development of competent assistants, through autonomous agents, is to bridge the lack of 'naturality' of the interaction technology between the styluses and the digital boards. We are building on to our task-specific agents to accomplish this in a manner that is ubiquitous to the users, while taking advantage of the speed of the recognition algorithm. By quickly processing input SketchObjects from the UI, the agents are expected to immediately identify selected *ink* data on screen as parameters, and react to *command* gestures by despatching RecognisedGestures as commands to invoke program menus or to run predefined applications. Figure 5 gives an example of this natural interaction of simple gestures.

The library of RecognisedGestures, which is being constantly referred to by the GestureAgent (Figure 3), will depend on how much familiarity it needs to

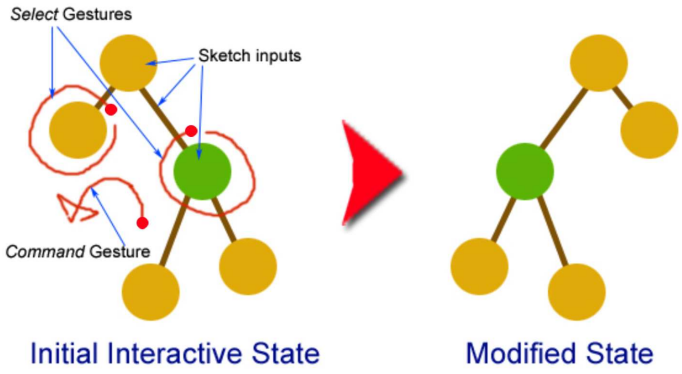

Fig. 5. Domain-specific application example: Rotating leaves in trees.

know in order to understand the various domain-specific processes that make up its task. This flexible tailoring should be seen as an advantage; tasks in specific domains should be well-modeled and clearly defined before being developed into reference libraries, the performance of the agents will not be hampered by the unnecessary search times to match too many RecognisedGestures, and that domain-personalisation can be achieved quite easily.

Domains that may affect this path of research work is listed below.

- *Education*: Intelligent classrooms.
- *Business*: Trends for effective meetings on networks.
- *Gaming*: Touch-screen, gesture-enabled joysticks.
- *Disabled*: Gesture-adaptability.
- *Medicine*: Gesture-aided surgeries.

7 Conclusion

Equally essential in a learning environment, apart from the recognition and interpretation of gestures, is the management of SketchObjects and GestureObjects. For in order to manipulate them capaciously, the system will need a basis for it to successfully match the diagrammatic similarity between the sketched-shape and its spatial relations [5, 6]. As the entire system stabilises after many uses, it would have amassed a collection of useful diagram references within its databases, that may contain helpful information such as its holding properties, transformation abilities, and its compatibility with other diagrams.

Since the contents in the database would already be sorted and tagged appropriately, the retrieval of these information for referencing may be indexed on its conceptual design features, on functions and purposes, on visual similarity, or shape – depending on the domain specificity of the system's implementation.

The three agents employed for this experiment will form the principal foundation of the intelligence of our system in our attempt to put together temporal

and spatial information of the *ink*-objects for recognising more complex formations of sketches and gestures. The next step in our "protocolling" gestures (and sketches) will involve additional agents to help in both the learning and recognising of special perspectives and a different paradigm to achieve our targeted goal of gesture programming.

Acknowledgement

This research is funded by the Deutschen Forschungsgemeinschaft (DFG) - The German Research Foundation - with a project intended for the "Algorithmen und Datenstrukturen für ausgewählte diskrete Probleme (DFG-Projekt Ot64/8-3)".

References

1. Barrientos, F. A., and Canny, J. F. Cursive: Controlling expressive avatar gesture using pen gesture. In *Proceedings of the 4th International Conference on Collaborative Virtual Environments* (2002), ACM Press, pp. 113–119.
2. Blum, L., Cucker, F., Schub, M., and Smale, S. *Complexity and Real Computation*, 2nd ed. Springer-Verlag, New York, Inc., 1998.
3. Corporation, N. The DPA (Digital Presentation Appliance) [online], Available: http://www.numonics.com/ipmindex.html [Accessed 22 June 2003]. *Numonics Corporation* (2002).
4. Cutler, R., and Turk, M. View-based interpretation of real-time optical flow for gesture recognition. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition* (1998), IEEE, pp. 416 –421.
5. Do, E. Y.-L. What's in a diagram (that a computer should understand)? *Computer Aided Architectural Design Futures '95* (1995), 469–482.
6. Do, E. Y.-L., and Gross, M. D. Drawing analogies - Supporting creative architectural design with visual references. In *3rd International Conference on Computational Model of Creative Design* (1995), M.-L. Maher and J. Gero, Eds., pp. 37–58.
7. Franklin, D., and Hammond, K. The intelligent classroom: Providing competent assistant. In *Proceedings of the Fifth International Conference on Autonomous Agents* (May 2001), ACM Press, pp. 161 – 168.
8. Group, W. Hitachi interactive whiteboards - front projection [online], Available: http://www.interactive-whiteboards.co.uk/hitachi_whiteboards.htm [Accessed 26 June 2003]. *Wedgewood Group* (December 2002).
9. Hong, J. I., and Landay, J. A. Satin: a toolkit for informal ink-based applications. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology* (2000), ACM Press, pp. 63–72.
10. Hu, M. K. Visual pattern recognition by moment invariants. *IRE Transitions on Information Theory 8* (1962), 179–187. As cited in Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison–Wesley Pub Co, pp 516, 1992.
11. Igarashi, T., Matsuoka, S., and Tanaka, H. Teddy: A sketching interface for 3d freeform design. *Proceedings of SIGGRAPH* (1999).

12. Kendon, A. An agenda for gesture studies [online], Available: http://www.univie.ac.at/wissenschaftstheorie/srb/srb/gesture.html [Accessed 19 January 2003]. *Semiotic Review of Books* (2001).
13. Lam, Z. GUIR projects [online], Available: http://guir.cs.berkeley.edu/projects/ [Accessed 14 April 2003]. *Group for User Interface Research* (2003).
14. Ltd, F. M. S. Panasonic UK - KXBP800 Interactive Board [online], Available: http://www.panasonic.co.uk/product/interactiveboard/kxbp800.htm [Accessed 27 June 2003]. *Panasonic UK Ltd* (2002).
15. Moran, T. P., Chiu, P., and van Melle, W. Pen-based interaction techniques for organizing material on an electronic whiteboard. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology* (1997), ACM Press, pp. 45–54.
16. Mynatt, E. D., Igarashi, T., Edwards, W. K., and LaMarca, A. Flatland: New dimensions in office whiteboards. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1999), ACM Press, pp. 346–353.
17. Rubine, D. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques* (1991), ACM Press, pp. 329–337.
18. Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, USA, 1995.
19. Smyth, K. The digital pen: Mapping and data aquisition. In *The Annual Conference and Exhibition: A GEOSPATIAL ODYSSEY, GITA Conference* (April 1998), GIS Development, pp. 261–268.
20. Subrahmonia, J., and Zimmerman, T. Pen computing: Challenges and applications. *Proceedings of the 15th International Conference on Pattern Recognition 2*, 1 (September 2000), 60–66.
21. Wexelblat, A. Gesture at the user interface: A CHI '95 workshop. *ACM SIGCHI Bulletin 28*, 2 (1996), 22–26.