SVEX: A KNOWLEDGE-BASED TOOL FOR IMAGE SEGMENTATION

Hernández-Sosa, D., Cabrera-Gámez, J., Falcón-Martel, A., Hernández-Tejera, M.
Departamento de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria
Campus Universitario de Tafira, 35017 Las Palmas, Spain
daniel@gias720.dis.ulpgc.es

ABSTRACT

SVEX is a multilevel knowledge-based tool for developing applications in image segmentation. Both numerical and symbolic computations take place at each level, being the transition between these two domains defined by the computational structure itself. SVEX incorporates evidence combination [Stra-92] and uncertainty control mechanisms. SVEX is programmed by means of a specific purpose declarative language based on a reduced set of objects. All the knowledge involved in the solution of a given segmentation problem is made explicit due to the declarative nature of the programming language [Nazi-84]. The results obtained by the application of SVEX in the segmentation of a set of outdoor images set are also shown in this article.

1. LEVELS OF ORGANIZATION AND SYSTEM ARCHITECTURE

SVEX is a multilevel knowledge-based tool for developing applications in image segmentation programmed by means of a specific purpose declarative language. SVEX is made up of two levels, characterized by the nature of the "information grain" that is numerically and symbolically described within each of them. These units are the pixels at the lower level and the segments (aggregations of pixels) at the upper level. The control of both levels is goal-oriented, so that computational processes are started by the reception of a computation request from an upper level. Accordingly, data flows in two directions across each level. In a first top-down processing, upper level requests are received and transformed into the corresponding commands for satisfying them. In the opposite bottom-up direction, each level processes data coming from lower level, transforms them and sends results to the upper level. For the processing of both data flows, each level has a TD unit (for control, planning and decoding) and a BU unit (for diagnosis, codification or abstraction).

Each level can be seen as a machine (BU-TD module) that interacts with a certain "virtual world." This interaction is performed with the aid of real or "virtual" sensors and actuators, depending on this world being the physical world or the world built by the lower level. This organization permits, on one hand, to provide a clearly defined and scalable computational structure (in terms of the complexity of the problem to be solved) and, on the other, to keep the same organization inside each level.

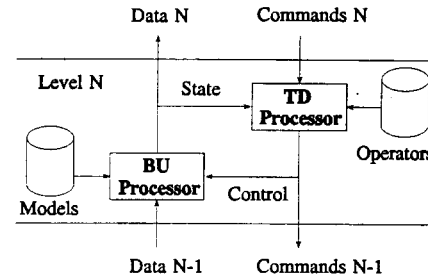

Figure 1. BU-TD Module.

2. DESCRIPTION OF THE LEVELS

2.1. Pixel Processor

Starting from the source image, the first level of SVEX, the Pixel Processor [Mend-94], produces diagnostic maps in terms of membership degrees of each pixel to certain symbolic classes. The BU unit receives images as input data from camera sensors and produces symbolic diagnostic images as output data. The TD unit receives requests for computing pixel diagnoses and transforms these requests into execution orders addressed to the BU unit and control commands directed to the image acquisition controllers.

At the pixel level, the numerical representation consists of feature maps (gradient, color, . . .) obtained using image processing algorithms. The symbolic representation handles symbolic images or maps ("HighGradientPixel", "GreenPixel", . . .) [Wils-88] generated from features using fuzzy symbolization

0-7803-2431-5/95 $4.00

processes that include evidence combination [Stra-92] and uncertainty control.

The programming language objects used at this level are shown in Figure 2. The object *Picture* identifies the input image. *Feature* objects are image features obtained from the image or from other features by the application of image analysis procedures (object *Procedure*). Objects of type *Class* are symbolic images over which logical conditions can be imposed to define other classes by means of *Rule* objects. Finally, the object *Interface* (not included in Figure 2) contains the names of the classes that are accessible from upper levels.

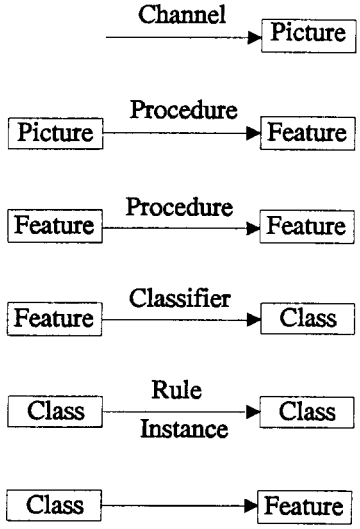

Figure 2. Pixel Processor's image types (inside boxes) and their possible transformation by means of operators (over arrows).

The classifiers (object *Classifier*) carry out the transformation of a numerical description into another of symbolic nature by means of two consecutive processes. The first process defines the features that are used and how they are combined (lineally, quadratically) to generate a descriptor. The second process maps the value of the descriptor into a fuzzy logic range [0-100] according to a certain decision function model (sigmoid, threshold, exponential, . . .).

2.2 Segment Processor

The second level of SVEX is conceived around the Segment Processor. The output of this level is an image partition formed by segments (connected aggregation of pixels defined by shape and/or property criteria) along with their corresponding assignment to symbolic segment classes, their spatial localizers and the spatial relations among neighbors. The BU unit receives data pixel diagnostic maps (pixel classes), previously requested by the TD unit to the Pixel Processor, and produces symbolic diagnoses for the segments of the partition. The TD unit receives diagnostic requests about segments, and transforms these requests into sequences of commands for the BU unit and into requests for pixel diagnostics directed to the Pixel Processor.

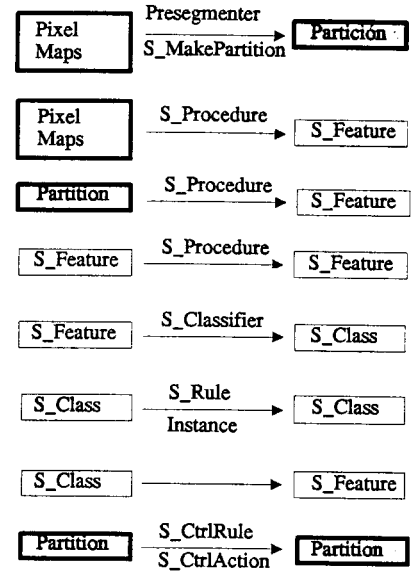

Figure 3. Segment Processor's data types (inside boxes) and their possible transformation by means of operators (over arrows).

The Segment Processor imposes two specific requirements. The first one, the need for a definition of the segments from the pixel maps, is met by means of the presegmenter action, which generates the initial image partition (Figure 3). The second one is the desirability of having a refinement or control mechanism over the partition. For this purpose the control actions make possible the modification of the spatial definition of the segments using different operations (merge, split, include, . . .).

From the point of view of functionality, the Segment Processor's architecture is organized around three blocks. The first one is the presegmenter, conceived as an interchangeable part within the BU unit (currently an adaptation of the Watershed transform [Vinc-91]). The second block has the goal of computing the segment diagnoses, and is structured, as in the Pixel Processor, by the distinction between a numerical domain, where the segments are described by features, and another symbolic domain based on classes. The third functional block is located at the TD unit and manages the diagnostics requests received by the Segment Processor as well as the partition control.

The objects used in the programming of the Segment Processor are shown in Figure 3. The *S_MakePartition* object controls the presegmenter action. *S_Procedure*, *S_Classifier*, *S_Class* and *S_Rule* objects are equivalent to their homonyms in the Pixel Processor, but being now applied to segments instead of pixels. The *S_Condition* object permits to express premises for the rules that may include spatial relations (below, left, contains, . . .). *S_CtrlRuleSet*, *S_CtrlRule* and *S_CtrlAction* objects can be used to define control actions over the partition. The *S_Interface* object determines which segment diagnoses are visible to the upper level (typically an user-provided program module) and the *P_Interface* object indicates which pixel classes are to be requested to the Pixel Processor.

3. APPLICATION

The pictures shown in figures 4-12 illustrate SVEX's results in the segmentation of two outdoor images. Both images have been processed with the same programs set, without computing complex or high computational cost features.

The pictures include for each example the source image (figures 4 and 5), the final partition obtained by means of refinement of the initial partition (figures 6 and 7), and a selection of some segment classes from this final partition, which are depicted as white areas overlapped to the gradient map.

4. CONCLUSIONS

SVEX proposes a scalable multilevel computational structure that allows both numerical and symbolic computations to take place at each level. The computational structure itself defines the transition between the numerical and symbolic domains clearly and consistently. Conceived as a tool, SVEX programming language permits a fast and flexible development of applications, making easier its maintenance. The result is an open system that can be easily adapted to a great variety of segmentation problems.

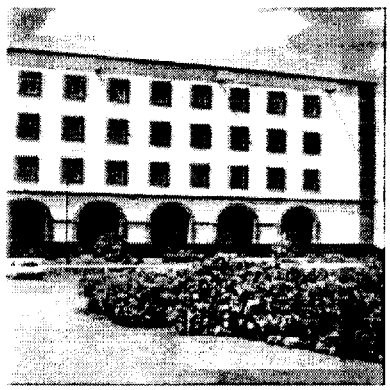

Figure 4. Source image A.

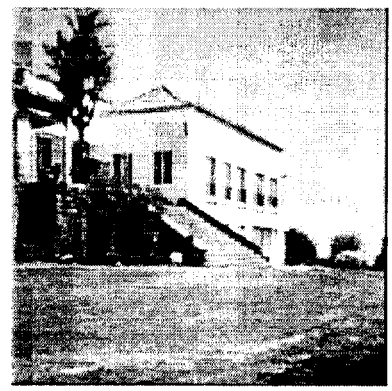

Figure 5. Source image B.

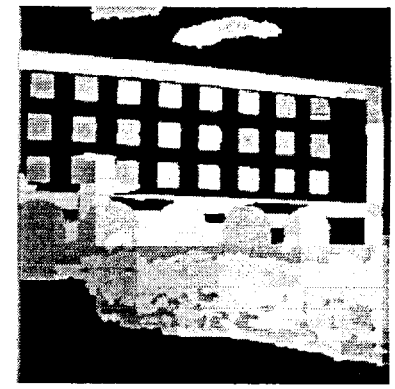

Figure 6. Final partition (A).

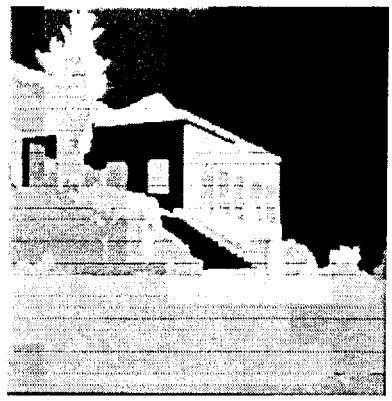

Figure 7. Final partition (B).

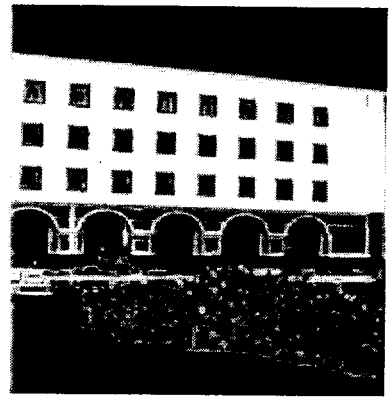

Figure 8. Front class (A).

Figure 9. Front class (B).

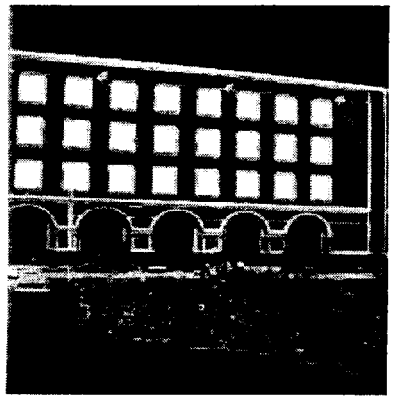

Figure 10. Window class (A).

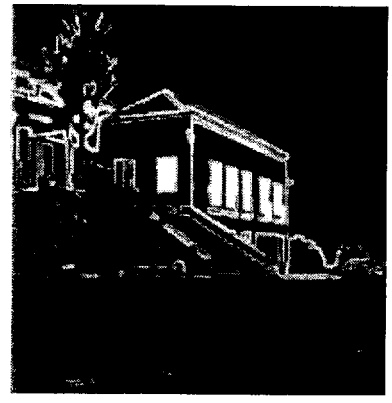

Figure 11. Window class (B).

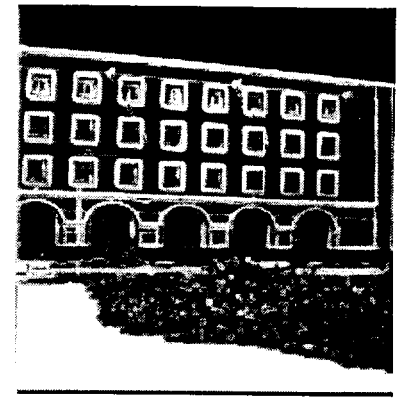

Figure 12. Road class (A).

5. REFERENCES

[Mend-94] Méndez J, Falcón A., Hernández F., Cabrera J., *"Development Tool for Computer Vision Systems at Pixel Level"*, Cybernetic & Systems, Vol 25, n°. 2, pp. 289-319, 1.994.

[Nazi-84] Nazif A., Levine M., *"Low Level Image Segmentation: An Expert System"*, IEEE Trans. on Pattern Anal. and Mach. Intell., Vol 6, n°. 5, pp. 555-577, 1.984.

[Stra-92] Strat T.M., *"Natural object recognition"*, Springer-Verlag, New York, 1992.

[Vinc-91] Vincent L., Soille P., *"Watersheds in Digital Spaces: An efficient algorithm based on immersion simulations"*. IEEE Trans. on Pattern Anal. and Mach. Intell., Vol 13, n° 6, pp. 583-598, 1991.

[Wils-88] Wilson R., *"Is Vision a Pattern Recognition Problem?"*, in "Pattern Recognition", Ed. J. Kitller (Ed), 1-25, Springer-Verlag, 1988.