An Intelligent and Task-independent Controller for Video Sequence Analysis

José BINS, Thor LIST, Robert B. FISHER, David TWEED

Abstract—**This paper describes a task-independent controller that allows for an easy implementation of vision systems for processing video sequences. The controller does not have a fixed dataflow or any fixed steps. The dataflow is constructed by the modules by describing themselves for the controller.**

During operation the modules and their parameters are selected using an independent decision module. This makes the system flexible and allows comparison of different learning techniques and decision strategies. The controller is being used by the CAVIAR system and its current decision module is a rule-based system written in Clips.

Index Terms—**Intelligent controller, Computer Vision, Video sequences analysis, Image Understanding**

I. Introduction

Over the past 2 decades there has been a great amount of work on the development of generic image understanding front-ends so they can be reused on different image understanding problems, what could be called an image understanding shell [7]. We have still to achieve that goal, but the characteristics of those systems are beginning to get clearer.

Among the problems that prevented the achievement of such systems are the complexity and computational cost of low-level and intermediate level operations necessary for image understanding. This problem is smaller nowadays since there are a great number of off-the-shelf Computer Vision libraries of operators available on the Internet. These libraries provide the necessary operators for many understanding tasks, but they need a method to automatically select which libraries to execute, over which data and with which parameters. That means an intelligent controller capable of reasoning and acquiring knowledge about the task, the operators, etc.

Although most researchers agree on the importance of studying and formalizing image understanding controllers, there is no agreement, though, on how that should be achieved. Some researchers ([2] & [9]) advocate addressing the control problem as a separate problem from the image understanding task, while others ([10]) claim that vision systems can not be designed in isolation from the task. The controller described here tries to achieve a balance between those two approaches. The designed controller is independent from the task but it allows the acquisition of task specific knowledge and acts over it. It is one further step in creating a shell for image understanding problems. Its main characteristics are: it is independent of the desired task but it allows for specific knowledge to be added; it is independent of the learning and decision techniques used; it is a centralized controller but it allows the use of agents[1] to compute useful features and evaluate modules; it does not execute a fixed dataflow but instead constructs the dataflow for each run by asking the modules about their descriptions; it is capable of selecting the best module, between equivalent modules[2], for a given frame; it is capable of selecting the best set of parameters for a given module at a given frame; and it tries to maximize frame output rate and output quality

II. Control for Image Understanding

Up to the 70's most Image Understanding Systems had embedded controllers. The study of control techniques began in the 80's. Systems from this decade began to use expert knowledge to control them. Most of those systems were Rule-based systems [16], Blackboard systems [1] and Semantic Networks [12]. The main flaw of these systems was that they were too specific which made then not robust enough when applied to new domains, probably due to their ad-hoc construction [9]. In the 90's a new breed of less ambitious IUS arose which tried to explicitly model the control process. Those systems used Bayes Nets [18] and Markov models ([17], [8]). For more information see reference [7] that contains a good survey on image understanding systems. Future directions show a second generation of Bayes Nets and Markov model systems and High Dimensional Decision making. The controller presented here tries to be independent of the reasoning technique. This makes it a good environment for comparing different approaches.

III. Controller description

A. The Architecture

The controller described here was designed to be task independent, although able to use task specific knowledge, if such knowledge is available. It is written in Scheme and runs

José Bins, Robert B. Fisher, Thor List and David Tweed work at the IPAB (Institute of Perception Action and Behavior) of the University of Edinburgh, Scotland, UK. This work was developed for the CAVIAR project (Context Aware Vision using Image-based Active Recognition) under EC project number IST-2001-37540. (Emails: jbfilho/tlist/rbf/dtweed@inf.ed.ac.uk)

[1] Agents here are program that act on behalf of the controller computing features and/or evaluating data. They are not part of the sequence of operators that compute the task and although we call them agents they are not currently autonomous programs. This may change in the future.

[2] Modules are considered equivalent if they generate the same o

Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)
0-7695-2255-6/05 $20.00 © 2005 IEEE

on the Imalab environment [15]. The modules are written in C++ and use the Caviar Base System [13] to interface with the controller. The overall architecture of the controller is shown in Figure 1. To run the controller the only input needed is which operators to use, here called modules, which agents are available, and which sequences of video to use (if offline option). The controller then gets information dynamically from the modules, from the agents, and from the decision module and uses this information to run the system. The controller is independent of the decision-making module. That is achieved by defining a small set of functions that interfaces the controller and the decision module. The learning controller is an offline control for learning task specific knowledge. It is also independent of the learning technique implemented by the learning module.

1) Control loop

The control loop executed by the controller is shown in Figure 2. At the initialization step the controller asks the modules to auto-describe. This description is an XML string and contains information such as parameters and their domains, inputs and outputs. Figure 4 shows an example of such description. With this information the controller creates the dataflow, which can have different paths to achieve the same task. This dataflow is created in a bottom-up approach. Alternatively, only a goal-output may be given to the controller. This allows the controller to create the dataflow using a top-down approach. This dataflow has the advantage that operators that do not contribute to the computation of the goal-output will not be included.

An example of dataflow created by the controller is shown in Figure 3, which shows a graph of the CAVIAR system (see description on section IV). In this dataflow the squares represent modules and the ellipses represent data. There are two modules "sensing1" and "sensing2" that are equivalent. This means that the controller will select only one of them during execution. There are no inputs shown because the input is the camera stream and not a file.

After the initialization the controller executes the loop itself. For each step of the loop: the controller asks the decision module for the necessary information concerning the step; executes the step; and sends the decision module the information necessary for it to update its knowledge. A brief description of each step is given below.

- The selection step selects which module to execute. This implies not only which file to generate next, but also which module to generate it, if more that one is available. For example, if there are two versions of the module that tracks people the controller may select one or the other depending on which one is better for the current frame.
- The execution step executes the module selected for a set of parameters. These parameters are adjusted depending on some simple features computed over: the current frame; the inputs for the module, and feedback given by the modules that generated the inputs.

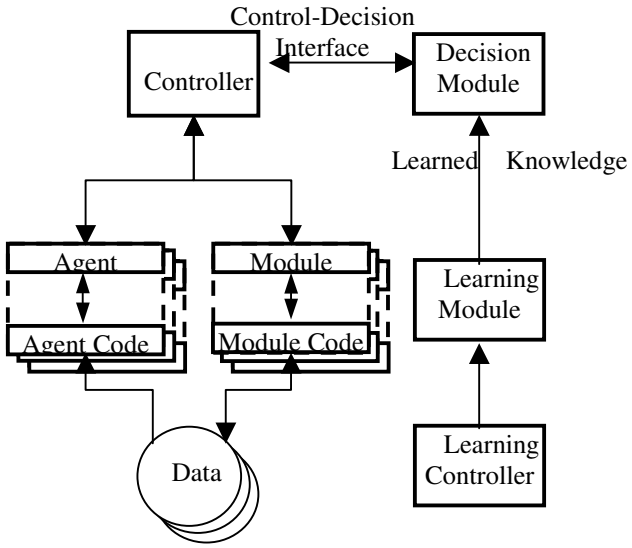

Fig. 1. Controller architecture

- The module evaluation is done based on each module's auto-evaluation plus information acquired during the offline-learning phase. In addition, during the learning phase the controller decides if it should trust the module's auto-evaluation by comparing it with the evaluation over the ground truth.
- In the repair step the controller may decide to rerun some modules with different parameters for one or more of the previous frames based on the evaluation result.

B. Controller-modules interface

Each module and agent has an interface with the controller provided by the Caviar Base system [13]. This base system is written in C++ and uses the PrimaVision library [15], which provides the functionality for video and image manipulation, and the CoreLibrary [6], a powerful multi-platform library for C++ and implements the CVML language (see below). This interface is capable of inputting files and parameters, saving and restoring the program state, and communicating with the controller. The controller is able to send commands for the modules and the modules can return feedback. (See [13] for a more in depth description of the base system). This allows for the modularization of modules releasing their authors to concentrate on the specific task of each module

The most interesting aspect of the interface is that it provides mechanisms for modules to be auto-regulatory, auto-descriptive and auto-critic, allowing the controller to use this information to improve the control. This follows an approach proposed by Crowley & Reignier ([11]). Our approach is a little different though, since in their case they proposed a hierarchy of controllers where each controller, and not the modules, were auto-regulatory, auto-descriptive and auto-critic.

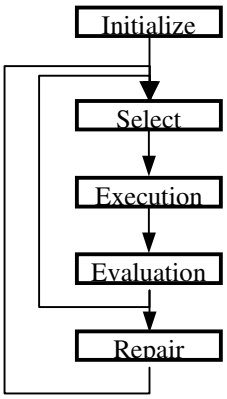

Fig. 2. Control loop

1) Module auto-description

Each module describes itself for the controller using an XML based markup language called CVML [14]. A simple example is given in Figure 4. This description is very powerful and essential for the controller. The controller uses the input and output description to create the dataflow; uses the parameters descriptions to learn strategies to set them during execution; uses the output variables to decide which evaluation agent should be called and much more. A full grammar of this description is given in [4].

2) Module auto-critic

The controller expects each module to evaluate its outputs. This evaluation will be passed to the controller by a feedback mechanism with two levels. The high-level feedback contains the overall performance of the module and has the number of objects outputted, the execution duration and an estimate of the module's output quality. The low-level feed-back contains more detailed information like, for example, a confidence level for each object, the position of each object, etc. Most of the information provided on the feedback is straightforward to compute, but auto-estimating the output quality can be difficult, so the controller does not trust that information a-priori. Instead, the auto-evaluation is compared with an evaluation computed over the ground-truth during the offline learning. From this comparison the controller computes a confidence factor on the module's auto-evaluation. If this confidence is high the module auto-evaluation can then be used during execution to evaluate the module's output and to dynamically learn new decision strategies, if the learning module has this capability.

3) Module auto-regulation

Besides controlling the modules, the controller is capable of sending recommendations to the modules. A module then can decide to auto-regulate its parameters to meet the recommendation or not. An example of a recommendation can be: <controls recommendation="increase threshold" />. The module then would have to interpret the recommendation and change its behaviour to reflect it. Although this feature is already implemented in the Caviar Basic system it is not yet being used by the controller.

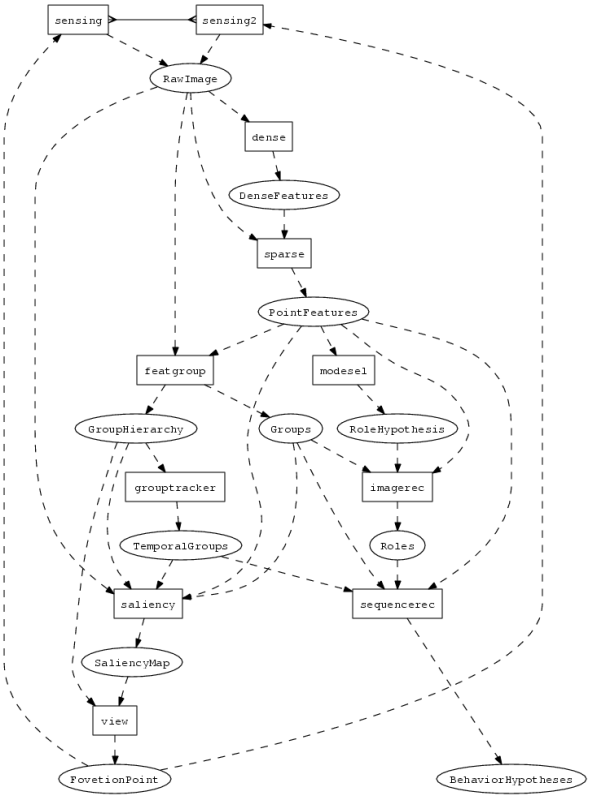

Fig. 3. Example of dataflow constructed by the controller using the modules information. This data flow shows the complete dataflow of the CAVIAR system. The modules connected by ">--<" are equivalent modules.

C. Decision module

The decision module implements the reasoning algorithm used to make the decisions. It is independent of the controller allowing the user to plug and play different reasoning and knowledge strategies. The controller asks the decision module for suggestions and sends all information used by the decision module. The decision module does not compute any features, run modules or agents, and the controller does not do any reasoning. In order for that independence to work a good interface between the controller and the decision module is necessary. The proposed interface uses very simple functions that can be written in any language accepted by Imalab (Scheme, C++, Prolog). The current functions used are shown in Table I.

D. Learning module

The learning module implements the learning algorithm, such as a rule based system, Bayes network, neural nets system or any other desired strategy. The learning module receives from the learning controller the feedback from the module (auto-evaluation, duration and number of objects); all the computed features; and an evaluation of the modules output for combinations of: module, video sequence, parameters and frame. It can then use this information to create a knowledge base and learn decision strategies. The learning module can select how this information will be generated by setting how the parameter space will be searched and if the parameters should be considered independent or not.

```
<description>
  <parameters count="1">
    <parameter name="ThetaAz" type="float" optional="no">
      <description>Azimuth for the camera</description>
      <range from="0" to="360"/>
      <default>0</default>
    </parameter>
  </parameters>
  <dataflow>
    <inputs count="1">
      <input frame="-1" dataset="FoveationPoint" />
    </inputs>
    <outputs count="1">
      <output dataset="RawImage">
        <variable name="Time" type="Time" />
        <variable name="Image" type="TBitmapByte" />
      </output>
    </outputs>
  </dataflow>
</description>
```

Fig. 4. An example of a module description in CVML

TABLE I
Control-Decision module interface functions

Name	Description
initializeknowledgebase	Initializes the files and defaults necessary to run the knowledge module
knowledgebaseok?	Returns the status of the knowledge base (if it is initialized and ready to be inquired)
finalizeknowledgebase	Finalize the knowledge base; close files, end logs
updateknowledge	Asks the decision module to make the knowledge base up-to-date.
resetknowledgebase	Puts the knowledge base in to the initial state
getnextmodule	Selects the next module to run
getparametersuggestion	Returns a suggestion of the value to be used for a module's parameter
getdatasuggestion	Returns the module that created the file to be used
updatedata	Updates the data base for a new file
setprogdone	Tells the decision module that a module was executed
setprogout	Tells the decision module that a module can not be executed
savefeedback	Saves the module feedback on the knowledge base
saveframeinfo	Saves the frame features on the knowledge base
getfeaturestocompute	Returns the name of features to compute for this run of the system

IV. Example of Use

A. The CAVIAR project

The controller was implemented as part of the CAVIAR (Context Aware Vision using Image based Active Recognition) project [5]. The CAVIAR project is funded by EC and is a collaboration between three institutes: Instituto Superior Técnico in Lisbon, Portugal; Laboratorie GRAVIR-IMAG in Grenoble, France; and Institute for Perception Action and Behaviour, Edinburgh. The main objective of the CAVIAR project is to recognize behavior in video sequences. The two applications that the project addresses are: city centre surveillance, trying to identify fights, unusual events, etc.; and marketing, trying to identify customer behaviour. The CAVIAR project constructed a data set with about a 100 sequences of videos with an average length of 1000 frames each. Every sequence has a corresponding ground truth file, which is an XML description of the objects (person or group of persons) found in each frame. Each object is encased in a box and a scenario is attached to it. A scenario is a set of 4 attributes: movement, role, situation and context. See CAVIAR Home page [5] for details on the ground truth format.

B. The decision module implemented

To test the controller, two decision modules have been implemented: a rule-based system using Clips and a back-propagation neural net system. In this paper only the rule-based system will be described.

The rule-based decision module is able to select the next module to execute and to suggest parameter values using both learned and user defined facts and rules. It allows an expert user to provide rules in addition to the learned ones, covering cases not addressed by those. Examples of user rules are given in Figure 5. The rules are written in Clips, what makes it easy to reuse them since Clips is used as rule manager in many rule based systems. In addition to Clips commands the decision module provides a set of facts and functions to help the user.

The facts include: the current frame; features computed over each frame; the module's feedback and evaluation, etc. The functions include functions to: set priority between programs; force or deny the execution of a program; set parameter values or increments, etc. The rules in Figure 5 exemplify some of those facts and functions.

The first rule states that if the module **sensing1** is on a frame for which feature **luminance** is between 0.5 and 0.2 then module **sensing1** has a priority over module **sensing2** of 1.2. This means that when the comparison criterion[3] is tested, **sensing2** will only be selected over **sensing1** if its criterion multiplied by 1.2 is still less than **sensing1**'s criterion. The second rule states that when the output rate of the system computed at a given frame is less than 80% of the desired output rate the parameter **ThetaAz** for module **sensing1** should be decremented by 50%.

C. The learning module implemented

The learning module implemented is very simple. It computes a function of each parameter for each feature and creates rules to do hill-climbing on them. Here the features are assumed independent for simplicity. We are aware that this assumption does not hold in many cases. A neural nets learning module is being finalized and more sophisticated versions of the learning module are planned, but that is a first implementation of the learning module as a proof of concept. The rules and facts exemplified here were computed over a different system from the one shown on Figure 3. Figure 6 shows a dataflow graph of this system, which is a simplified version of the CAVIAR system and uses 6 modules. Module **tracker** tracks objects and puts boxes around them; **grabber** gets a frame from disk[4]; modules **movement1** and **movement2** hypothesize about the movement of objects (active, inactive,

[3] Currently the knowledge module is able to use the following features as comparison criterion to select modules: time they were ready last; time they executed last; frame they executed last; last duration; best evaluation and no feature. More features will be incorporated with time

[4] In this case the controller is running offline.

```
(defrule luminance1
    (userrules)
    (proginfo (prog  sensing1) (name frame) (value ?fr))
    (frameinfo (number ?fr) (name Luminance) (value ?v))
    (test (and (< ?v 0.5) (>= ?v 0.2)))
    =>
    (setprogpriority  sensing1 sensing2  1.4)
)
(defrule outputrate
    (userrules)
    (currentframe ?fr)
    (frameinfo (number ?fr) (name outputrate) (value ?v&:(< ?v 0.8)))
    =>
    (setincrementrate sensing1 ThetaAz -0.5)
)
```

Fig. 5. Example of user rules. The modules referred here relate to the dataflow shown on Figure 3.

walking, or running) using two different approaches (logical and probabilistic); module **role** hypothesizes about the role of objects; and module **context** hypothesize about the context (behaviour) of objects.

The function computed for module **movement1** and its parameter Base_Classify_Threshold is shown in Figure 7. The quality feature is the module auto-critic, which in this case was fixed at 1, and the evaluation is the result of the ground truth comparison. From this function a set of facts and rules are created to be incorporated in the knowledge base of the decision module. Figure 8 shows two facts and one rule generated for module **movement1,** parameter Base-Classify-Threshold, and feature evaluation.

D. Running the controller

We tested the controller over the simplified version of the CAVIAR system (see Figure 6). The system is able to control the system, selecting between movement1 and movement2 depending on features computed over the output of modules and selecting the best values for parameters. Figure 9 shows a graph where modules are selected depending on the average distance of the boxes from the plane of the camera. Each point represents a module. Points with **y** coordinate greater than zero represent the module selected by the control for the frame. The module's **y** coordinate represents the value of the feature when the module was executed. The module that was not executed is plotted with **y = 0**.

From the graph is easy to see that when the feature goes below a threshold of 215 the control switch modules. This rule is user defined[5]. Figure 10 shows the evaluation of the results of **module1** when the system is executed over 30 frames. The graph is cumulative in the sense that at each frame a file containing all previous frames is evaluated.

Nevertheless, at each time the module computes the movement for all boxes of all frames what means that values for previous frames may be changed.

[5] A learned rule was not used because module **movement1** was always the best module independent of the feature used consequently module **movement2** would never have been selected. It is important to note that we are not claiming that the technique used on module **movement1** is better or worst than the used on module **movement2.** Those modules are still in development and they are used here only as tools to show the execution of the controller

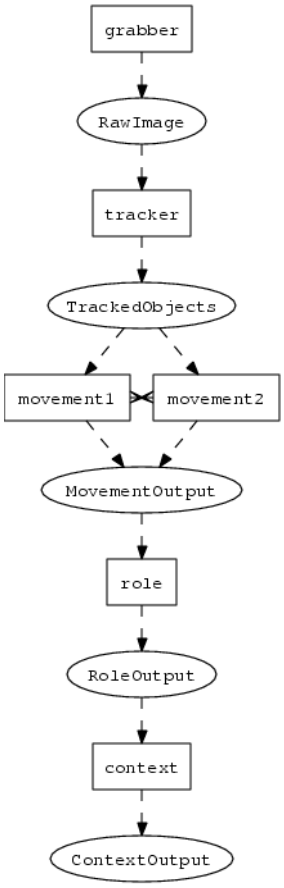

Fig. 6. Simplified version of the CAVIAR system used to learn rules for the decision module. Here are two equivalent modules **movement1** and **movement2**.

The graph shows that the controller was able to maintain a high level of quality while changing the module's parameters.

V. Future Work

The proposed controller is being used on the CAVIAR system to control the execution of modules, although, to show the full potential of it some work must still be done. The repair phase of the control loop is not implemented yet; more features must be incorporated to the controller so decisions can be made more accurately; more modules must be incorporated; etc.

Nevertheless, the more important future step is the construction of alternative decision and learning modules. An learning module using neural-nets similar to the ADORE system [8] is been finalized and others as a Bayes network module or adjusting the parameters by constructing a function by least square approximation [3] are being considered.

VI. Conclusion

We presented a task-independent controller for video sequence analysis, which is a new step in the search for a shell for image understanding system. This controller is independent of the reasoning and learning techniques, which makes it very useful for testing and comparing those techniques. Incorporating a new learning technique to the system is very easy and is done by a simple set of interface functions that can

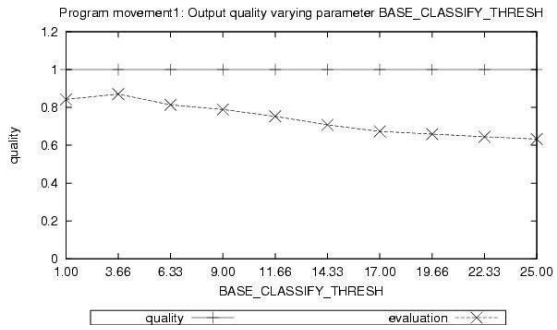

Fig. 7. Auto-critic (quality) and output evaluation for module **movement1**.

be implemented in three different computer languages. The Caviar Base system provides easy to use functions that allow the modules to be auto-descriptive, auto-critic and auto-regulatory.

In order to show the potential of such controller we developed a decision module in Clips that accepts learned and user defined rules and facts. The current controller is being used by the CAVIAR project to run a simpler version of its modules.

A lot of work still needs to be done to show the full potential of the system, especially the construction of better decision modules.

```
   (of searchrate (module (symbol-to-instance-name
movement1)) (parameter (parametername movement1
BASE_CLASSIFY_THRESH)) (feat evaluation) (begin
1.00000) (end 3.66667) (rate 0.001062)) ))
   (of searchrate (module (symbol-to-instance-name
movement1)) (parameter (parametername movement1
BASE_CLASSIFY_THRESH)) (feat evaluation) (begin
3.66667) (end 6.33333) (rate -0.005716))

  (defrule
learned_movement1_BASE_CLASSIFY_THRESH_evaluation
   (userrules)
   (proginfo (prog movement1)(name frame)(value ?fr))
   (proginfo (prog movement1)(name evaluation)
     (value ?gl&:(< ?ql 0.900000)))))
   (object (is-a floatpar) (name
     [movement1BASE_CLASSIFY_THRESH]) (value ?v))
=>
   (bind ?inc (computeincrementrate movement1
     BASE_CLASSIFY_THRESH evaluation upper ?v ?gl))
   (setincrementrate movement1 BASE_CLASSIFY_THRESH
     ?inc)
)
```

Fig. 8. Examples of facts and rules learned by the learning module for the module **movement1**.

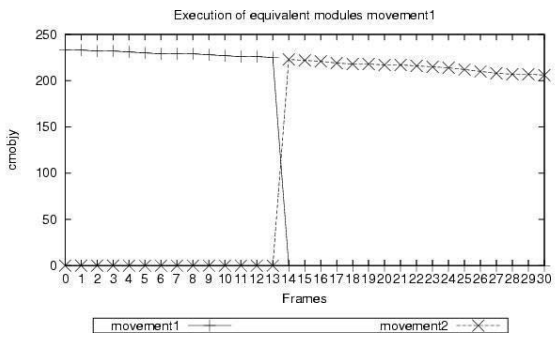

Fig. 9. Module selection. The module selected is represented by a point with the feature value and the one not selected with a point over the abscissa.

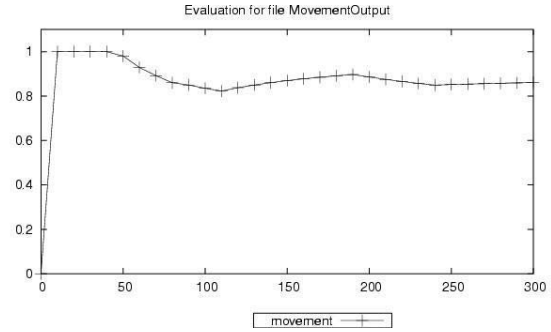

Fig. 10. Evaluation of the output of **module movement1** by comparing it with ground truth.

REFERENCES

[1] K. M. Andress and A. C. Kak, Evidence Accumulation & Flow of Control in a Hierarchical Spatial Reasoning System, AI Magazine, 1988, 9(2), pp. 75-94.

[2] V. Bulitko, G. Lee, I. Levner, and *L. Li*, Open challenges in learning vision systems. In *NIPS-03 Workshop on the Open Challenges in Cognitive Vision*, Whistler, BC, Canada, December 2003.

[3] A.Caporosi, D. Hall, P. Reigner, and J. L. Crowley, "Robust visual tracking from dinamic control processing", Proc. of 6th International Workshop on Performance Evaluation for Tracking and Surveilance, PETS-ECCV, pp 23-32, May, 2004.

[4] CAVIAR module description grammar page: http://homepages.inf.ed.ac.uk/jbfilho/caviar/module_description.html

[5] CAVIAR Homepage: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

[6] CMLabs CoreLibrary Homepage: http://www.cmlabs.com/corelibrary

[7] D. Crevier and R. Lepage, Knowledge-based Image Understanding Systems: A Survey, Computer Vision and Image Understanding, V 67, N 2, August, pp. 161-185, 1997.

[8] B. Draper, J. Bins and K. Baek, ADORE: Adaptive Object Recognition. Videre, 2000, V 1, N 4, p.86-99.

[9] B. Draper, From Knowledge Bases to Markov Models to PCA, Workshop on Computer Vision System Control Architectures (VSCA'03), Invited Talk, 2003

[10] J. Crowley and H. Christensen, Integration and control of active visual processes, Workshop on Integration and Control for Computer Vision, on Conference on Intelligent Robots and Systems (IROS'95), 1995

[11] J. L. Crowley and P. Reignier, An Architecture for Context Aware Observation of Human Activity, Workshop on Computer Vision System Control Architectures (VSCA'03).

[12] V. Hwang, L. Davis and T. Matsuyama, Hypothesis Integration in Image Understanding Systems, Computer Vision, Graphics and Image Processing, 36(2):p. 321-371., 1986.

[13] T. List, J. Bins, R. B. Fisher, and D. Tweed, A plug-and-play architecture for cognitive video stream analysis, IEEE CAMP'05 International Workshop on Computer Architecture for Machine Perception, 4-6 July, 2005.

[14] T. List, R. B. Fisher, "CVML – An XML-based Computer Vision markup language", Proc. Int. Conf. On Pattern Recognition, Cambridge, V 1, pp 789-792, 2004.

[15] A. Lux, The Imalab Method for Vision Systems, International Conference on Vision Systems, pp 314-322, Graz, Austria, April 2003.

[16] D. M. McKeown, W.A. Harvey and J. Mcdermott, Rule-Based Interpretation of Aerial Imagery, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1985, 7(5), pp. 570-585.

[17] J. Peng and B. Bhanu, Closed-loop Object Recognition using Reinforcement Learning, IEEE Transaction on Pattern Analysis and Machine Intelligence, 1998, V. 20, N 2, pp 139-154.

[18] R. Rimey and C. Brown, Control of Selective Perception using Bayes Nets and Decision Theory, International Journal of Computer Vision, 1994, V 12, pp. 173-20