

# Exploring techniques for behaviour recognition via the CAVIAR modular vision framework

David Tweed, Wanli Feng, Robert Fisher, José Bins & Thor List  
Institute for Perception, Action and Behaviour,  
Dept of Informatics, Edinburgh University,  
{tweed,rbf,jbfilho,tlist}@inf.ed.ac.uk

## Abstract

Visual analysis tasks are often very difficult and computer systems for tackling them are most effectively implemented as a network of routines which solve constituent subproblems with well-defined specifications. Most of these tasks do not have a clear-cut best algorithm, particularly dealing with real-world data. The problem arises how to characterise the performance on these sub-tasks in order to choose the best implementation for a given application, particularly when several techniques are used within a dataflow graph, with the output of later algorithms dependent on the output of earlier routines. In this work we look at competing algorithms for a simple behaviour recognition task within the framework of the CAVIAR modular vision platform.

## 1 Introduction

Creating software for challenging real-world visual analysis tasks is made much more tractable by adopting a modular approach, decomposing the problem into well-defined subtasks. Selecting the best algorithms the various sub-tasks is predicated upon characterising the performance of these routines.

Although the subtasks may correspond to operations a human is aware of performing – eg, detecting sudden movement – often they are motivated by creating intermediate data representations for use as a stepping-stone between low- and high-level representations, eg, consider image segmentation. This can be defined as dividing an image into contiguous regions maximising some measure of ‘intra-region homogeneity and cross-region difference’. Many applications segment an image into ‘potential objects’ before processing this reduced data to, eg, detect moving objects or decide if certain objects are present. Segmentation is clearly a useful and meaningful computer vision sub-task. However, virtually no real world task has subjects aware of segmenting their visual fields.

For a natural task, performance can be characterised by obtaining a *ground truth dataset* for an input dataset from human test subjects and then comparing this with the system’s output via established methodologies such as *Receiver Operating Characteristic (ROC)* curves [3]. This assumes ‘success’ is defined by human performance on the task. The extension to less natural sub-tasks by explaining the task to test subjects and getting them to produce ground truth data raises two issues: Firstly, the task definition has often been guided by algorithmic foreknowledge, so there can be an influence on the supposedly independent ground truth by the algorithm. More significantly, success is really defined not by human performance *on this subtask*, but rather by how the *overall system* performance on the end application (measured wrt a ground truth) is affected by using a given sub-algorithm. As an example, imagine a system for detecting cars starting with segmented images where the detection performance is governed just by the how accurately the distinctively shaped wing-mirrors are segmented, making comparison against a general segmentation ground truth misleading. If one *knew* wing-mirrors were crucial a tailored ground truth could be created, but in general one does not know these details; indeed, black box modules make *non-experimental* determination of the important aspects of the subtask output impossible. (We use segmentation in this and our later *examples to illustrate the framework* since segmentation is a more widely known basic subtask than the specialised behaviour analysis tasks which we evaluate later.)

This paper looks at how the final performance of a pipeline of analysis modules, which build up estimates of the behaviour of tracked individuals in a video sequence, varies with the performance of the earlier modules, an issue arising within the development of the surveillance system CAVIAR. It is important because the potential interactions/correlations between subtasks make attempting to tune the module in isolation misleading. This problem will worsen as the vision system becomes more modular.

**Previous work.** There has been much impressive work in the vision community on building systems for complex visual analysis, especially recently (see eg, the survey article [14]). Since in this work we are focusing on performance evaluation within the context of a larger task, we discuss in detail previous work in the setting of understanding and characterising algorithm performance.

Most machine learning focuses on two errors: false positives  $p$  and false negatives  $n$ , which a given algorithm can trade-off by varying its parameters. Algorithms thus have *ROC curves* [3] in  $p$ - $n$  space characterising achievable

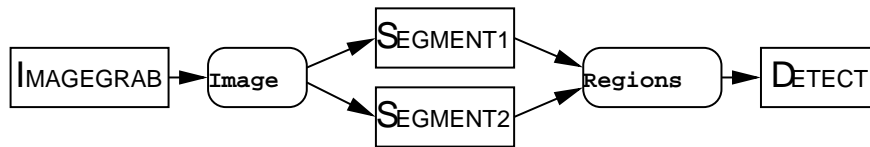


Figure 1: From [SEGMENT1, in=Image, out=Regions] and [SEGMENT2, in=Image, out=Regions] CAVIAR deduces SEGMENT1 & SEGMENT2 are functionally equivalent

results. To build a system we choose a  $p$ - $n$  ratio and select the associated parameters. Work such as [5] extends this to the general situation where we have  $N$  measures – eg, runtime, storage, etc – where it’s difficult to specify the trade-off *a priori* and there are multiple algorithms to compare. It should also be noted there is extensive literature (eg, [7, 9]) on statistically valid ways of estimating the variance of performance outputs with respect to variability in the input data, particularly given limited test data. Estimates of variance are necessary to decide whether differences between algorithms are significant.

In this paper we consider pipelined processes where the output of earlier modules becomes the input of later ones. In this vein, [16] considers the task of retrieving images with between  $l\%$  and  $h\%$  of the image being in some class (eg, sky, foliage, etc). Their detectors process sub-blocks of the image independently but can produce both false positives and negatives. A preprocessing module is developed which converts the user query to an synthetic query ( $l', h'$ ) designed to ‘cancel out’ expected detector errors and produce output matching to the original query. [11] investigates how the performance of various face recognition algorithms depends upon the accuracy of the eye localisation algorithm used for preprocessing. Experimental results suggest (counterintuitively) that absolute eye position errors have less effect on the recognition rates than merely eye separation and orientation (essentially because these features are used for normalisation). Finally, [10] analyses the performance of region classification algorithms given different segmentation algorithms in terms of extended ROC curves highlighting performance (with variance) for multi-class assignment. They demonstrate that a sensitivity to the choice of segmentation algorithm that is (again non-obviously) class-dependent.

## 2 The CAVIAR modular vision architecture

The name ‘CAVIAR system’ [2] refers to both a general software framework for implementing modular vision systems and a test system using example modules for visual surveillance/behaviour analysis. We give an overview, concentrating on issues of relevant to performance characterisation and omitting other interesting aspects of the architecture.

One way to look at the CAVIAR system is to start with the notion of *dataset* descriptions. These are ‘meta-data’ which essentially specify names and associated data structures. For example, there is a dataset named `TrackedObjects` which defines a syntactic representation for bounding boxes of individuals as they move through a video sequence. The lowest-level software element is the *module*, which is a block of code combined with a record of the datasets it takes as input and produces as output. The module shares the common understanding about what datasets mean, analogous to programming language interfaces, (eg, ‘`TrackedObjects` describes individuals from a low-level tracker’).

The highest level software is the *controller*, a standard routine which is responsible for taking a collection of modules and figuring out firstly how they connect to form a complete system and then running this system over some input data. Connections are determined by building a directed-graph-like structure with two kinds of nodes – modules and datasets – and edges connecting a module with its inputs and outputs. In the case where each dataset is listed as an output for exactly one module this generates a simple dataflow analogous to a flowchart. However, the real utility arises when there is more than one module producing output  $x$ . In the simplest case (illustrated in Fig 1) we have two modules `SEGMENT1` and `SEGMENT2` which both produce dataset `Regions` as output. The controller automatically deduces that it can obtain `Regions` from *either* `SEGMENT1` *or* `SEGMENT2`. This knowledge of functional equivalence can be used by the system controller to prompt the user to make a fixed choice of modules before running the system, to switch based upon static rules learned from offline performance characterisation, or to use a sophisticated auto-critical framework utilising intra-module feedback to swap modules (cf [4]). The first two approaches can use the analyses we develop here.

This highlights that the system is ‘discovered’ *from the bottom up* rather than specified top down: the controller does not have any built-in expectation about what inputs the `Regions` dataset needs or whether a single or multiple modules (using intermediate datasets) generate it, instead it splicing together modules to generate all outputs.

## 3 Behaviour analysis & interpretation

We used 80 image sequences of 500–2700 frames obtained from 3 separate camera-setups France, PortugalAcross, PortugalFront (taken from the CAVIAR datasets available from [1]). The ground truth describes the behaviour of both the individuals in the sequence and of groups of people, although here we only use individual behaviour. Since we will

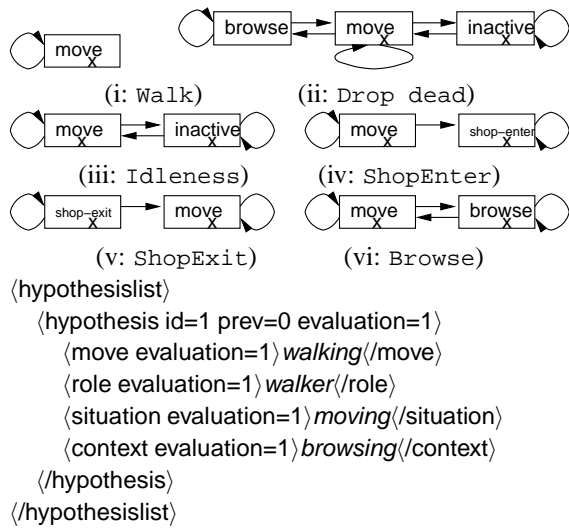


Figure 2: (a) some context graphs; (b) excerpt from ground truth/module output format (see [8]); (c) images 100 frames apart from typical sequences from the test datasets

be assessing performance against a pre-existing ground truth, the human interpretation incorporated into the ground truth must be structured into a form algorithms can output. The ground truth describes the *behaviour* of people within the sequences at a various levels of sophistication via four attributes per frame (expanded upon in [4])

The two low level attributes are move and role. move divides the gross individual motion into {inactive, active, walking, running}. The role, which specifies more intensional short-term behaviour such as browsing, walking and shop-entering/shop-exiting, provides an immediate indication of the type of activity. (The role extends to describing relationships between individuals during group interactions, eg, context fighting involves participants in with roles of fighter and victim.)

Whole-sequence behaviour is specified via a set of transition graph models, with some typical examples shown in Fig 2c. The type of node is called the situation and identifies individual frame activity *within the context of the overall behaviour* and is typically probabilistically related to the move and role determinations. The arrows show allowable transitions between the situation nodes for adjacent frames (including the explicitly shown self-transitions). Finally an x indicates that the given situation node *must* occur in order for the behaviour to occur. Without this we could, eg, match the *Idleness* graph in Fig 2a:iii against a sequence of move nodes, a result which is both intuitively wrong and creates ambiguity with the *walk* behaviour.

By providing these whole-sequence constraints the system can identify a whole sequence as, eg, browsing, even though the frames containing ‘looking into shop-windows’ might be a small fraction of the frames where the individual is walking between windows. The ‘true’ behaviour of an individual over the video sequence is (by fiat) described by only one graph: we refer to this choice as the context. Matching the sequence against the context graph requires associating a node in the transition graph with each frame (ensuring the graph topology is respected and additional constraints are met).

The graphs were finalised before sequences were annotated with ground truth. A typical set of images from one video sequence and the common data representation used for both the ground truth and analysis output is shown in Fig 2c,b. Consistency with the context graphs was enforced during markup by the labelling tool. However, the context graphs still leave many features that might be used for modelling unspecified, eg, module PROBCONTEXT estimates transition probabilities to enable HMM-based decoding.

This division into low- and high-level attributes is not completely clear-cut. The role can be relatively high level, e.g., browsing, but they are fundamentally discernable *using only short-term analysis*. In contrast, the situation describes a stage within a *specific* behaviour. Although we use this progression from move & role to situation and finally context in our analysis framework, it may not be necessary to get lower level attributes totally correct to get higher level ones correct: in particular within the correct context graph the precise frames where transitions between situation nodes occur may be highly ambiguous; however the overall match with the correct context may be clear.

**Modular behaviour analysis.** We want to break down the behaviour inference into three stages: estimating the move, estimating the role and estimating the situation & context. We also want to experiment with *hard assignment* and *probabilistic assignment* based estimation algorithms. This leads to the overall structure in Fig 3. The two inputs to each ‘pass-through switch’ node (⊗) are estimates of the same quantity (eg, the move values for the top right ⊗). In the full CAVIAR system the controller dynamically adjusts these switches based upon various performance feedback and other information. If instead we hardwire each switch node to pass through a given one of its inputs, we obtain eight distinct ‘systems’ which can produce the final output containing the estimates of all four attributes. (We

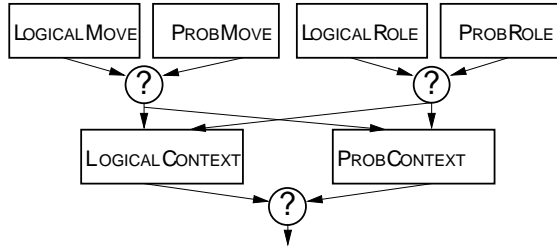


Figure 3: How various modules and implementations can be combined to generate a complete analysis in 8 distinct ways, where switches  $\textcircled{?}$  pass through a selected input

assume move and role estimation are independent.) Clearly we would like to know how accuracy varies between these ‘implementations’ of the system.

There are two further sources of variability. Firstly, different kinds of deployment can have visual scenes, different camera details, and varieties of behaviour attributes, eg, surveillance of a shopping centre atrium vs web-cam based workplace analysis. Secondly, each module has settable *module parameters* which affect its output. Ideally, the parameters would be ‘tuned’ to compensate for the first type of variability. However, good system tuning (particularly with limited ground truth) is non-trivial, so we want to understand how performance varies with the degree of tuning of component modules, especially since low-level module performance may not relate linearly to high-level performance.

If each move module has 1 parameter quantised into 16 values, we have  $16 \times 2$  ways of generating the move output. With the same assumptions on the role, a context module has  $(16 \times 2)^2$  unique inputs. Thus there are  $2 \times (16 \times 2)^2 = 2048$  systems to consider for optimal performance over our test data. We could use, eg, hill-climbing to attempt to find the maximum of this ‘black-box function’ without exhaustively mapping the space. However, because we want to understand the spectrum of behaviours rather than just obtain a single number we choose here to exhaustively evaluate the possibilities.

Our fundamental representation is to associate with each individual within each frame a set of *hypotheses* ( $\pi$ , move, role, situation, context), ie, a probability and values for the various attributes (some of which may be unset); this is allowed for by the XML representation in Fig 2 and is useful because hard assignment can be incorporated by represented by a single probability 1 entry, whilst input to a hard decision module takes the highest weighted hypothesis and resets its weight to 1. In practice we can take a few highest weighted entries without much error, reducing storage requirements. We now give *very brief* sketches of the estimation algorithms with more complete descriptions available in [13], using  $T_i$  to denote *module parameters* which are going to be systematically varied.

**Fitting moves and roles.** We define some common notation: the images contain multiple numbered individuals with the bounding box record for individual  $i$  in frame  $\tau$  denoted  $b_i[\tau]$ . The vector  $b_i[\tau].\mathbf{p}$  denotes the estimated ground plane position for box  $i$  at  $\tau$  and the number of the currently considered frame is always  $t$ . Finally, we take the current velocity (estimated over a fixed temporal window  $w$ ) as  $\mathbf{v} = b_i[t].\mathbf{p} - b_i[t-w].\mathbf{p}$ .

- **LOGICALMOVE.** This module only distinguishes between inactive/ active and walking/ running since simple rules were unable to distinguish further.

1.  $\text{move} \leftarrow$  if  $|\mathbf{v}| < T_1$  then  $\{(1/2, \text{inactive}), (1/2, \text{active})\}$  else  $\{(1/2, \text{walking}), (1/2, \text{running})\}$

- **LOGICALROLE.** In rules 2 & 3  $\text{signed\_vel} = \text{signum}(\mathbf{v} \cdot \mathbf{e})|\mathbf{v}|$ , where  $\mathbf{e}$  is a chosen vector along the direction out of the shop doorway.

1. if  $b_i[t'] \in \text{browse\_areas}$  for all  $t' \in [t - T_2, t]$  then  $\text{role} \leftarrow (1, \text{browse})$
2. if  $b_i[t].\mathbf{p} \in \text{shop\_doorway}$  and  $\text{signed\_vel} > 0$  then  $\text{role} \leftarrow (1, \text{shop-enter})$
3. if  $b_i[t].\mathbf{p} \in \text{shop\_doorway}$  and  $\text{signed\_vel} < 0$  then  $\text{role} \leftarrow (1, \text{shop-exit})$
4. if no other rule has triggered, then  $\text{role} \leftarrow (1, \text{walker})$

These rules were found by guessing a functional form from training data; the motivation was to obtain a simple, fast module to compare with PROBRROLE.

- **PROBMOVE & PROBRROLE.** We want to approximate the generating distribution of a class from a training set. The *Kernel Density Estimator (KDE)* [12] estimates the density of observation  $\mathbf{f}$  for class  $c$  from set of training points  $\{\mathbf{f}^c_i\}_{i=1}^N$  using

$$p(\mathbf{f}|c) = (1/N) \sum_{i=1}^N K(\mathbf{f}, \mathbf{f}_i^c; T_k) \quad (1)$$

$$K(\mathbf{x}, \mathbf{y}; T_k) = \max(1 - \|\mathbf{x} - \mathbf{y}\|^2 / T_k^2, 0) / Z \quad (2)$$

(where  $T_k$  is a parameter affecting the width of the local kernels and  $Z$  is a normalisation factor). Providing kernel  $K$  is roughly bell-shaped the exact form doesn't particularly affect the accuracy; we use the *Epanechnikov kernel* [12] which has optimal asymptotic performance, a simple form allowing efficient implementation and converges to the true distribution as  $N \rightarrow \infty$ ,  $T_k \rightarrow 0$ .

We use KDE for both the move and role. To obtain the feature vectors we apply a simple image-to-ground-plane transformation to the bounding boxes and then select a subset of effective values into an observation vector. In common with many other density estimators, KDE methods have two disadvantages: As they perform less implicit smoothing than other methods, eg GMMs, the estimate closely approximates the training set, even if this is non-representative. Secondly, the higher the dimension of the feature input space the more samples are needed for adequate estimates (*'the curse of dimensionality'*). We found that for testing we could divide the ground truth into training and testing sets containing equal numbers of sequences and use 4 parameters to form the feature vectors in Eq 1. We use the position of the centre of the box projected onto the ground plane,  $|\mathbf{v}|$  and  $\sqrt{|\text{change in box area}|}$  (both over a 32 frame window), applying empirical scale factors to each component to attain the same degree of variability over all axes.

**Fitting against context graphs.** As the ground truth satisfies the context models in Fig 2, we could find the correct context directly if we could assume *perfect* lower-level attributes. As this is unrealistic, we need matching techniques which are tolerant of errors. The first task is to convert lists of (move,role) pairs into situations (with probabilities). From the ground truth data we can tabulate (move, role, situation) triples to obtain function  $f: \text{move} \times \text{role} \times \text{situation} \rightarrow R$  giving the probability of a situation given the move and role. We can use this to extend the set of hypotheses  $\{(\pi_i, m_i, r_i)\}$  to a new set  $\{(\pi_i \times f(m_i, r_i, s_j), m_i, r_i, s_j)\}$  which now incorporates the situations. (In the case of logical inputs we discard all but the highest entry.) We now ignore all but the probability and situation when matching the sequence against the context graphs.

• **LOGICALCONTEXT.** Each context  $c$  is considered in turn against the situation list:

1. Run length compress the situations list wrt the situation node labels.
2. Unwind the graph representation into a tree and apply interpretation tree matching (with wild-cards allowed) technique to find the least cost match [6].
3. Take 'probability' of context  $c$ , with situations  $s_{0:N}$ , as  $\propto \exp(-\lambda(\#\text{wild card frames}))$ .

• **PROBCONTEXT.** We generate probabilities for each context  $c$  using observation probabilities  $\pi_{\tau, s_\tau}$  and situation transition probabilities  $p(s_i | s_{i-1})$  using a HMM algorithm:

1. For each final situation  $s_\tau = s$ : find most likely situation assignment  $s_0:s_\tau$  using the Viterbi recursion  $p(s_0:s_\tau) = \pi_{\tau, s_\tau} \max_{s_{\tau-1}} p(s_0:s_{\tau-1}) p(s_\tau | s_{\tau-1})$  (see [15]).
2. Choose most likely solution  $s_0:s_\tau$  also satisfying the constraints on must-visit nodes.
3. Take 'probability' of context  $c$ , with situations  $s_0:s_N$ , as  $\propto \prod_{\tau=0}^t \pi_{\tau, s_\tau} p(s_\tau | s_{\tau-1})$ .

## 4 Experiments

We used the CAVIAR system to run the behaviour interpretation algorithms over the 80 test sequences (using the module combinations shown in Fig 3) for a range of parameters for the move and role modules. Even with an optimised implementation, this took 37 hours. For space reasons we present a limited number of *preliminary, untuned* examples; more extensive data and analysis can now be found in [13]. (We abbreviate module names by initials, eg, PR is PROBRROLE.)

- **movement/role performance on France dataset.** Fig 4(a,b,d,e) show performance of the low-level modules as functions of parameters  $T_i$  for the entire France dataset. (The 'thresholding' nature of logical rules explains the 'steppiness' in a & b.) For (d) & (e) the solid line is proportional to the probability weight on the correct attributes and the dashed line is the proportion of times the highest probability attribute was correct. The graphs show that, except for low parameter values for the PM module the two measures are essentially the same.
- **situation/context performance wrt move/role.** Fig 4(c,f) show the situation output as function of the inputs to LM & LR and PM & PR and shows that, whilst smooth, the surface has many local maxima.
- **move/role performance on all datasets.** The top two rows of Fig 5 show the performance of LM, PM, LR & PR over the three datasets. Note all modules have different scores for the same parameter value in different datasets, and module PR clearly does not have a 'dataset-independent' optimal parameter value.
- **situation/context performance wrt move/role module parameters.** The bottom two rows of Fig 5 shows the correctness of the context (third row) and situation output of the PC module with respect to the parameters of the LR

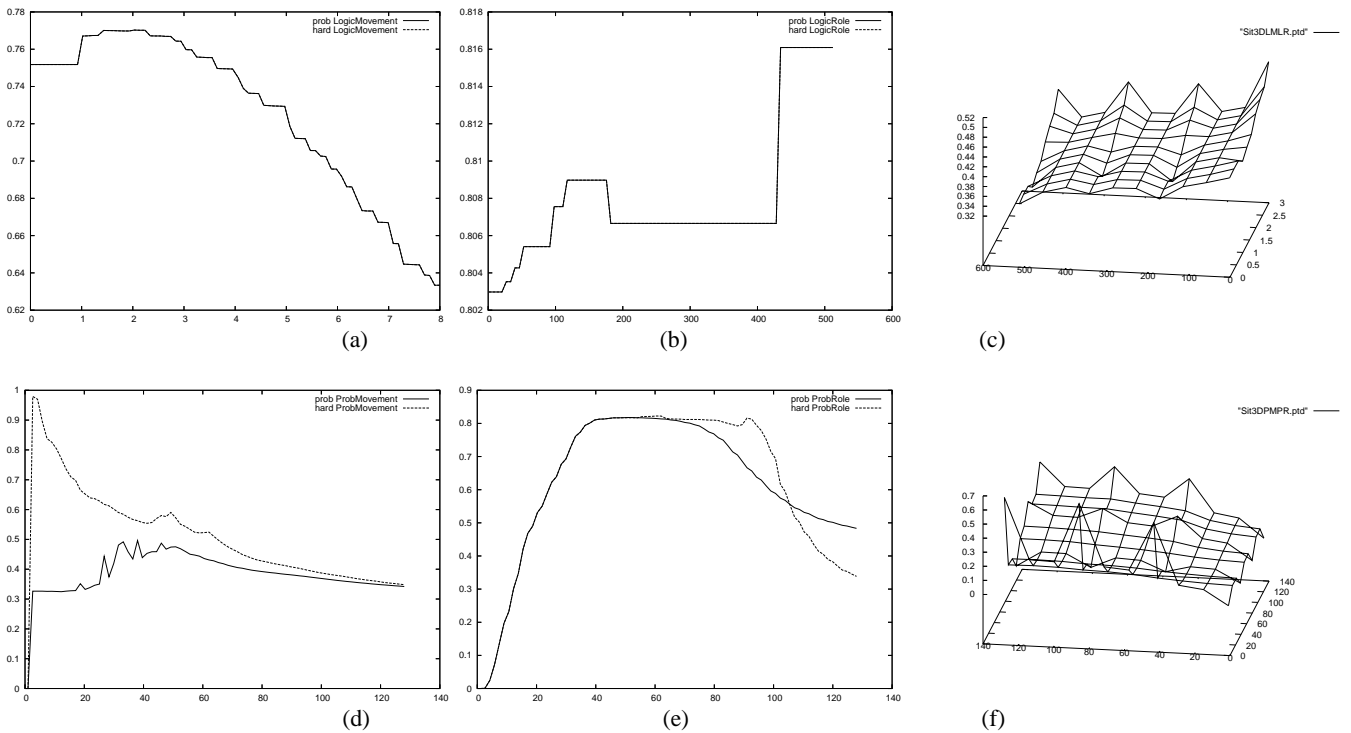


Figure 4: For the France dataset: (a) LM (with parameter  $T_1$ ); (b) LR (parameter  $T_2$ ); (c) situation correctness from LC as a function of  $T_1$  &  $T_2$ ; (d) PM (parameter  $T_3$ ); (e) PR (parameter  $T_4$ ); (f) situation correctness from PC module as a function of  $T_3$  &  $T_4$

& PR modules, giving an indication of how sensitive the overall system performance at the given role parameter value is to the setting of the move module parameter.

To explain the format, the first graph on the third row has a horizontal axis corresponds to the LR module's parameter value. For a complete network of modules from Fig 3 we also need a move module, so the solid line plots the correctness achieved on context when LR module is used in conjunction with the LM module, averaged over all possible parameter values for LM. The error bars show the standard deviation of this score wrt the LM parameters. Likewise, the dashed line and error bars shows performance when the PM module is used, again with the average and standard deviation wrt PMs parameter values. The plot shows that whilst *on average* the combination with LM gives better results than the PM, the standard deviation (uncertainty) for PM is so large that using a single parameter values for the move modules could lead to an erroneous plot suggesting PM is the better companion module. Indeed, there are large error bars on all four plots.

In general the performance is relatively poor, indicating the need for some fine-tuning of details (eg, better features for KDE); however these observations highlight the need to explore inter-module interactions when attempting to evaluate module performance.

## 5 Conclusions & further work

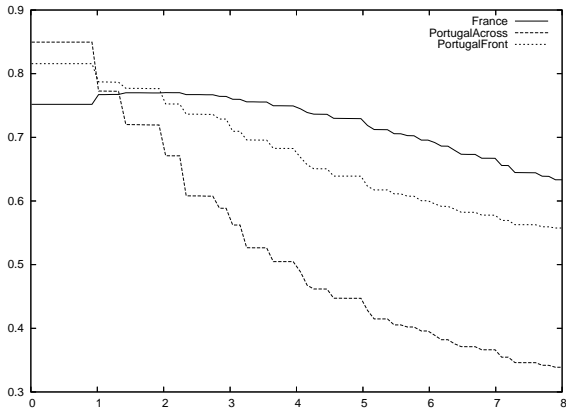
We developed a system to run empirical tests to gain understanding of the performance characteristics of a modular system for behaviour recognition, in particular relationships between correctness and the parameters of multiple modules; attempting to optimise module performance in isolation is can be confounded by interactions with other modules.

In future we hope to develop an intelligent sampling approach to make tractable analysis of much more complicated module ensembles. This will enable us to use the CAVIAR framework to characterise a wider set of vision algorithms from the vision community.

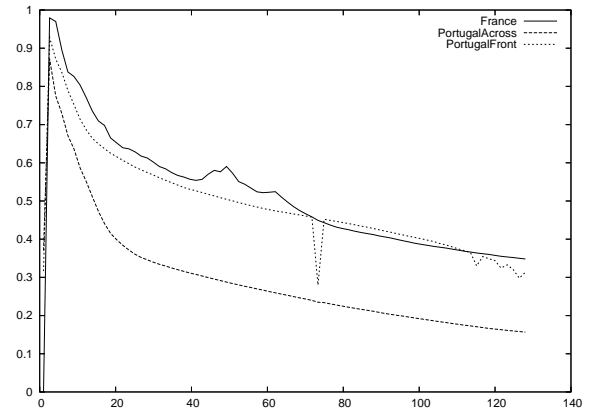
## References

- [1] The CAVIAR website [pages.inf.ed.ac.uk/rbf/CAVIAR/](http://pages.inf.ed.ac.uk/rbf/CAVIAR/).
- [2] J Bins, T List, R Fisher, and D Tweed. An intelligent and task-independent controller for video sequence analysis. In *IEEE Int. Workshop on Comp. Architecture for Machine Perception*, 2005.
- [3] P Courtney and N Thacker. *Imaging and Vision Systems: Theory, Assessment and Applications*, chapter Performance Characterisation in Computer Vision: The role of statistics in testing and design. NOVA Science Books, 2001.

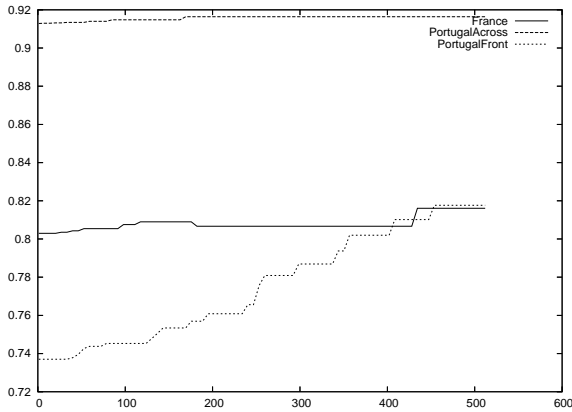
LM module results



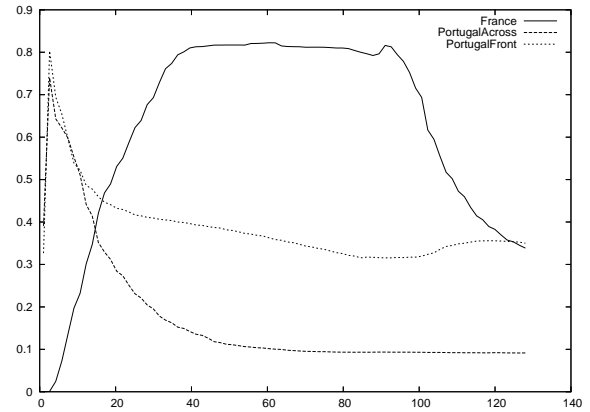
PM module results



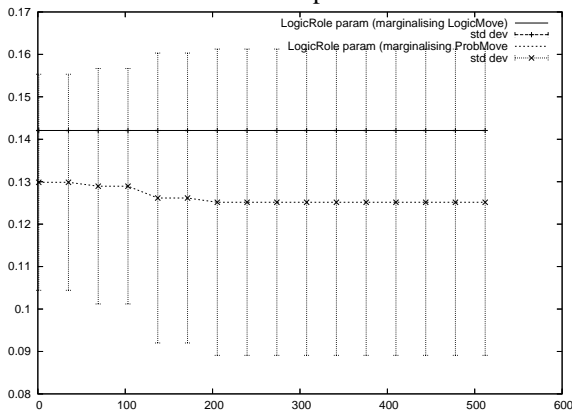
LR module results



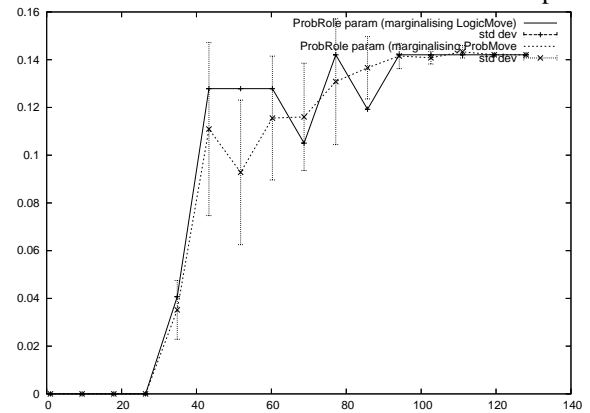
PR module results



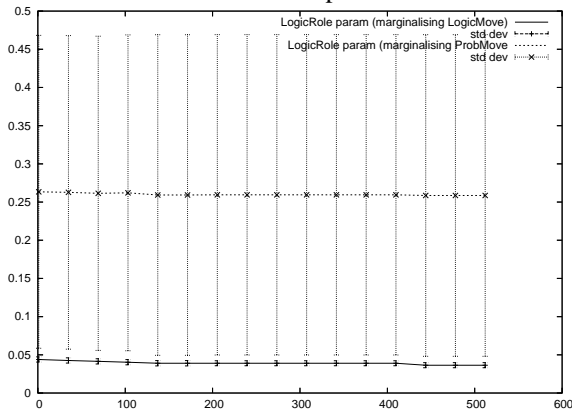
PC module context results vs PR param.



PC module context results vs PR param.



PC module situation results vs PR param.



PC module situation results vs PR param.

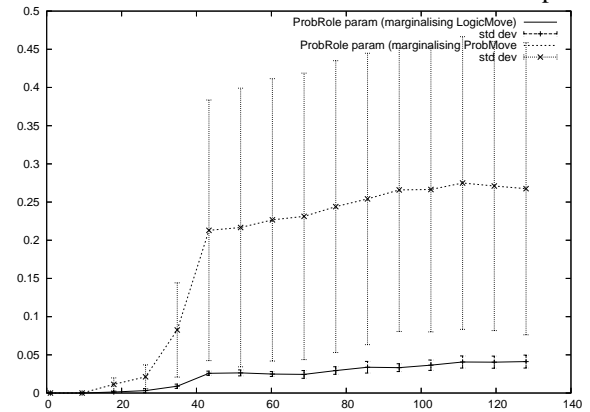


Figure 5: Top two rows: different relationships between parameters & performance of LM, PM, LR & PR for the datasets; bottom two rows: performance of context (3rd row) & situation (4th row) output from PC module as function of parameter for LR & PR modules.

- [4] J L Crowley and P Reignier. Dynamic composition of process federations for context aware preception of human activity. In *Int Conf on Integ of Intensive Multi-Agent Systems*, 2003.
- [5] M Everingham, H Muller, and B Thomas. Evaluating image segmentation algorithms using the Pareto Front. In *Proc. ECCV*, 2002.
- [6] W Grimson and T Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans on PAMI*, 9(4):469–482, 1987.
- [7] B Draper J R Beveridge, K She and G Givens. Parametric and nonparameric methods for the statistical evaluation of human ID algorithm. In *IEEE Workshop on Empirical Evaluation Methods in Computer Vision*, 2001.
- [8] T List and R Fisher. Computer vision markup language. In *Proc ICPR*, pages 789–792, 2004.
- [9] R Micheals and T Boulton. Effi cient evaluation of classifi cation and recognition sytems. In *IEEE Conf. Comp. Vis. and Pattern Recog.*, 2001.
- [10] G Rees, W Wright, and P Greenway. ROC method for evaluation of multi-class segmentation/classifi cation algorithms with infrared imagery. In *Proc. BMVC*, pages 537–546, 2002.
- [11] T Riopka and T Boulton. The eyes have it. In *ACM Workshop on Biometric Methods & Applications*, 2003.
- [12] B W Silverman. *Density Estimation*. NY Chapman and Hall, 1986.
- [13] D Tweed and R Fisher. Exploring behaviour recognition strategies within CAVIAR. Technical report, Edinburgh University, 2005.
- [14] M Valera and S A Velastin. Intelligent distributed surveillance systems: A review”, special issue on intelligent distributed surveillance systems. 2005.
- [15] A J Viterbi. Convolutional codes and ther performance in communication systems. *IEEE Trans on Comm. Tech.*, 13(2):260–269, 1967.
- [16] J Vogel and B Schiele. On performance characterisation and optimization for image retrieval. In *Proc. ECCV*, 2002.