

# AVINSOR

(A Visually Intelligent Navigation System for Application in Robotics)

Tej Birring

Email

gsb4@kent.ac.uk

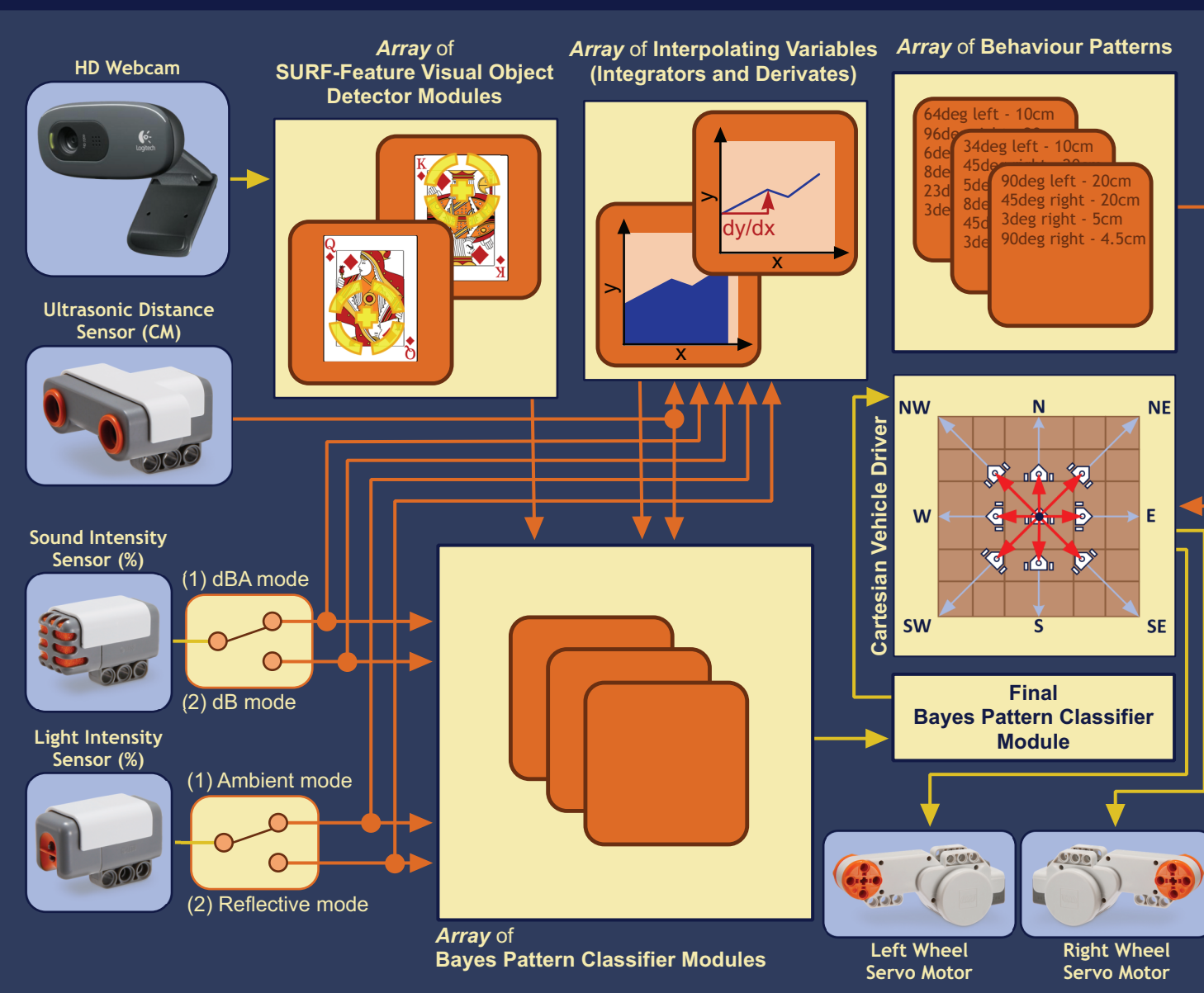
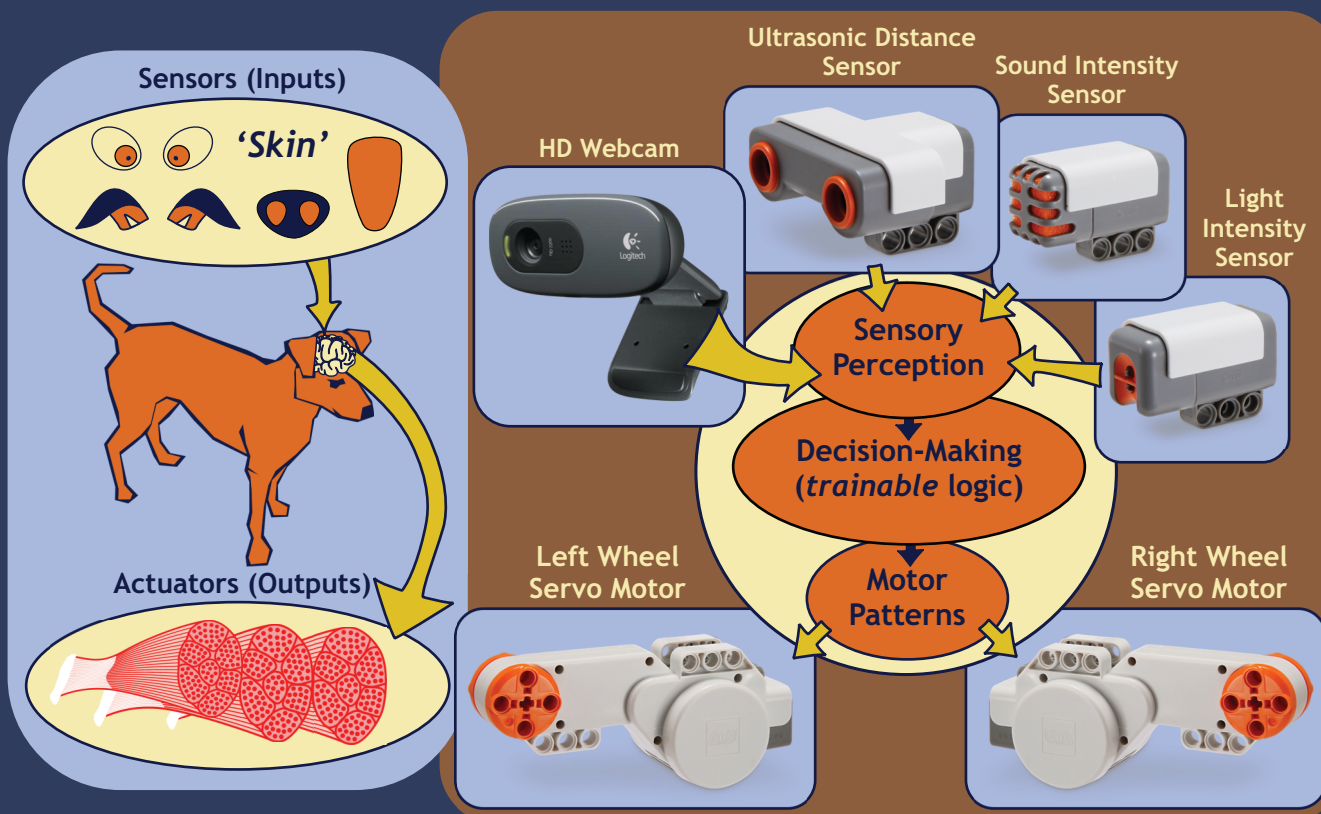
Supervisor

Dr. Peter Lee

The (philosophical) aim of the project was to allow the ability to "train a robot as one would a pet dog, or rather, an obedient pet dog." (for the purposes of exploratory navigation).

The AVINSOR concept was formulated on the foundation established by the preliminary 6-week research phase *briefly* covering a range of different fields including (1) *physiology of vision*, (2) *the psychology of visual perception*, (3) *simulation of (psychological) cognition models*, (4) *machine learning and pattern-classification methods*, and (5) *methods of accurate steering in miniature ground vehicle robots*.

The objective of the *initial* prototype was to *establish a hardware/software framework* such that would allow a LEGO® Mindstorms® NXT based robot assembly to demonstrate '*user-trainable*' interactive movement.



## DELIVERABLES

AVINSOR Library for NXT

Client Application

Robot Assembly

Ask for a demonstration!

Left: A non-technical diagram illustrating the data flow concept of the system. Inputs from the various sensors eventually lead to a number of software components - which ultimately function to *emulate learning* and *produce output signals* for the two servo motors. Note that the *orange lines* represent user arrangement of inputs into the user-configurable modules such as 'Bayes Pattern Classifier Modules', 'Interpolating Variable (Modules)', 'SURF-feature Object Detection Modules', and the 'Behaviour Patterns Memory'.

## FEATURES

*Modularly-structured library (and client application).* Written in C#, utilizes *MATLAB® COM Automation Server* for *background processing*.

*Intelligent SURF-feature visual object detector module associates object detection success with likelihood probabilities;* useful for making autonomous decisions via 'reinforced learning'.

Multivariate, multicategory, Bayes pattern classification operations are *encapsulated* in a *MATLAB® class*. Complementary C# class for *easy integration* in application(s).

*Cartesian Vehicle Driver* converts (hardware or software generated) cartesian input coordinates into *motor output signals* - allowing *tachometric control* of both wheels to control vehicle's position.

## USER-DEFINED INPUTS VARIABLES

Apart from system inputs from any NXT robot(s) that may be connected\*, the AVINSOR software framework allows users to define their own input variables into the Bayes Pattern Classifiers they create, configure and arrange to form a *cognition system*. Such input variables include *SURF-Feature Visual Object Detector Modules* that operate by detecting and comparing SURF features between a (recently captured) *test frame* and that of a *reference frame* (captured earlier) - the module is designed to be trained so that it can accurately indicate a *successful (on unsuccessful) match* (finding the particular object it is created to detect).

Another type of input variable that can be created is an *interpolating variable* (i.e. a *numeric differentiator* or *numeric integrator*). Differentiators are useful at *deducing rates of change between variables*, and integrators can be used to *classify quadratures of variables*. Interpolating variables can also be configured to take the average of the last 'n' number of samples before producing an output (for performance).

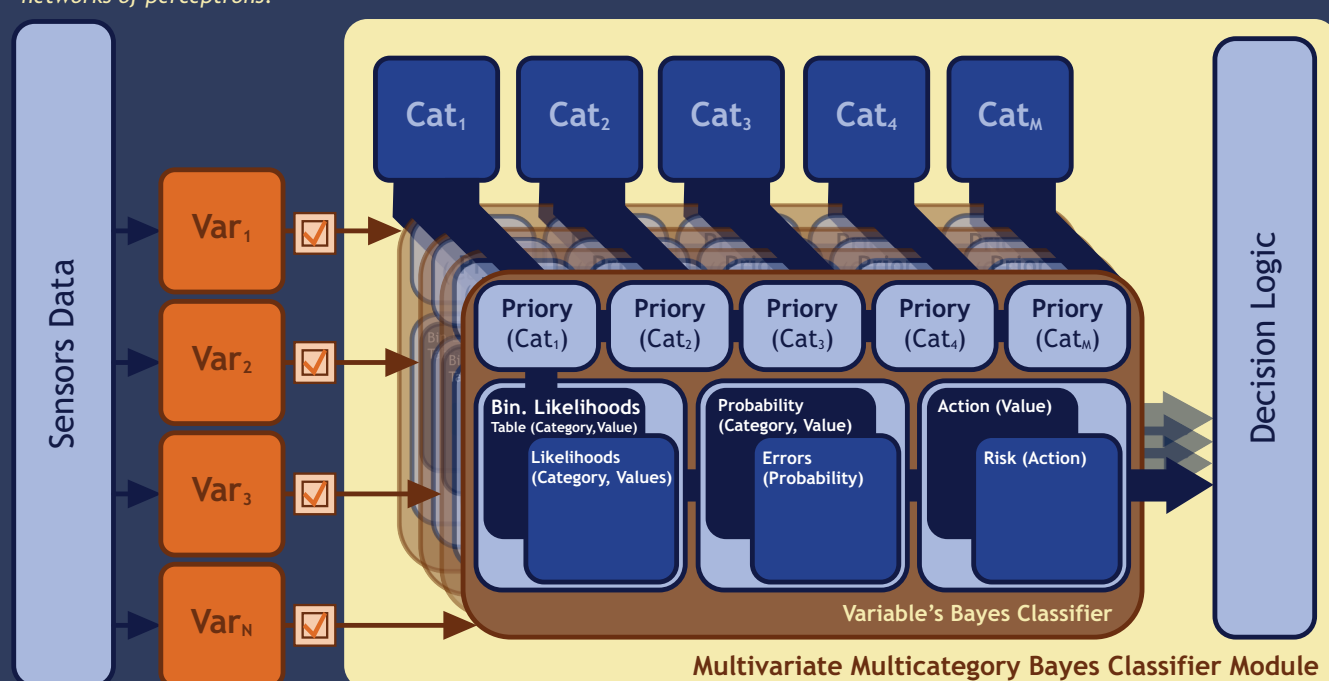
\* Consideration of the use of more than one robot in an application was taken into account when designing the library.

## WHY SURF?

Speeded-Up Robust Features is a scale-invariant feature detection method i.e. its ability to discern between (visual) objects is not affected by the change in scale of the object. Hence a SURF-Feature Visual Object Detection Module was designed. An input frame's feature pattern's are compared against feature pattern's of the reference image and classified.

## BAYES PATTERN CLASSIFIER MODULE

This software module allows multiple input streams of data (let's call these the *variables*) to select an output from a finite set (the *categories*). The outputs are dependant on matrices of probabilities which can be manipulated (through GUI abstraction) to adjust how the module classifies patterns of the values of the *input variables* that it has been associated with. Allowing (1) trainability through a GUI (interactive graphs), and (2) the ability to freely create, configure and arrange these objects (to allow the output of any module to become an input of any other) makes this feature seem like a compromise between *fuzzy logic* and *networks of perceptrons*.



Multivariate Multicategory Bayes Classifier Module