<u>DESIGN OF CONTEXT-AWARE SYSTEM FOR VECHILE</u>

INTRODUCTION

When we humans interact with other persons and the surrounding environment we make use of implicit situational information. We can intuitively deduce and interpret the context of the current situation and react appropriately. For example, a person discussing with another person automatically observes the gestures and voice tone of the other party and reacts in an appropriate manner.

Computers are not as good as humans in deducing situational information from their environment and in using it in interactions. They cannot easily take advantage of such information in a transparent way, and if they can they usually require that it is explicitly provided. This is a challenge for human computer interaction. For example, present user interfaces still seldom can sense and adapt automatically to current light and noise level without the user providing the information. Another example comes from the area of mobile computing. Would it not be nice, if one could obtain services and information according to the current location and activity? For example, if at a stadium watching a football game, one could obtain additional information about the players, be able to participate in a local betting game, and check the traffic situation outside the stadium. There are very many different ways how context information could be used to make computer systems and applications more user-friendly, flexible, and adaptable.

The number of devices and services in vehicles has rapidly increased during the last decades. Nowadays it is possible to make a phone call, activate cruise control, manipulate a navigation system and for many other things while driving. The number of controls for steering the devices has increased concurrently. This leads to two problems. The first, and most serious, is the safety issue. The driver spends more and more time looking at menus and pushing buttons which occupies both hands and eyes and takes the attention from the primary task - of driving. The second issue is a matter of space; the dash board is a limited area and there simply is not enough room for all the controls. Car manufacturers are struggling with multi functional control solutions and the design problem that comes with them; the controls have to be sophisticated enough to control several devices and yet simple enough to be intuitive to manage, preferably without even having to look at them. The use of context-aware system is especially important in improving the safety of driver behaviour.

Use of Context

In human-human interaction, a great deal of information is conveyed without explicit communication. Gestures, facial expressions, relationship to other people and objects in the vicinity, and shared histories are all used as cues to assist in understanding the explicit communication[2]. These shared cues, or *contex*t, help to facilitate grounding between participants in an interaction. The use of context allows an application to be tailored to a user's specific situation , providing increased benefits to the user.

Context

Dey and Abowd refer to **context[7]** as “any information that can be used to characterize the situation of entities(i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.”

Context-aware

Context-awareness means that one is able to use context information. A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use. The challenge for such systems lies in the complexity of capturing, representing and processing contextual data. To capture context information generally some additional sensors and/or programs are required. To transfer the context information to applications and for different applications to be able to use the same context information a common representation format for such information should exist. In addition to being able to obtain the context-information, applications must include some "intelligence" to process the information and to deduce the meaning. This is probably the most challenging issue, since context is often indirect or deducible by combining different pieces of context information. E.g. if three persons meet in a certain office room at a certain time, it can mean that it is the weekly strategy meeting.

Need of Context-aware in vehicles

The increasing use of mobile phones in vehicles and reports that talking on the phone will distract the driver has led to an increasing concern that speech technology will distract the driver even more than manual controls[2]. However, recent studies point in a different direction. A study carried out by Esbjörnson et al. reveals that drivers adapt their conversational behaviour to the driving situation by giving implicit and explicit signals to announce that they need to pause the dialogue in order to concentrate on the traffic. Another study compared a context-aware dialogue system to a context-unaware system to study the effects on dialogue behaviour. Context awareness made it possible to start a new task without having to repeat earlier agreements, e.g. the driver can ask for directions to a city and then ask “what is the weather like there” without having to repeat the name of the city. The context awareness reduced the degree of user distraction; the drivers could maintain car speed in much higher degree compared to a context-unaware system, it also took less time and less user turns to complete the given task.

OVERVIEW OF THE PROPOSED SYSTEM

Our approach consists in observing the driver in his/her real driving condition with sensor technology. The observation is a learning process that can improve the prediction capability. We use Fuzzy logic approach as a form of uncertain reasoning from observatioh ns. The proposed Fuzzy approach will be implemented based on Mamdani fuzzy interference systems.

Driver's cognitive concepts, such as risks, are deduced from various sensors such as the dynamics of the vehicle in a certain situation or physiological measures. Observations are technically possible due to the advent of sophisticated in-vehicle sensors and context aware systems which can gather and analyse data about (i) the physiological state of the driver, (ii) the behaviour of the driver (iii) the dynamics of the vehicle and (iv) the description of the environment surrounding the vehicle and the driver.

Fuzzy logic is a convenient way to map an input space to an output space. The Fuzzy Logic Toolbox for use with MATLAB is a tool for solving problems with fuzzy logic.

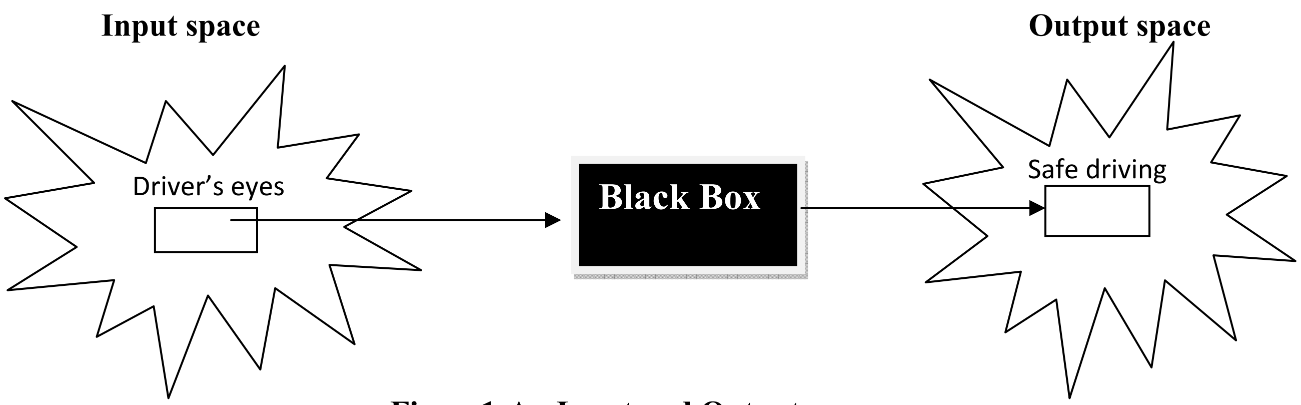


Figure1:An Input and Output map

Why Fuzzy logic?

- Fuzzy logic is conceptually easy to understand.

The mathematical concepts behind fuzzy reasoning are very simple. What makes fuzzy nice is the "naturalness" of its approach and not its far-reaching complexity.

- Fuzzy logic is flexible.

With any given system, it's easy to massage it or layer more functionality on top of it withoutstarting again from scratch.

- Fuzzy logic is tolerant of imprecise data.

Everything is imprecise if you look closely enough, but more than that, most things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tacking it onto the end.

• Fuzzy logic can be built on top of the experience of experts.
In direct contrast to neural networks, which take training data and generate opaque, impenetrable models, fuzzy logic lets you stand on the shoulders of people who already understand your system.

Why Fuzzy Logic for this project?

Context aware will have dynamic nature so to analyse the dynamic nature we use fuzzy Logic.

We cannot use Hidden Markov Model(Probabilistic scheme) in this because one particular aspect of a HMM is the presence of transition from one hidden state to another. This aspect could be very useful to predict human behaviour or a sequence of activities. However, to determine a single complex activity by considering various correlated observable aspects (primitive contexts which can be captured by employing sensors), an HMM may not be optimal.

•Fuzzy sets describe vague concepts (fast runner, hot weather, weekend days)

•A fuzzy set admits the possibility of partial membership in it (Friday is sort of a weekend day, the weather is rather hot)

•The degree an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. (Friday is a weekend day to the degree 0.8)

•A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

The point of fuzzy logic is to map an input space to an output space, and the primary mechanism for doing this is a list of if-then statements called rules. All rules are evaluated in parallel, and the order of the rules is unimportant. The rules themselves are useful because they refer to variables and the adjectives that describe those variables. Before we can build a system that interprets rules, we have to define all the terms we plan on using and the adjectives that describe them. If we want to talk about how hot the water is, we need to define the range that the water's temperature can be expected to vary over as well as what we mean by the word hot.

Interpreting if-then rules:

1 Fuzzify inputs

Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. If there is only one part to the antecedent, this is the degree of support for the rule.

2 Apply fuzzy operator

If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.

3 Apply implication method

Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. If the antecedent is only partially true, then the output fuzzy set is truncated according to the implication method.

In general, one rule by itself doesn't do much good. What's needed are two or more rules that can play off one another. The output of each rule is a fuzzy set, but in general we want the output for an entire collection of rules to be a single number. How are all these fuzzy sets distilled into a single crisp result for the output variable? First the output fuzzy sets for each rule are *aggregated* into a single output fuzzy set. Then the resulting set is *defuzzified*, or resolved to a single number. The next section shows how whole process works from beginning to end.

DESIGN OF THE PROPOSED SYSTEM

Information flows from left to right, from three inputs to a single output. The parallel nature of the rules is one of the more important aspects of fuzzy logic systems[5].

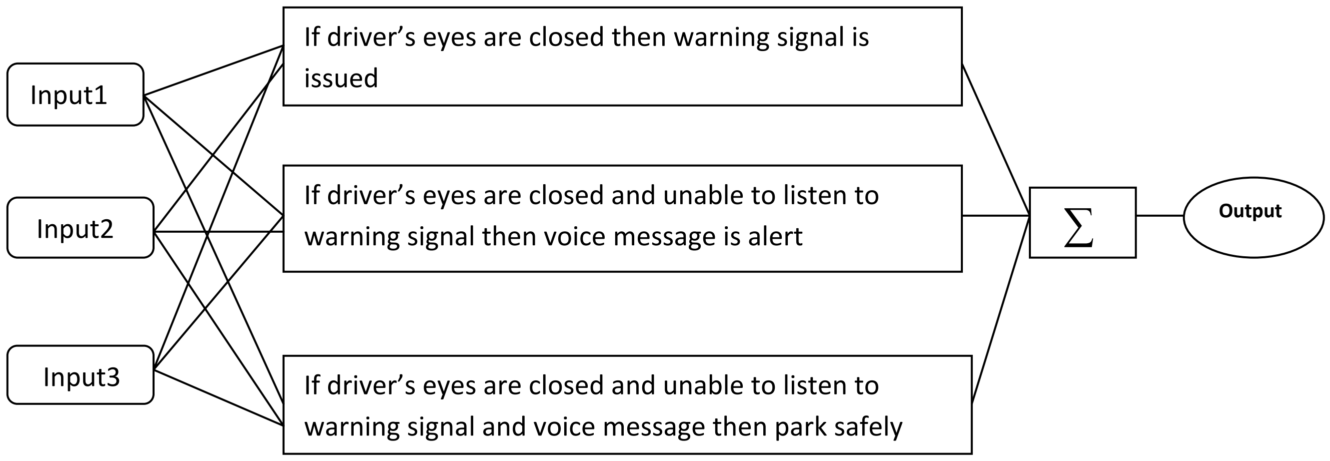

Input1
Input2
Input3
If driver's eyes are closed then warning signal is issued
If driver's eyes are closed and unable to listen to warning signal then voice message is alert
If driver's eyes are closed and unable to listen to warning signal and voice message then park safely
∑
Output

Figure2:Fuzzy Logic rule system

In the Fuzzy Logic Toolbox, there are five parts of the fuzzy inference process: fuzzification of the input variables, application of the fuzzy operator (AND or OR) in the antecedent, implication from the antecedent to the consequent, aggregation of the consequents across the rules, and defuzzification. These sometimes cryptic and odd names have very specific meaning that we'll define carefully as we step through each of them in more detail below.

Step 1. Fuzzify Inputs

The first step is to take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. The input is always a crisp numerical value limited to the universe of discourse of the input variable (in this case the interval between 0 and 10) and the output is a fuzzy degree of membership (always the interval between 0 and 1). So fuzzification really doesn't amount to anything more than table lookup or function evaluation.

Step 2. Apply Fuzzy Operator

Once the inputs have been fuzzified, we know the degree to which each part of the antecedent has been satisfied for each rule. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one number that represents the result of the antecedent for that rule. This number will then be applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value.

Step 3. Apply Implication Method

The implication method is defined as the shaping of the consequent (a fuzzy set) based on the antecedent (a single number). The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication occurs for each rule. Two built-in methods are supported, and they are the same functions that are used by the AND method: *min* (minimum) which truncates the output fuzzy set, and *prod* (product) which scales the output fuzzy set.

Step 4. Aggregate All Outputs

Aggregation is when we unify the outputs of each rule by joining the parallel threads. It's just a matter of taking all the fuzzy sets that represent the output of each rule and combining them into a single fuzzy set in preparation for the fifth and final step, defuzzification. Aggregation only occurs once for each output variable. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable.

Step 5. Defuzzify

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number—crispness recovered from fuzziness at last. As much as fuzziness helps the rule evaluation during the intermediate steps, the final output for each variable is generally a single crisp number. So, given a fuzzy set that encompasses a range of output values, we need to return one number, thereby moving from a fuzzy set to a crisp output.

Building Systems with the Fuzzy Logic Toolbox

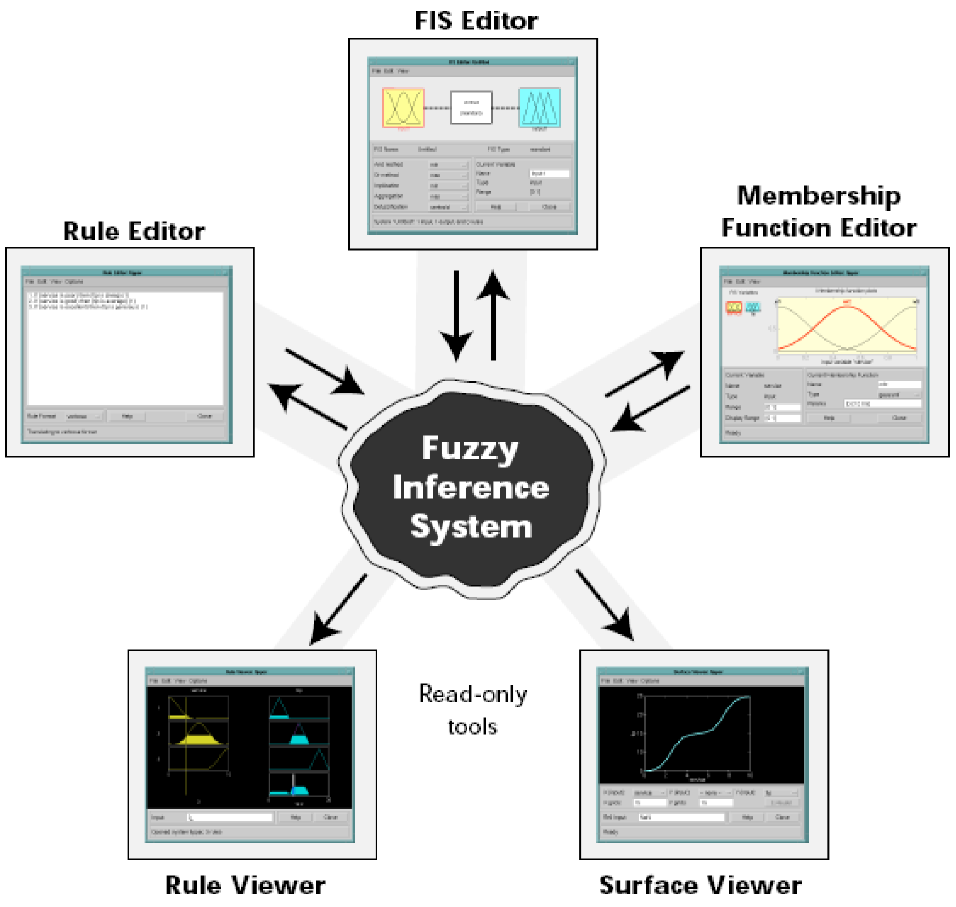


Figure3:Fuzzy Logic ToolBox

The FIS Editor handles the high level issues for the system: How many input and output variables? What are their names? The Membership Function Editor is used to define the shapes of all the membership functions associated with each variable. The Rule Editor is for editing the list of rules that defines the behavior of the system. The last two GUIs are used for looking at, as opposed to editing, the FIS. They are strictly read-only tools. The Rule Viewer is a MATLAB-based display of the fuzzy inference diagram shown at the end of the last section. Used as a diagnostic, it can show (for example) which rules are active, or how individual membership function shapes are influencing the results. It's a very powerful window full of information. The last of the five GUI siblings is the Surface Viewer. This tool can display how one of the outputs depends on any one or two of the inputs—that is, it generates and plots an output surface map for the system. Some of the GUI tools have the potential to influence the others. For example, if you add a rule, you can expect to see the output surface change.

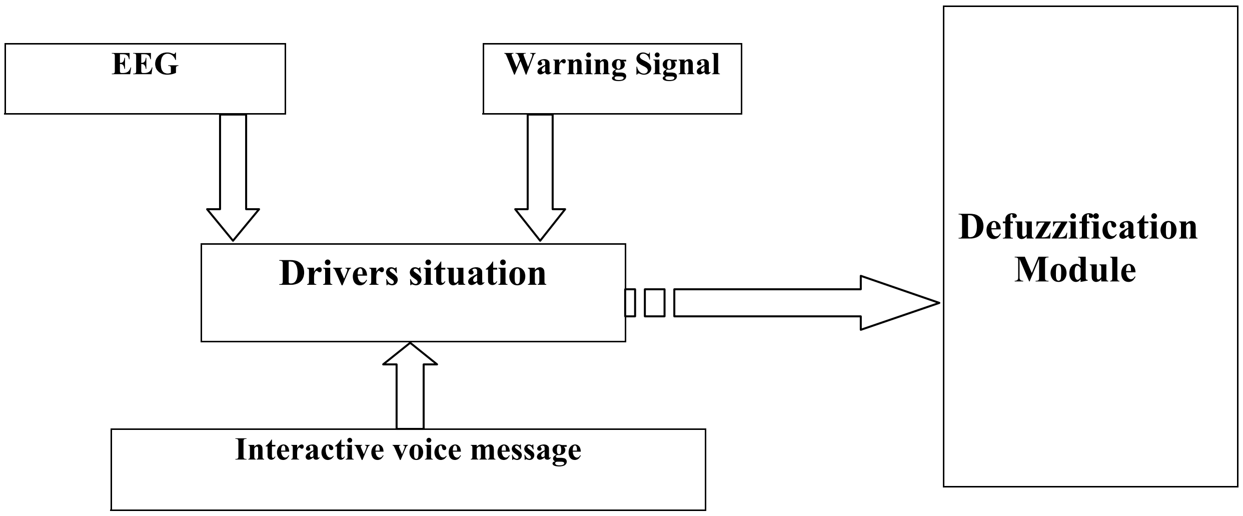


Figure4:Observing Driver Behaviour

IMPLEMENTATION AND TESTING

This project mainly focuses on “Drowsy Driver Detection System using Fuzzy Logic”. When driver is in drowsy state then system must give some indication about his sleepiness either by alarm or voice message. If the driver still is not aware about his unconscious state then the system must the vehicle that is car to a safe side or it must halt it slowly. This can be implemented using “Fuzzy Logic”.

Inputs for detecting the drowsy state of a driver would be continous eyes closed,warning signal,voice message. Output will be warning signal, interactive voice message, and safe parking. Liguistic states are blinking,raised,safe parking.Since we are using 1 input and 3 Linguistic states we can frame atmost three rules.So here we frame 3 rules and the rules are as follows:

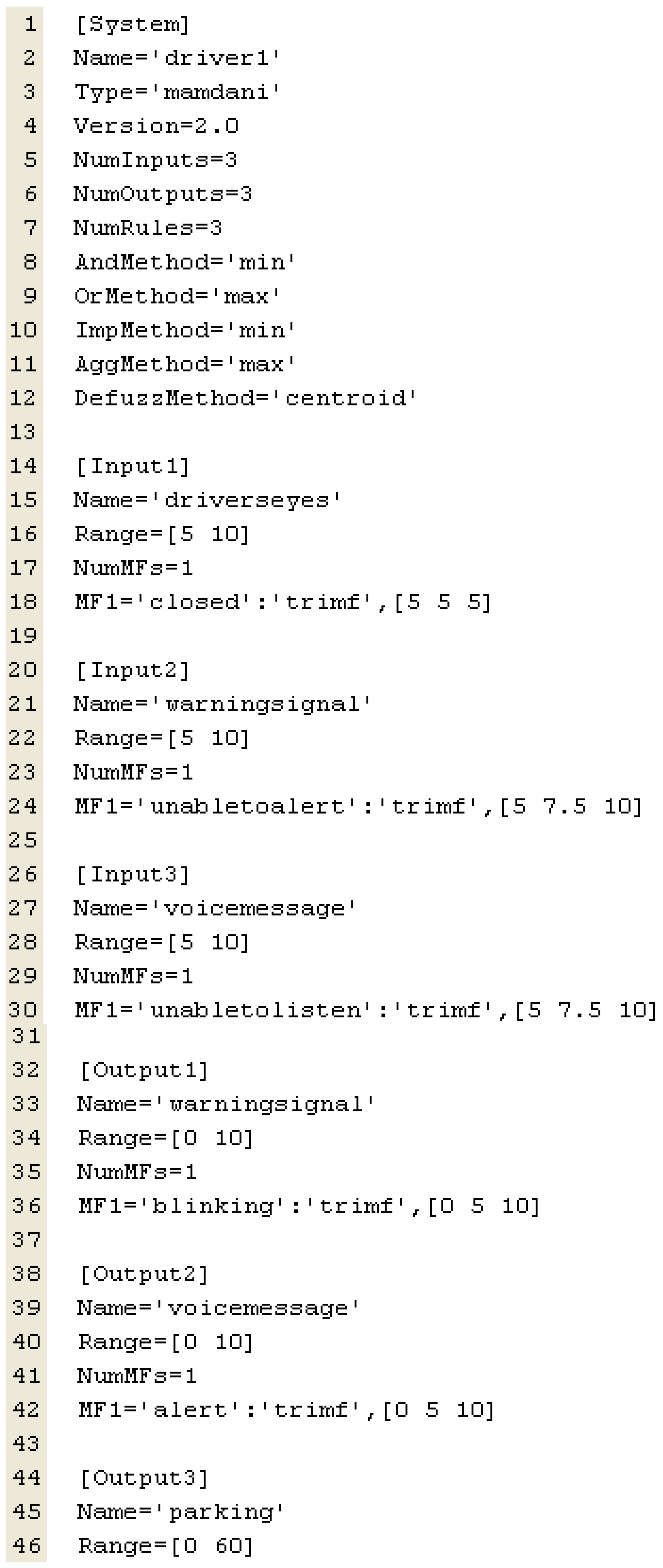
1.If the driver’s eyes are closed then warning signal is issued.

2. If the driver’s eyes are closed and unable to listen warning signal then voice message is alert.

3. If the driver’s eyes are closed and unable to listen warning signal and voice message then park it to a safe side.

Now we implement these rules using Fuzzy Logic Toolbox in Matlab.
For each and every rule we get one output fuzzy set.All these rules are aggregated so the resulting set is defuzzified or resolved to a single number.

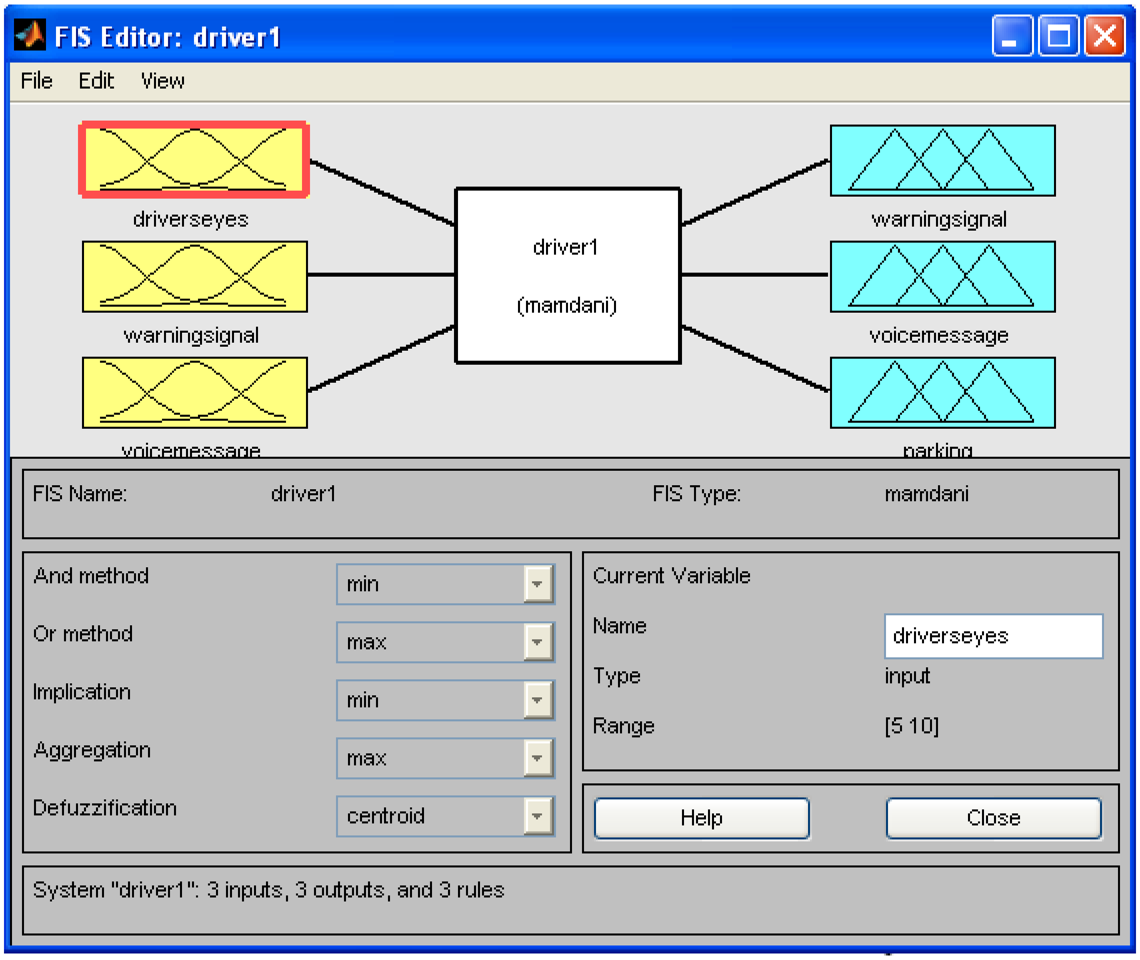
The code for drowsy driver system is given below



```
[System]
Name='driver1'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=3
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='driverseyes'
Range=[5 10]
NumMFs=1
MF1='closed':'trimf',[5 5 5]

[Input2]
Name='warningsignal'
Range=[5 10]
NumMFs=1
MF1='unabletoalert':'trimf',[5 7.5 10]

[Input3]
Name='voicemessage'
Range=[5 10]
NumMFs=1
MF1='unabletolisten':'trimf',[5 7.5 10]

[Output1]
Name='warningsignal'
Range=[0 10]
NumMFs=1
MF1='blinking':'trimf',[0 5 10]

[Output2]
Name='voicemessage'
Range=[0 10]
NumMFs=1
MF1='alert':'trimf',[0 5 10]

[Output3]
Name='parking'
Range=[0 60]
```

```
NumMFs=1
MF1='parkingsafely':'trimf',[0 30 60]

[Rules]
1 0 0, 1 0 0 (1) : 1
1 1 0, 0 1 0 (1) : 1
1 1 1, 0 0 1 (1) : 1
```

FIS Editor



FIS Name: driver1 FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

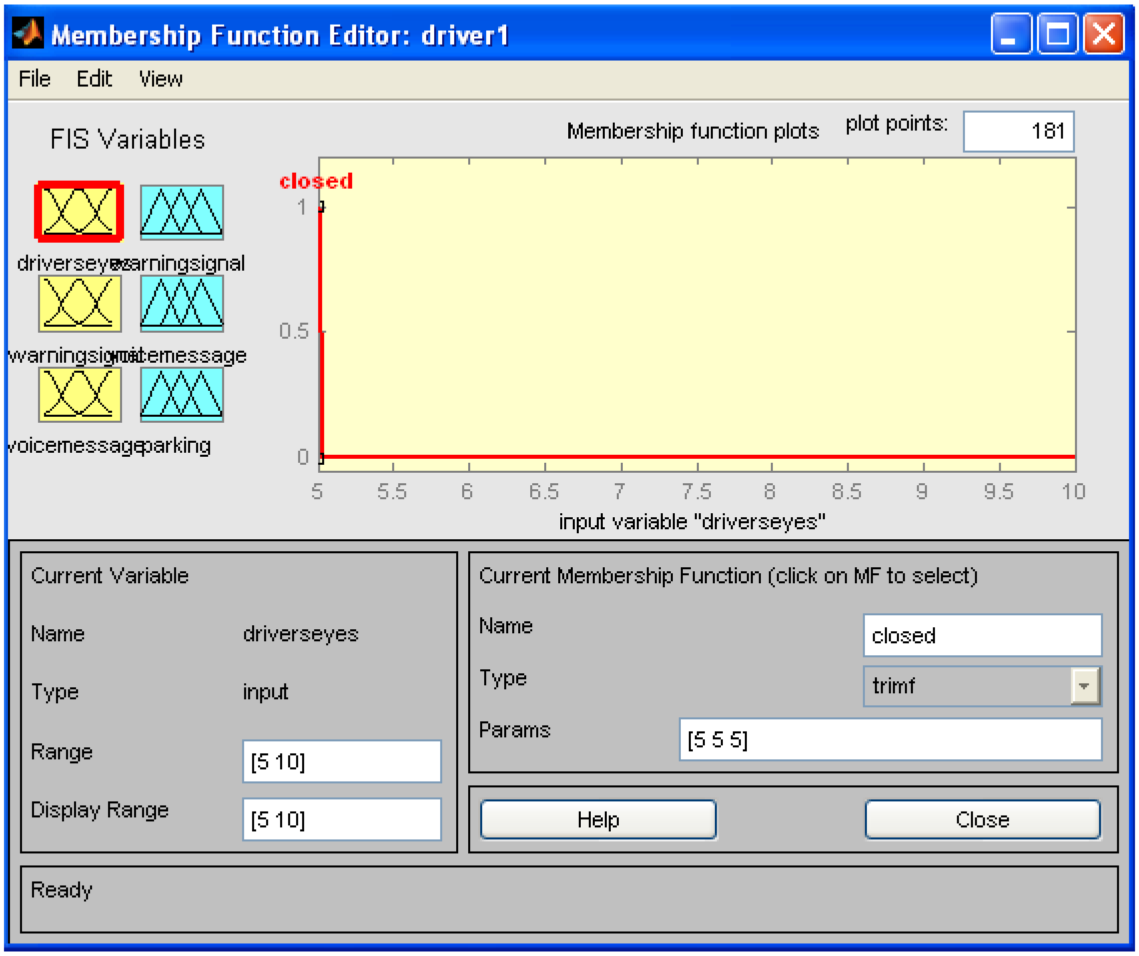
Current Variable
Name: driverseyes
Type: input
Range: [5 10]

System "driver1": 3 inputs, 3 outputs, and 3 rules

This GUI tool allows you to edit the highest level features of the fuzzy inference system, such as the number of input and output variables, the defuzzification method used, and so on. Refer to Chapter 2, Tutorial, for more information about how to use fuzzy. The FIS Editor is the high level display for any fuzzy logic inference system. It allows you to call the various other editors to operate on the system. This interface allows convenient access to all other editors with an emphasis on maximum flexibility for interaction with the fuzzy system[5].

The diagram displayed at the top of the window shows the inputs, outputs, and a central fuzzy rule processor. Click on one of the variable boxes to make the selected box the current variable. You should see the box highlighted in red. Double-click on one of the variables to bring up the Membership Function Editor. Double-click on the fuzzy rule processor to bring up the Rule Editor. If a variable exists but is not mentioned in the rule base, it is connected to the rule processor block with a dashed rather than a solid line[5].

Membership Fuction Editor



Current Variable

Name driverseyes

Type input

Range [5 10]

Display Range [5 10]

Current Membership Function (click on MF to select)

Name closed

Type trimf

Params [5 5 5]

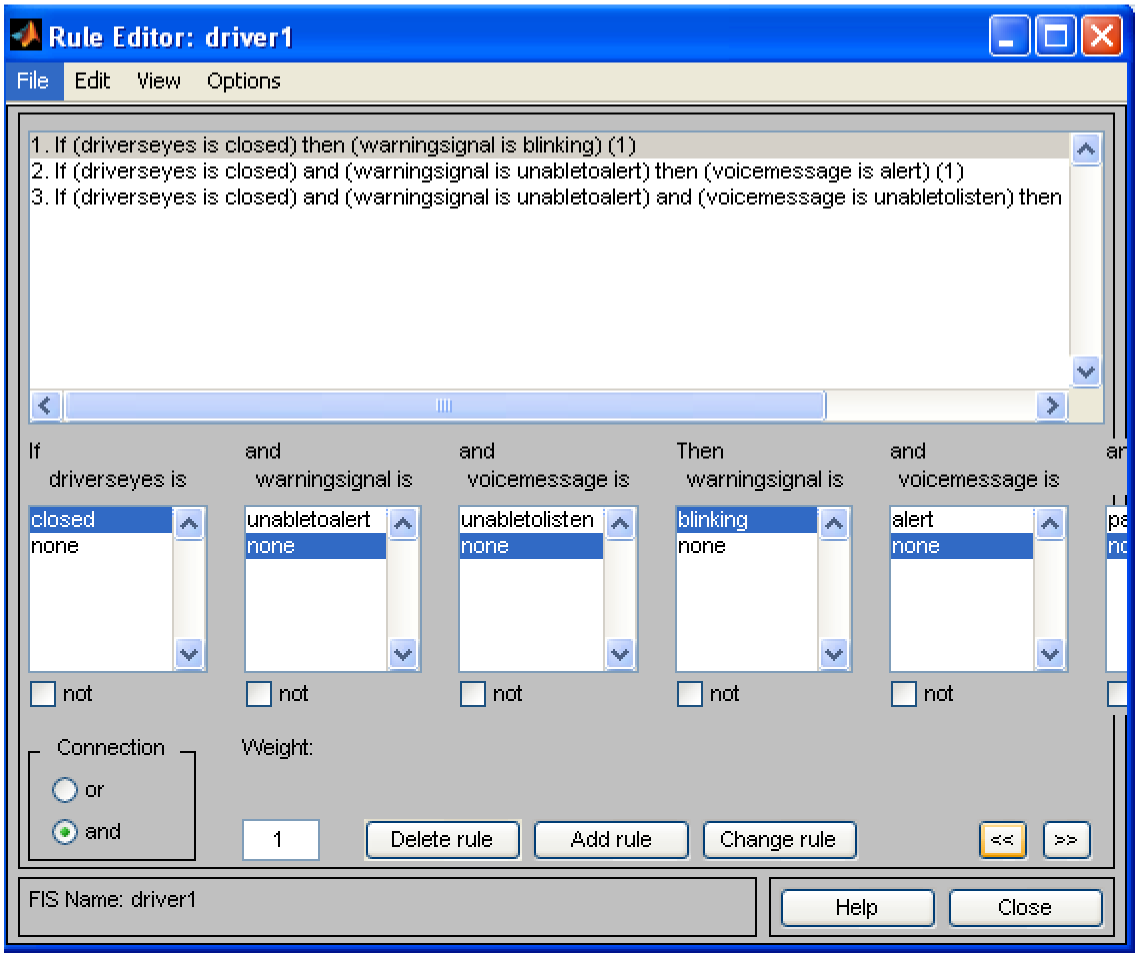
Ready

The Membership Function Editor shares some features with the FIS Editor. In fact, all of the five basic GUI tools have similar menu options, status lines, and Help and Close buttons. The Membership Function Editor is the tool that lets you display and edit all of the membership functions associated with all of the input and output variables for the entire fuzzy inference system.

When you open the Membership Function Editor to work on a fuzzy inference system that does not already exist in the workspace, there are not yet any membership functions associated with the variables that you have just defined with the FIS Editor[5].

On the upper left side of the graph area in the Membership Function Editor is a "Variable Palette" that lets you set the membership functions for a given variable.To set up your membership functions associated with an input or an output variable for the FIS, select an FIS variable in this region by clicking on it.

Rule Editor


Rule Editor: driver1

```
1. If (driverseyes is closed) then (warningsignal is blinking) (1)
2. If (driverseyes is closed) and (warningsignal is unabletoalert) then (voicemessage is alert) (1)
3. If (driverseyes is closed) and (warningsignal is unabletoalert) and (voicemessage is unabletolisten) then
```

If driverseyes is	and warningsignal is	and voicemessage is	Then warningsignal is	and voicemessage is
closed	unabletoalert	unabletolisten	blinking	alert
none	none	none	none	none

Connection: or [✓ and]

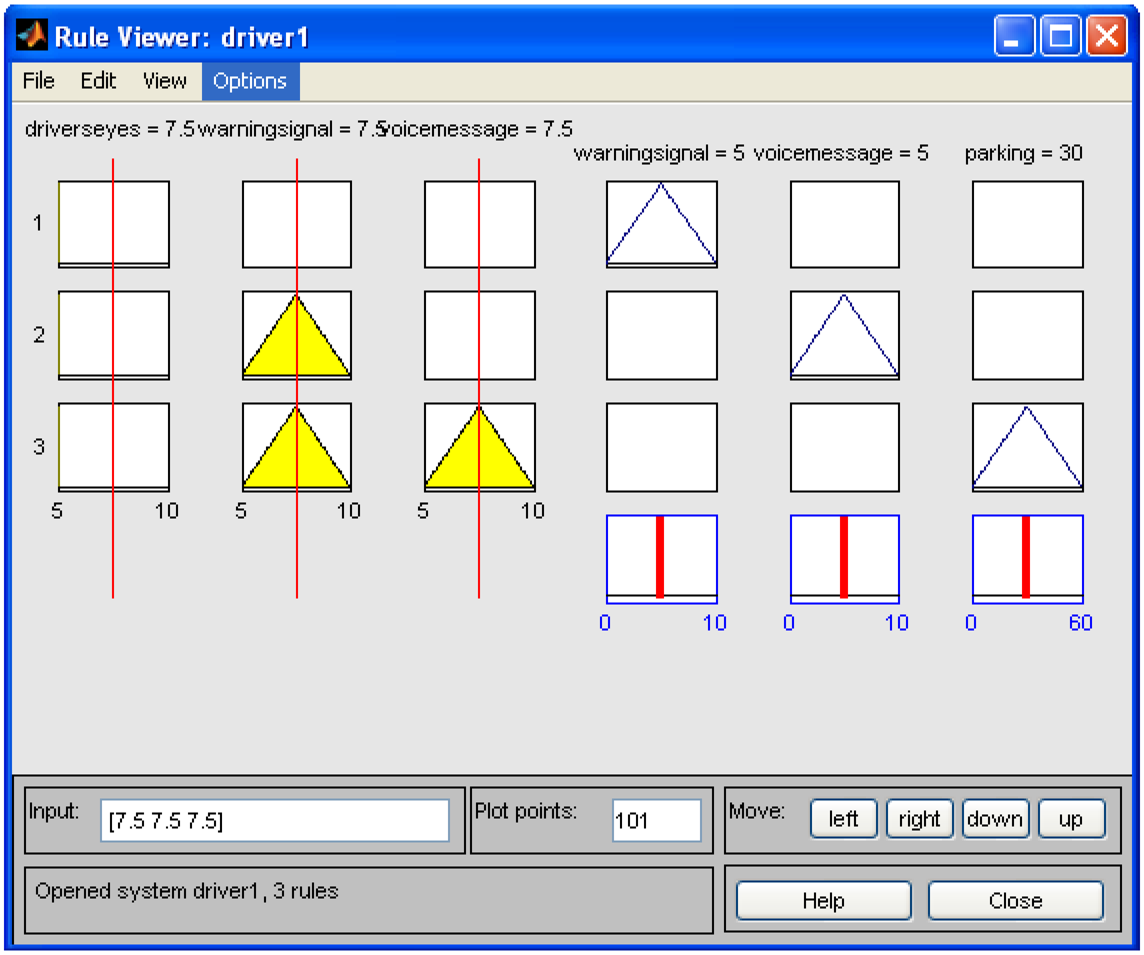
Weight: 1

FIS Name: driver1

Constructing rules using the graphical Rule Editor interface is fairly self evident. Based on the descriptions of the input and output variables defined with the FIS Editor, the Rule Editor allows you to construct the rule statements automatically, by clicking on and selecting one item in each input variable box, one item in each output box, and one connection item. Choosing none as one of the variable qualities will exclude that variable from a given rule. Choosing not under any variable name will negate the associated quality. Rules may be changed, deleted, or added, by clicking on the appropriate button[5].

The Rule Editor also has some familiar landmarks, similar to those in the FIS Editor and the Membership Function Editor, including the menu bar and the status line. The Format pop-up menu is available from the Options pull-down menu from the top menu bar—this is used to set the format for the display. Similarly, Language can be set from under Options as well.

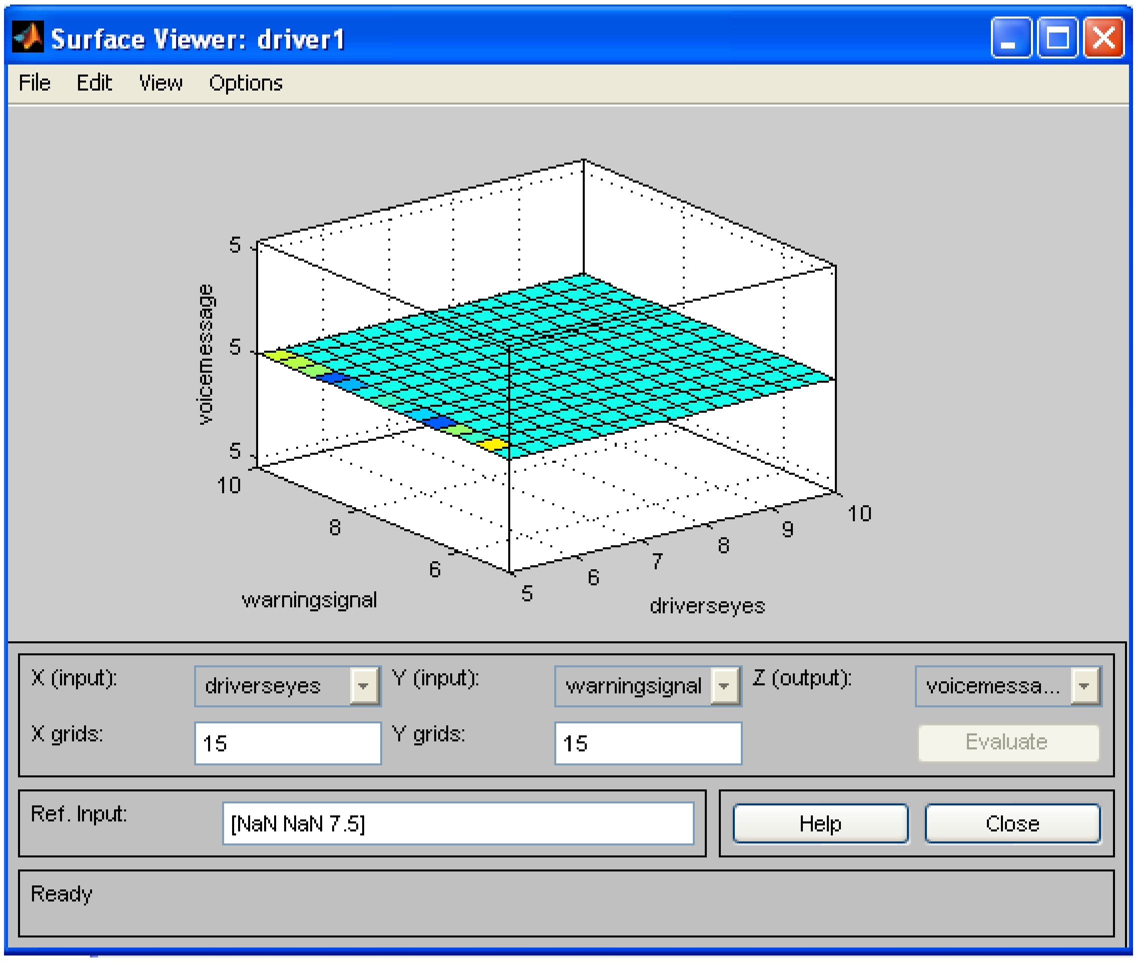
Rule Viewer



Opened system driver1 , 3 rules

The Rule Viewer displays a roadmap of the whole fuzzy inference process. It is based on the fuzzy inference diagram.

Surface Viewer



X (input): driverseyes
Y (input): warningsignal
Z (output): voicemessa...
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 7.5]
Ready

The Surface Viewer has a special capability that is very helpful in cases with two (or more) inputs and one output: you can actually grab the axes and reposition them to get a different three-dimensional view on the data. The Ref. Input field is used in situations when there are more inputs required by the system than the surface is mapping. Suppose you have a four-input one-output system and would like to see the output surface. The Surface Viewer can generate a three-dimensional output surface where any two of the inputs vary, but two of the inputs must be held constant since computer monitors cannot display a five-dimensional shape. In such a case the input would be a four-dimensional vector with NaNs holding the place of the varying inputs while numerical values would indicate those values that remain fixed. An NaN is the IEEE symbol for not a number.

RESULTS AND DISCUSSION

The output of the fuzzy system matches with our requirements using fuzzy mapping from driver’s continuous closed eyes and to park the vehicle safely.

CONCLUSION AND FUTURE WORK

Driving task is a highly complex behavior influenced by a large number of factors. The use context aware systems in cars aims at improving driver behavior. However the evaluation of the benefits brought by context-aware systems is difficult due to the complexity of driving task. This conceptual paper uses Fuzzy Logic to configure the design of a context aware system for vehicles and evaluates its benefits.

This system only looks at the number of consecutive frames where the eyes are closed. At that point it may be too late to issue the warning. By studying eye movement patterns, it is possible to find a method to generate the warning sooner.

REFERENCES

[1] Waltenegus Dargie,“Role of Probabilistic Schemes in Multisensor Context-awareness, ” *IEEE Computer Scociety 2007*.

[2] Andry Rakotonirainy “ Design of Context-aware Systems for Vehicles using Complex system Paradigms”.

[3] Thomas Stiefmeier, Daniel Roggen, and Gerhard Tröster *ETH Zurich* Georg Ogris and Paul Lukowicz *University of Passau* “Wearable Activity Tracking in car Manufacturing,”*IEEE Pervasive Computing2008.*

[4] George J.Klir and Bo Yuan “*Fuzzy Sets and Fuzzy Logic “ Theory and Applications.*

[5] J.S.Roger Janf , Ned Gully ”MATLAB Fuzzy Logic ToolBox”April 1997 second print.

[6] Jianghui Xin, Shunming Li, Qingbin Liao, Rixin Zhan “The Application of Fuzzy Logic in Exploration Vehicle” – IEEE 2007.

[7] Andry Rakotonirainy , Frederic Maire “Context-aware Driving Behaviour Model”

[8] Seng Loke “Context-aware pervasive systems”,Auerbach publications 2007.