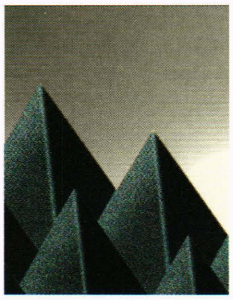


Fuzzy Systems for Simulating Human-Like Reasoning and Control

Therese F. Quaranta

Fuzzy theory is a powerful mathematical tool for simulating human-like reasoning and control. This advanced capability has been demonstrated in many commercial and noncommercial applications such as vehicle control, stock trading prediction, and pattern recognition automation systems. The principles involved in applying this theory to diverse problems are consistent and generalizable. Developing a model algorithm for adapting fuzzy theory to the solution of a broad range of control and information management problems is the goal of an Independent Research and Development project, the Fuzzy Development Model.

INTRODUCTION

As we enter an era of increased automation, we are developing computer algorithms to control systems and make timely decisions in the presence of uncertainties in data. Since humans are adept at handling information with uncertainties, an algorithm containing a technique for simulating human-like reasoning and control would be a definite asset. Fuzzy theory, which is a mathematical tool for representing and utilizing vague and ambiguous data in decision and control solutions, provides such a technique. Algorithms incorporating fuzzy mathematics merge our high-speed ability to evaluate information using computers with human-like reasoning and control.

The logic involved in these algorithms and the techniques used in system automation can be applied uniformly to diverse problems. A general-purpose method for utilizing the theory to automate systems can therefore be defined, and general software can be developed to implement the logic. The information gleaned from demonstrating this method with a model problem will serve as an example of this approach's application.

This article documents a general-purpose algorithm for incorporating fuzzy theory into diverse control and information management system solutions. The objectives of the Fuzzy Development Model, an APL Independent Research and Development project under the Software Engineering Thrust Area, were to develop an algorithm consisting of a defined methodology for applying the theory in system automation solutions, to implement general software employing the logic, and to demonstrate the approach using a model ship collision-avoidance problem. Results of the project, which has fulfilled all of its objectives, are detailed.

FUZZY CONCEPT

People are good at making decisions and controlling systems using vague and ambiguous data. These data uncertainties arise from system nonlinearities, time variations, poor-quality measurements, ill-defined conditions, and the equivocalness of human language. One reason we can process data with uncertainties is that we think in terms of concepts, which are inherently vague and ambiguous, rather than crisp numbers. Concepts involve the use of variables whose values do not have sharp boundaries.

Lotfi Zadeh,[1] a professor at the University of California at Berkeley, conjectured that automated systems could deal better with vague and ambiguous data if they were modeled after the human method of reasoning. To implement this approach, he introduced the linguistic variable, whose values are words rather than numbers. For example, the linguistic variable *steering angle* could have linguistic values SMALL LEFT, ZERO, and SMALL RIGHT. (In this article, words in italics are linguistic variables, words in all capital letters are linguistic values, and words in double quotation marks are human concepts.) Systems could then be automated using instruction or control statements, written in terms of the linguistic variable, that define the dynamics of the decision or control processes. These statements are in the form of if–then rules such as

$$\text{if } X \text{ is A, then } Y \text{ is B,}$$

where X and Y are linguistic variables and A and B are their linguistic values.

Humans are very good at navigation and collision avoidance. We drive cars, pilot helicopters and boats, and operate remotely controlled robots and toys. The linguistic variables used in such activities include speed and steering angle, distance to obstacles, and, to determine course and speed adjustments, the speeds and angles at which obstacles approach. When these linguistic variables are expressed as rules relating the input and output variables, the dynamics of the human decision and control processes can be modeled. A possible collision-avoidance rule is

if *obstacle angle* is SMALL LEFT,
then *steering angle* is SMALL RIGHT .

Here, *obstacle angle* is the input linguistic variable with a linguistic value SMALL LEFT, and *steering angle* is the output linguistic variable with a linguistic value SMALL RIGHT. This rule implies that if the *obstacle angle* is perceived as being a "small left" angle, the automated system will be controlled to steer at an angle that is "small right."

In human thought, linguistic values of input and output variables do not have sharp boundaries. For example, in the instruction to make a "small right" steering angle when maneuvering a vessel, the bounds defining a "small right" steering angle are not well delimited because a smooth and gradual transition exists in human thought from the concept "zero" steering angle, to "small right," to "right" steering angle. A mathematical representation of a set with vague boundaries, such as a set of "small right" steering angles, is the fuzzy set. Fuzzy sets, therefore, provide a mathematical representation for the linguistic variable. Fuzzy sets are discussed in the next section.

We can model the human method of reasoning with instruction or control statements using linguistic variables and the underlying mathematical representation of those variables. This human-like reasoning methodology has been termed fuzzy logic. A fuzzy model is the culmination of statements governing the decision or control process dynamics, their underlying vocabulary of linguistic variables and values, and inferential logic procedures. The resulting automated system is called a fuzzy system.

FUZZY SETS

Fuzzy set theory, introduced by Zadeh[2] and based on key ideas envisioned by Black[3] and Lukasiewicz,[4] is a generalization of set theory. In set theory, sets are defined on a universe of discourse such that elements within a set's domain belong to the set; otherwise, the element does not belong. For example, given the universe of possible steering angles, the set of SMALL RIGHT (SR) steering angles is defined over the domain of angles from 22.5 to 67.5°. Mathematically, this set is represented as SR = $\{\theta \in \Theta \mid 22.5° \le \theta \le 67.5°\}$, where θ denotes the *steering angle* and Θ is the universe of possible steering angles. Figure 1 is a graphical representation of the set SR. Here, $f(\theta)$ is the two-valued

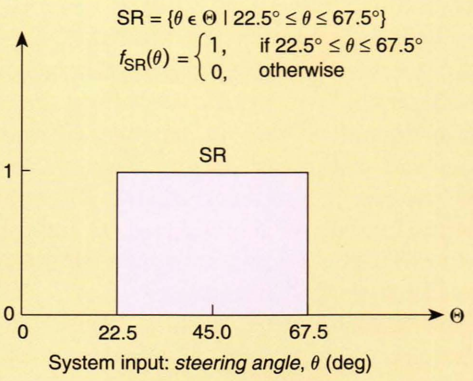


Figure 1. Set representation of a SMALL RIGHT (SR) steering angle.

characteristic function indicating the degree of membership of a *steering angle* θ in set SR as

$$f_{SR}(\theta)=\begin{cases}1, & \text{if } 22.5°\leq\theta\leq 67.5° \\ 0, & \text{otherwise}\end{cases}$$

This characteristic function represents a mapping of the universe of discourse of θ values, Θ, into the set $\{0, 1\}$ and is a two-valued or black and white mapping of the *steering angle* element θ into a concept of a "small right" steering angle.

Fuzzy sets are a generalization of these sets. Fuzzy sets are defined on the universe of discourse such that elements within a fuzzy set's domain can belong to the set to a partial degree. For example, a fuzzy set SR representing the concept "small right" steering angle is shown in Fig. 2. As illustrated, fuzzy sets add a dimension to the mathematical representation of a concept by allowing an element to belong to a set to a partial degree. The degree of membership $\mu_{SR}(\theta)$ represents the degree of compatibility between the steering angle element θ and the concept SR.

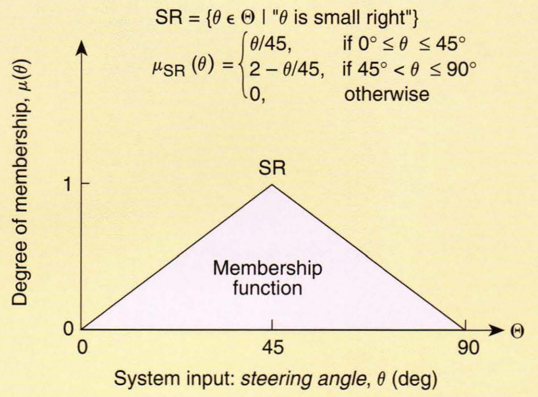


Figure 2. Fuzzy set representation of a SMALL RIGHT (SR) steering angle.

The triangular representation of the set SR shown in Fig. 2, which is called the membership function, is not unique. Examples of several other membership function representations are presented in Fig. 3. The membership function maps every element into the continuum [0, 1]. The shape and scope of the membership function are chosen to represent the compatibility of elements with a concept. For example, the triangular membership function in Fig. 2 models the concept of a "small right" steering angle to be totally compatible at a 45° turning angle, $\mu_{SR}(45°) = 1$, and to have monotonically decreasing compatibility as θ moves away from 45°, $0 < \mu_{SR}(\theta) < 1$. At steering angles less than 0 and greater than 90° the concept is totally incompatible, that is, $\mu_{SR}(\theta < 0° \text{ or } \theta > 90°) = 0$. The singleton membership function at the left of Fig. 3 models the concept of a "small right" steering angle that is completely compatible at 45° and incompatible at all other angles. Note that this is an example of the general fuzzy set converging to the classical set, which can be described by a two-valued degree of membership function or characteristic function.

The fuzzy set representation of concepts such as a "small right" steering angle is similar to the way humans perceive concepts and traverse among them, since the degree of compatibility of an element, such as a precise steering angle, with a concept, such as a "small right" angle, may be partial. An element can thus be partially compatible with several concepts at the same time, giving rise to a smoother transition between concepts than a representation with conventional sets. For example, in the fuzzy sets of Fig. 4a, the angles 67.5 and 67.6° represent the concepts "small right" and "big right" angles to about the same degree, since their degree of membership in the sets SR and BIG RIGHT (BR) is about the same. In the sets of Fig. 4b, these angles are characteristic of different concepts, even though they differ by only 0.1°.

The theory of fuzzy sets includes operators for combining and manipulating the sets. Fuzzy operators for complement, intersection, union, and containment for

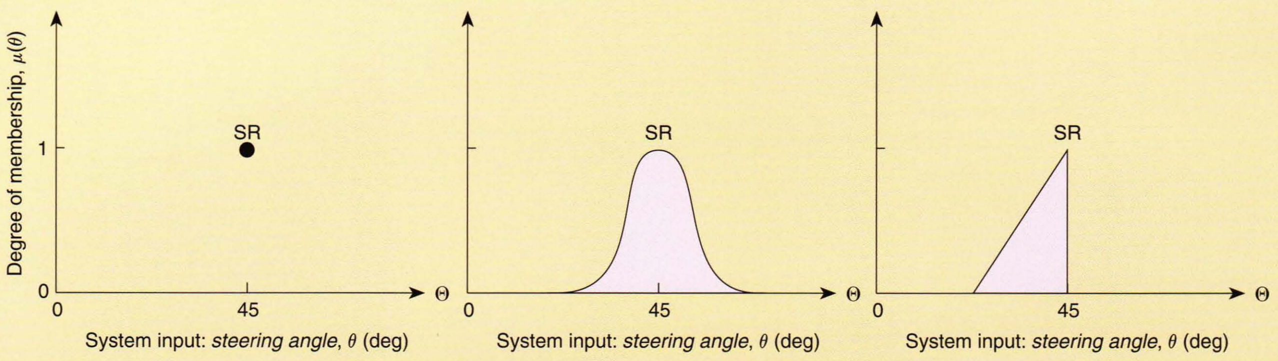


Figure 3. Several fuzzy set representations of a SMALL RIGHT (SR) steering angle illustrating that the membership functions are not unique.

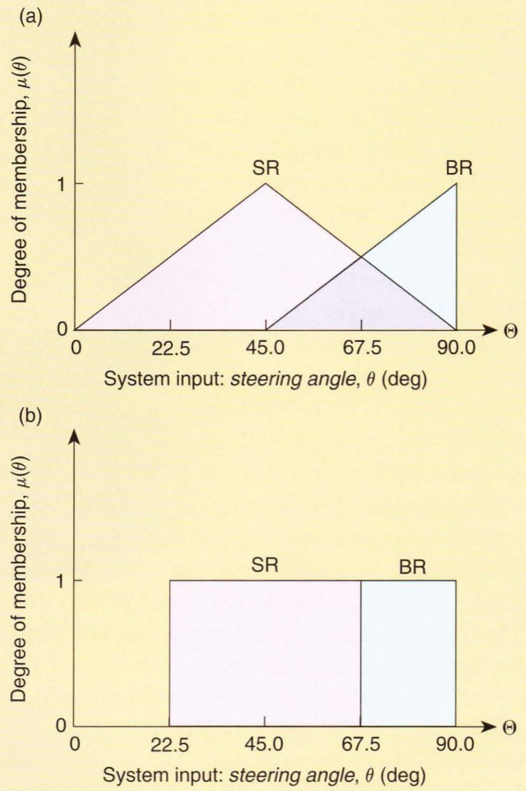


Figure 4. (a) Fuzzy set and (b) classical set representations of a SMALL RIGHT (SR) steering angle and BIG RIGHT (BR) steering angle.

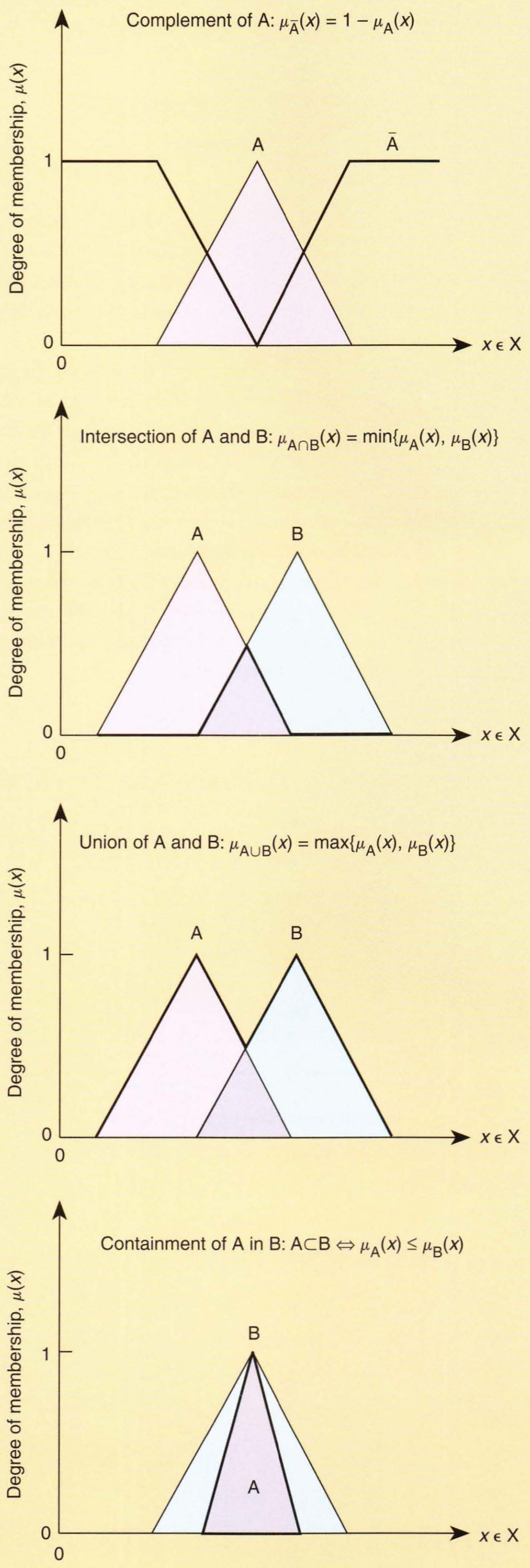


Figure 5. Fuzzy set complement, intersection, union, and containment operators.

two fuzzy sets A and B defined on the universe of discourse X are listed and illustrated in Fig. 5. In the figure, the fuzzy regions resulting from these operations are outlined in bold and can be interpreted as follows:[5]

Complement:	To what degree does an element not belong?
Intersection:	To what extent are items in both sets?
Union:	To what degree are items in either set?
Containment:	What groups belong to other groups?

Note that the vertical sum of the membership functions at a particular x value is restricted to 1 only if the overlapping sets are complementary. These definitions satisfy well-defined axiomatic principles. Different definitions meeting these principles have also been given and are expanded upon by Klir and Folger.[6] Note that when the degree of membership is restricted to the set {0, 1}, these functions define the classical set operators.

FUZZY SYSTEMS

The structure of a fuzzy system is similar to that of conventional automated systems. Figure 6a illustrates the three main components: the physical device or process, the system model, and the process logic. System automation entails developing a decision or control solution using the process logic on the basis of a system model and inputs from the physical device or process. The primary difference between conventional and fuzzy systems is in the system model approach. Conventional systems use models of the physical device or process, whereas fuzzy systems use models of the human operator's or decision maker's behavior.

In fuzzy systems, as illustrated in Fig. 6b, the process logic involves three primary operations: fuzzification, rule evaluation, and defuzzification. The following sections describe and demonstrate these three operations using representative system variables and rules. However, the procedures demonstrated on the problem-specific variables and rules are actually independent of the particular system problem. The problem-specific information is input to each operation block, as shown in the ovals on the right of Fig. 6b, for standard processing.

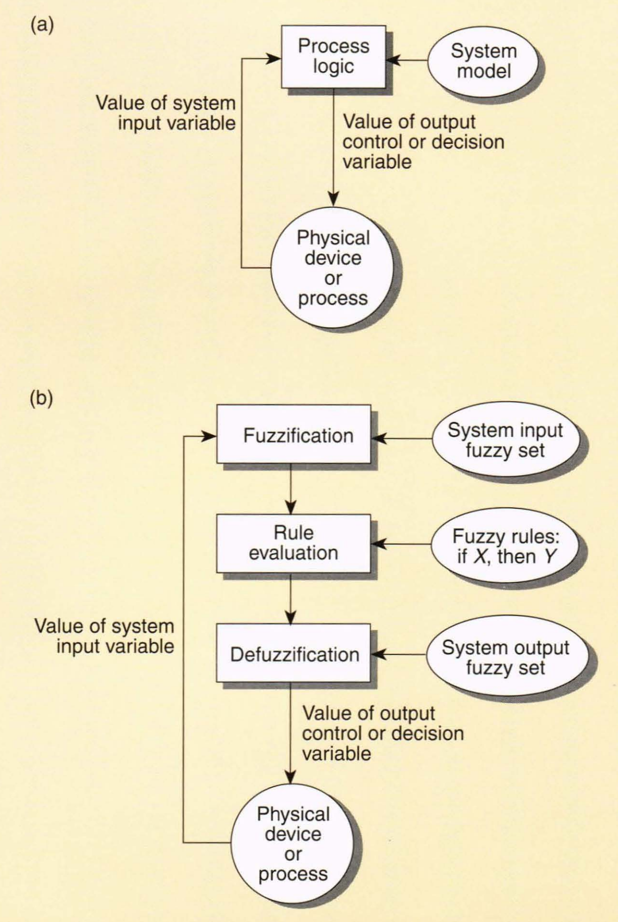

(a)
Process logic
System model
Value of system input variable
Value of output control or decision variable
Physical device or process
(b)
Fuzzification
System input fuzzy set
Rule evaluation
Fuzzy rules: if X, then Y
Value of system input variable
Defuzzification
System output fuzzy set
Value of output control or decision variable
Physical device or process

Figure 6. (a) A conventional structure for an automated system. (b) A fuzzy system performs the processing logic in three operating stages: fuzzification, rule evaluation, and defuzzification. Problem-specific information is input to each operation, as shown in the ovals on the right.

Because the three process logic operations are problem-independent, general software can be developed to perform these procedures. This software can then be applied to many problems by changing the problem-specific input information. General software that has been developed in the C language to implement the process logic procedures is described in the following sections.

Fuzzification

Fuzzification is the process through which the value of a system input variable, or crisp measurement, is taken from the physical device or process and mapped into fuzzy sets defined for this system input. The output of this operation is a *fuzzy input*, which is the degree of membership of the crisp measurement in the fuzzy sets. The *fuzzy input* is the input for the rule evaluation process.

For example, if the physical system is a vessel steering control, a system input variable may be the desired course change to reach a destination, with a crisp measurement of 38°. For this *desired angle* (*DA*) system input, five fuzzy sets stored in the system input fuzzy set are BIG LEFT (BL), SMALL LEFT (SL), ZERO (ZE), SMALL RIGHT (SR), and BIG RIGHT (BR). These fuzzy sets and their underlying membership functions are shown in Fig. 7a. The crisp value of the linguistic

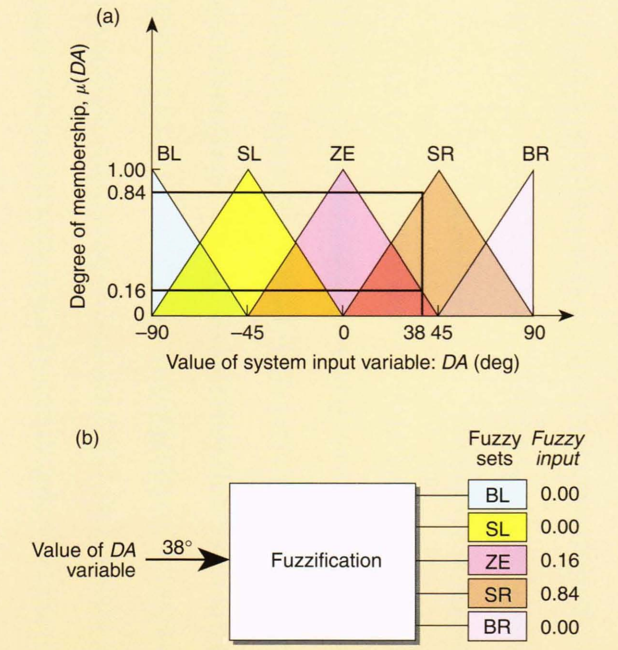


Figure 7. (a) System input is the *desired angle* (*DA*), which is defined by five fuzzy sets: BIG LEFT (BL), SMALL LEFT (SL), ZERO (ZE), SMALL RIGHT (SR), and BIG RIGHT (BR). (b) Fuzzification is the mapping of the crisp system input value into the fuzzy set membership functions to produce a *fuzzy input*.

variable *DA*, 38°, maps into a *fuzzy input*, which is a variable containing the degree of membership that the crisp angle maps into each of the fuzzy sets. This mapping is illustrated in Fig. 7a. The mapping defines a 0.16 compatibility of 38° with the concept "zero desired angle" and 0.84 compatibility with the concept "small right desired angle." A 38° value for *DA* is incompatible with the other concepts. These degrees of membership in the fuzzy sets constitute the *fuzzy input*, as illustrated in Fig. 7b.

Rule Evaluation

Rule evaluation, or fuzzy inference, is the process of calculating a fuzzy control or decision output based on linguistic rules governing the control or decision system dynamics.

Fuzzy rules are usually if–then statements that describe the action to be taken in response to various system inputs. They are written in terms of linguistic input and output variables and linguistic values or fuzzy sets. A representative rule format is

$$\text{if } \underbrace{X \text{ is A}}_{\text{antecedent}} \text{ and } \underbrace{Y \text{ is B}}_{\text{antecedent}}, \text{ then } \underbrace{Z \text{ is C}}_{\text{consequent}}.$$

Here, *X* and *Y* are input variables, *Z* is an output variable, and A, B, and C are their associated linguistic values or fuzzy sets. For example, a rule relating the *DA* (system input), *obstacle angle* (OA) (another system input), and *steering angle* (system output) is

if *DA* is SR and OA is SR, then *steering angle* is ZE .

Here, the OA's linguistic values, or fuzzy sets, and underlying membership functions are taken to be identical to those for the *DA* shown in Fig. 7a. The *fuzzy input* for a measured obstacle angle of 10° is illustrated in Fig. 8. The fuzzy sets stored in a system output fuzzy set for the *steering angle* are shown in Fig. 9.

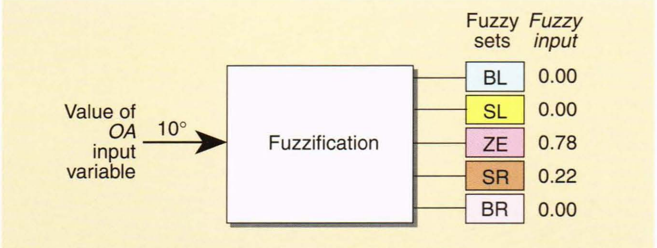


Figure 8. The system input *obstacle angle* (*OA*) is represented by the same five fuzzy sets describing the *desired angle* (*DA*). The *fuzzy input OA* is shown for a crisp *OA* input of 10°.

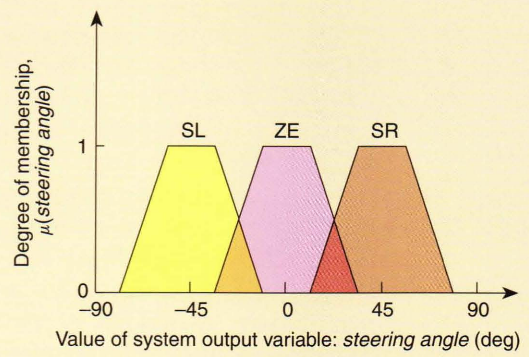


Figure 9. The system output is the *steering angle*, which is defined by three fuzzy sets: SMALL LEFT (SL), ZERO (ZE), and SMALL RIGHT (SR).

A popular rule evaluation method is called 'min–max inferencing.' This technique assumes the isomorphism between logic operators and set operators indicated in Table 1. Rule evaluation proceeds by first applying the logical 'and' operator to the rule antecedents, or taking the minimum of the antecedents' strengths. The antecedents' strength is equal to the corresponding *fuzzy input* value. The result is the strength of the rule as it applies to the consequent action.

For example, the strength of the antecedent '*DA* is SR' is 0.84, since the 38° desired angle is consistent with the concept SR by 0.84. The strength of the antecedent 'OA is SR' is 0.22, since the 10° obstacle angle is consistent with the concept SR by 0.22. The strength of the consequent action '*steering angle* is ZE' is the minimum of 0.84 and 0.22, or 0.22. Therefore, the rule conditions imply a 0° steering angle to the degree of 0.22. (The min–max inferencing process is further illustrated in the boxed insert, The Rule Evaluation Process.)

Defuzzification

Defuzzification is the process of mapping a *fuzzy output* onto a crisp system output value. One of the most popular defuzzification techniques is the centroid, or center-of-gravity (COG) method, which is performed by first mapping the *fuzzy output* onto the defined system output fuzzy sets. The area defined by this mapping is used in the COG calculation to derive the COG value for the output control or system

Table 1. Relation between set and logic operators.

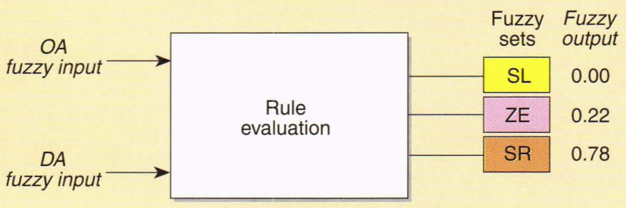
Set operator	Logic operator	Definition
Intersection	and	$\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$
Union	or	$\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$

THE RULE EVALUATION PROCESS USING MIN–MAX INFERENCING

Rule 1: If *DA* is SR and *OA* is SR, then *steering angle* is ZE.
Rule 2: If *DA* is ZE and *OA* is SR, then *steering angle* is ZE.
Rule 3: If *DA* is SR and *OA* is ZE, then *steering angle* is SR.

Step 1: Calculate strength of rule.
Strength of Rule 1 = $\min[\mu_{SR}(38°), \mu_{SR}(10°)] = \min(0.84, 0.22) = 0.22$
Strength of Rule 2 = $\min[\mu_{ZE}(38°), \mu_{SR}(10°)] = \min(0.16, 0.22) = 0.16$
Strength of Rule 3 = $\min[\mu_{SR}(38°), \mu_{ZE}(10°)] = \min(0.84, 0.78) = 0.78$

Step 2: Calculate *fuzzy output*.
μ_{ZE}(*steering angle*) = max(strength of Rule 1, strength of Rule 2) = max(0.22, 0.16) = 0.22
μ_{SR}(*steering angle*) = strength of Rule 3 = 0.78



Given the *fuzzy inputs* and fuzzy rules, the strength of the rule can be calculated by applying the logical 'and' operator. The final step is to calculate the *fuzzy output* by applying the logical 'or' operator.

When more than one rule suggests the same action, such as Rule 1 and Rule 2, which both imply a 0° course change, the rule that applies most is used. The resulting degree of membership of the output variable in its defined fuzzy sets is the *fuzzy output*.

The fuzzy additive method is another inferencing method. In this method each rule strength is calculated, as in min–max inferencing, by taking the minimum of the antecedent's rule strength. However, when more than one rule implies the same consequent action, the consequent strength is the minimum of 1.0 and the sum of the rule strengths implying that consequent. For the example here, the *fuzzy output* μ_{ZE} (*steering angle*) is calculated as

$$\begin{aligned}\mu_{ZE}\ (\textit{steering angle}) &= \min(1.0, \text{strength of rule 1} + \text{strength of rule 2}) \\ &= \min(1.0, 0.22 + 0.16) = 0.38\ .\end{aligned}$$

The *fuzzy output* μ_{SR}(*steering angle*) is the same, 0.78. An advantage of this method is that it allows all the rules to contribute something to the final model solution. This approach is used in decision models when accumulating evidence, or strength, on a consequent action is important, such as in risk assessment.

variable, as shown in the boxed insert, Calculating the Center of Gravity.[7]

General Software for Fuzzy Inferencing

The general software for the fuzzification, rule evaluation, and defuzzification processes is written in the C language. The system inputs, outputs, and membership functions and rules are associated using link list data structures, as originally defined by Viot[7] and listed in the boxed insert on C-language structures. An algorithm has been developed to accept the problem-specific information and set up the three required operation inputs: system input fuzzy set, fuzzy rules, and system output fuzzy set. The number of system inputs, outputs, and associated membership functions or rules is unlimited. The limitation of this software is that the membership functions must have a shape describable by two points and two slopes, such as triangles, trapezoids, and rectangles, as illustrated in Fig. 10.

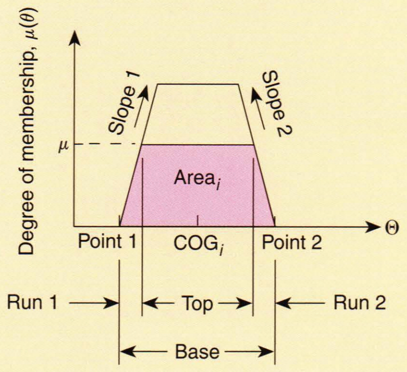
FUZZY SYSTEM DEVELOPMENT

Fuzzy system development is defined by the two stages illustrated in Fig. 11. In the initial stage shown in Fig. 11a, the overall system is first evaluated for its functional and operational characteristics to identify separate component processes. The output of this stage is an identification of system components that can be implemented using fuzzy logic. This output is the input to the second stage, as defined in Fig. 11b. In the second stage, a fuzzy system is developed for the component using the following processing steps:[8,9]

1. **Define Model Functional and Operational Characteristics:** This step entails defining the information

CALCULATING THE CENTER OF GRAVITY (COG)

Step 1: Calculate area of each fuzzy set i defined by *fuzzy output* θ.

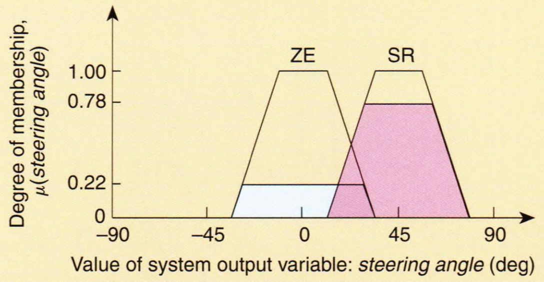


Run 1 = $\mu(\theta)$/slope 1
Run 2 = $\mu(\theta)$/slope 2
Base = point 2 − point 1

Top = base − run 1 − run 2
Area = $\mu(\theta) \times$ (base + top)/2

Step 2: Calculate the COG over all fuzzy sets.

$$\text{COG} = \frac{\sum_i \text{area}_i \times \text{COG}_i}{\sum_i \text{area}_i} \quad \forall_i$$



Defuzzification is performed by mapping the *fuzzy outputs* onto the system output fuzzy sets, computing the area defined, and finding its COG. Note that only the fuzzy sets with nonzero degrees of membership are shown. The figures illustrate the two steps for calculating the COG. The figure in Step 2, for example, shows the *fuzzy output* values of 0.22 and 0.78 degrees of membership in the fuzzy sets ZE and SR, respectively. The areas defined by these degrees of membership for each of the fuzzy sets are shaded. The COG, 37.4°, which represents the crisp output control command for the *steering angle* system output variable, is calculated as the centroid of the shaded areas.

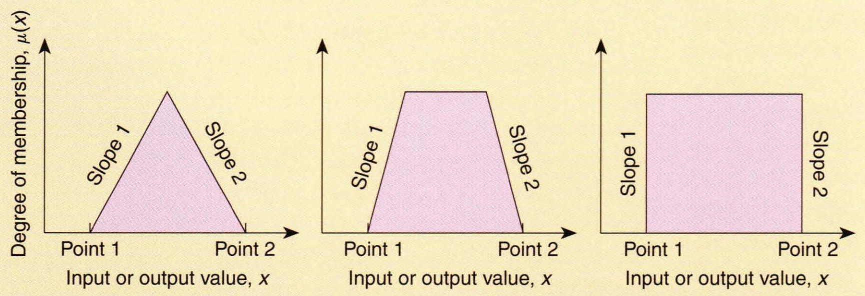


Figure 10. Membership functions for input into general software must be describable by two points and two slopes such as triangles, trapezoids, and rectangles.

flow into the system, the basic transformations performed on the data, and what data are output.

2. **Decompose Model Variables into Fuzzy Sets**: Each system input and output variable is decomposed into one or more qualitative labels or fuzzy sets. A membership function is defined for each fuzzy set that semantically represents the concepts associated with the label. Some rules of thumb are as follows:

- The number of labels associated with a variable should generally be an odd number between 5 and 9.
- Each label should overlap with its neighbors by 10 to 50% of the neighboring space, and the sum of the vertical points of the overlap should always be less than 1.
- The density of the fuzzy sets should be highest around the optimal control point of the system and should then decrease as the distance from that point increases.[8]

3. **Write Rules to Define Model Behavior:** The system dynamics are rules that tie the system inputs to the system outputs. Rules are usually expressed in the if–then format such as if X, then Y.
4. **Select a Defuzzification Method:** Many defuzzification strategies have been developed for various problem types; however, the most popular is the centroid method that was described in the previous section. Others have been well described by Cox.[9]

Once the fuzzy model has been constructed, the process of simulating the fuzzy system and tuning the model begins. The system's results are compared against known test cases for validation. When the results are not as desired, changes are made either to the fuzzy set descriptions or to the mappings encoded in the rules. Once the desired result is achieved, this fuzzy system component is incorporated into the larger system. These procedures will be demonstrated for a fuzzy ship collision-avoidance system.

FUZZY SHIP COLLISION-AVOIDANCE SYSTEM

Problem

Much attention has focused on open ocean and confined-water vessel collisions because of public interest in their environmental impact. For example, collisions involving vessels carrying large loads of oil have caused extensive environmental hazards, and there are concerns about ships carrying nuclear reactors. With this interest in protecting the environment, as well as the lives of seamen, many investigations have been made to develop automated navigation and collision-avoidance systems.[10–14] The goal of these automated systems is to eliminate human error resulting from such factors as fatigue by removing the human operator from continuous vessel control. To automate the navigation and collision-avoidance processes of the mariner, they must first be defined and modeled.

The functional and operational characteristics of a mariner's dynamic processes are illustrated in Fig. 12 and translated into a context diagram centered around the mariner in Fig. 13. The dynamic process begins with a mission that has two objectives: (1) to navigate the vessel to a destination and (2) to avoid obstacles encountered along the path. Given this mission and knowledge of the vessel's current location, speed, and heading, a mariner will chart the desired course to the mission destination. Using this desired course, heading

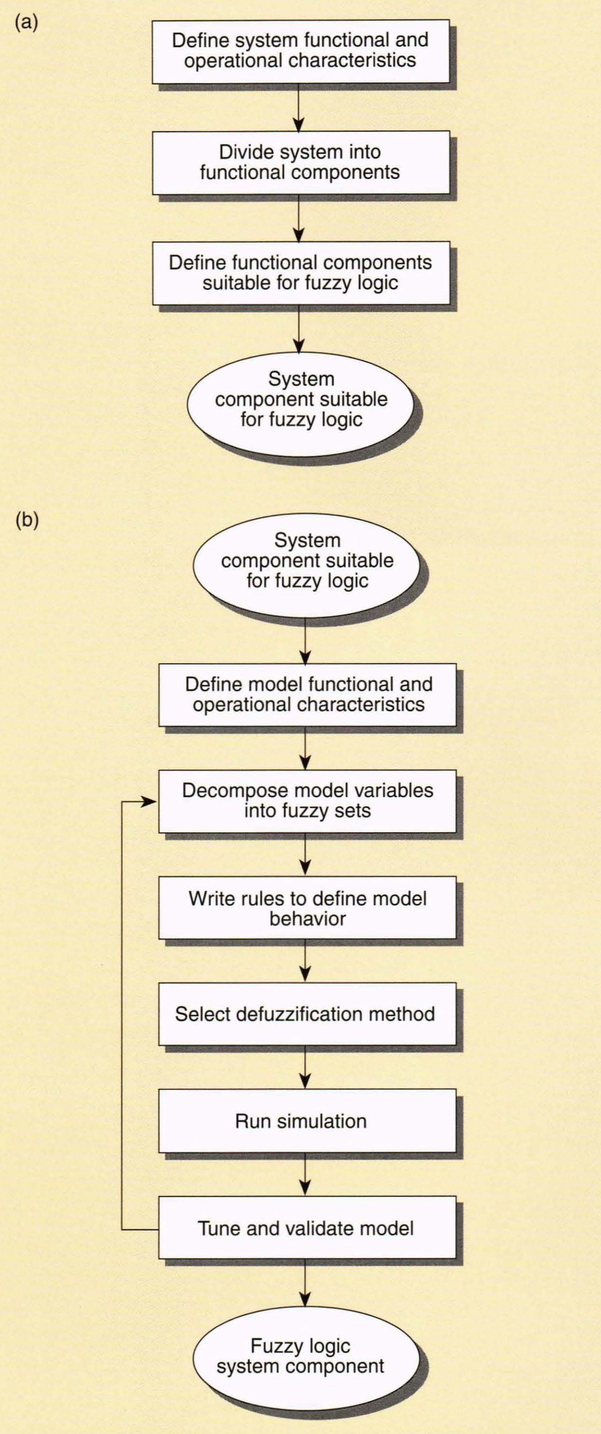

(a)
Define system functional and operational characteristics
Divide system into functional components
Define functional components suitable for fuzzy logic
System component suitable for fuzzy logic
(b)
System component suitable for fuzzy logic
Define model functional and operational characteristics
Decompose model variables into fuzzy sets
Write rules to define model behavior
Select defuzzification method
Run simulation
Tune and validate model
Fuzzy logic system component

Figure 11. Fuzzy system development is a two-stage process. (a) The first stage entails system evaluation for the appropriateness of fuzzy logic in the solution. (b) The second stage builds the fuzzy model for a component identified in the first stage.

C-LANGUAGE STRUCTURES USED FOR SYSTEM INPUTS, OUTPUTS, AND MEMBERSHIP FUNCTIONS AS WELL AS RULES ORIGINALLY DEFINED BY VIOT

```
/*  io_type structure used to build a link-list of system inputs and system outputs. */
struct io_type{
          char name [MAXNAME];               /*name of system input or output           */
          float value;                       /*crisp value of system input or output    */
          struct mf_type                     /*list of linguistic values, or fuzzy set labels */
                 *membership_fns;            /*     defined for this input or output    */
          struct io_type *next;              /*pointer to next input or output          */
          };

/*  Fuzzy sets and membership functions (mf) associated with each system input and output.  */
struct mf_type{
          char name [MAXNAME];               /*linguistic value, or fuzzy set label     */
          float value;                       /*degree of membership                     */
          float point1;                      /*left x-axis point of mf domain           */
          float point2;                      /*right x-axis point of mf domain          */
          float slope1;                      /*slope of left side of mf                 */
          float slope2;                      /*slope of right side of mf                */
          struct mf_type *next;              /*pointer to next mf for system input or output*/
          };

/*Rules are of the if–then form. The rule_element_type pointers point to the degree of
membership value mf_type->value for the system input or output specified by the
antecedent for the 'if' side and the consequent for the 'then' side. */
struct rule_type{
          struct rule_element_type *if_side;     /*pointer to io_type of antecedents in rule */
          struct rule_element_type *then_side;   /*pointer to io_type of consequent in rule  */
          struct rule_type *next;                /*pointer to next rule                      */
          };

struct rule_element_type{
          float *value;                          /*pointer to antecedent/consequent mf_type->value*/
          struct rule_element_type *next;        /*next antecedent/consequent element in rule */
          };
```

Note: This material has been adapted with permission from Ref. 7.

and speed are computed, and the vessel is navigated according to standard navigational rules. During vessel navigation, the mariner monitors the safety situation. If a hazard is detected, an avoidance action is taken based on collision regulations as well as navigational rules.

Although numerous approaches to automated navigation and ship collision-avoidance systems have been suggested, a satisfactory solution has yet to be achieved. Coenen et al.[11] reported that this failure has largely been due to the algorithmic approach, which generally entails mathematical modeling of the system and encounter situation. This approach becomes inadequate when problem complexity increases, as when multiple encounters occur simultaneously.

Additionally, the models developed using the algorithmic method do not capture the way mariners themselves navigate and solve collision-avoidance problems. Mariners interpret standard regulations on the basis of their training and experience and an evaluation of the situation. Rule-based expert systems, however, provide an automated approach to simulate the mariner's use of regulations. The conventional rule-based approach, which employs binary logic, has shown some success,[11] but its main limitation is the required hard coding of boundaries on concepts in the regulations, such as 'timely and positive' action to avoid a collision, and the need to recognize the point at which vessels are '*in extremis*' and must turn to avoid a collision. These boundaries define the conditions where a rule does or does not apply. Rules are frequently implemented through a tree-like reasoning structure.

A rule-based expert system employing fuzzy logic uses fuzzy sets to represent inexact or fuzzy concepts as well as fuzzy inferencing so that rules can apply partially. Actions are inferred by parallel processing the rules along with their degree of applicability. A fuzzy logic approach will be demonstrated here.

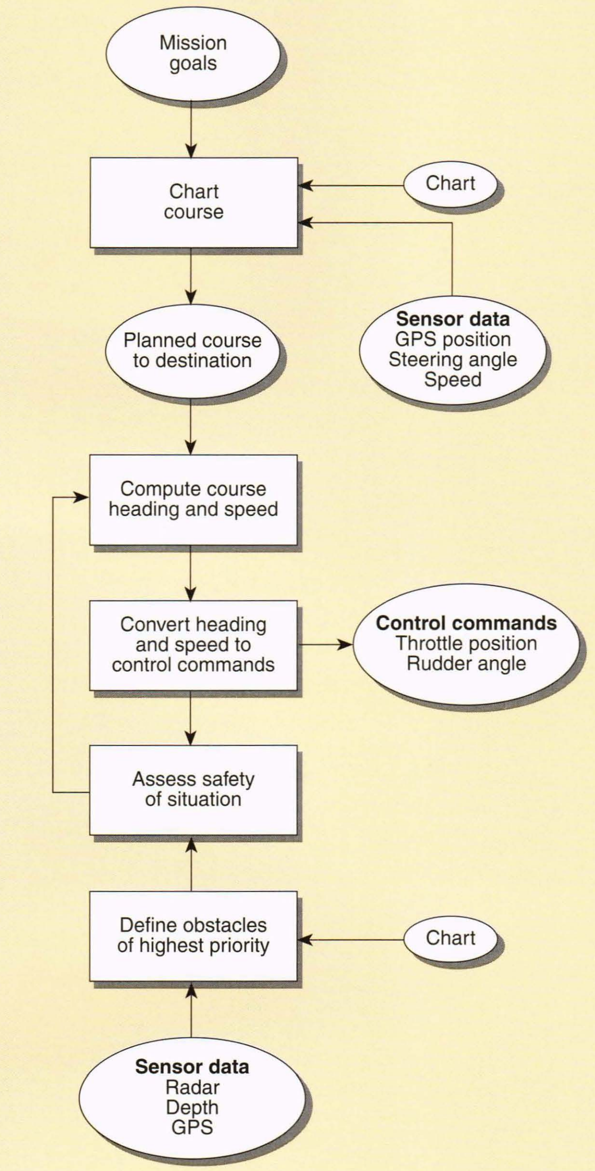

Mission goals
Chart course
Chart
Planned course to destination
Sensor data
GPS position
Steering angle
Speed
Compute course heading and speed
Convert heading and speed to control commands
Control commands
Throttle position
Rudder angle
Assess safety of situation
Define obstacles of highest priority
Chart
Sensor data
Radar
Depth
GPS

Figure 12. Functional and operational characteristics of a mariner's navigation and collision-avoidance process (GPS = Global Positioning System).

System Problem

The system problem is to design a control solution by employing the logic that mariners use for navigating a vessel to a goal location without collisions with unexpected obstacles and with minimum deviation from a defined course. On the basis of Figs. 12 and 13, this problem can be broken into several fuzzy control modules:

- A charting routine to define the optimal path plan
- A system for automatic navigation and steering control
- A system for obstacle identification at each sensor
- A system for sensor fusion and obstacle identification
- A collision-avoidance system for circumventing obstacles found by the sensor identification system

To reduce the scope of the problem for this Independent Research and Development project, only the fuzzy collision-avoidance module is being developed. A general simulation is employed to create sensor data inputs and the required ship heading to the destination. Figure 14 is a context diagram for this reduced problem. Here, the actual regulations governing collision-avoidance actions are not used. Quantification of the collision-avoidance regulation concepts is beyond the scope of this project; therefore, general rules are used instead to demonstrate the fuzzy logic concept. The system can be modified later to include more realistic rules.

Simulation Problem

The geometry of the simulation problem is defined in Fig. 15. This simulation is similar to that described by Kong and Kosko.[15] The system is defined by three position variables $\beta(t)$, $x(t)$, and $y(t)$, and a constant speed variable v, where $\beta(t)$ specifies the angle of the ship with respect to the vertical or northerly direction and t is the discrete simulation time step. The coordinate pair $x(t)$ and $y(t)$ specify the position of the vessel's bow center at time step t. The output, or control variable, is the rudder angle $\theta(t)$.

The initial simulation stage is defined as a plane, $(0, 100) \times (0, 100)$, with the initial starting position at $(0, 0)$ and the destination at $(100, 100)$. The planned course to the destination is a straight path along the diagonal joining the initial and final positions.

Five sensor measurements provide obstacle information to the vessel. As shown in Fig. 15, the sensors point in the -90, -45, 0, 45, and 90° directions relative to the direction of the vessel's motion angle $\beta(t)$. A sensor's measurement consists of the closing rate (positive for closing) and distance of the closest obstacle within its field of view. The field of view for each sensor is listed in Table 2. The sensors' fields-of-view do not overlap, and an obstacle can therefore not be seen by more than one sensor. An obstacle simulation system generates these sensor inputs using an obstacle database. The obstacle database consists of the speed and the obstacle's current and destination positions.

Figure 16 is the data flow diagram for the simulation, which begins with initial and destination positions along with speeds for the vessel and the obstacles. The simulation computes which obstacles are within the field of view of each sensor, the closest obstacle per sensor, and the closing rate of these obstacles. The desired vessel course is computed as the angle of a vector drawn from the current ship position to the planned course (pc) destination location $\beta_{pc}(t)$. The

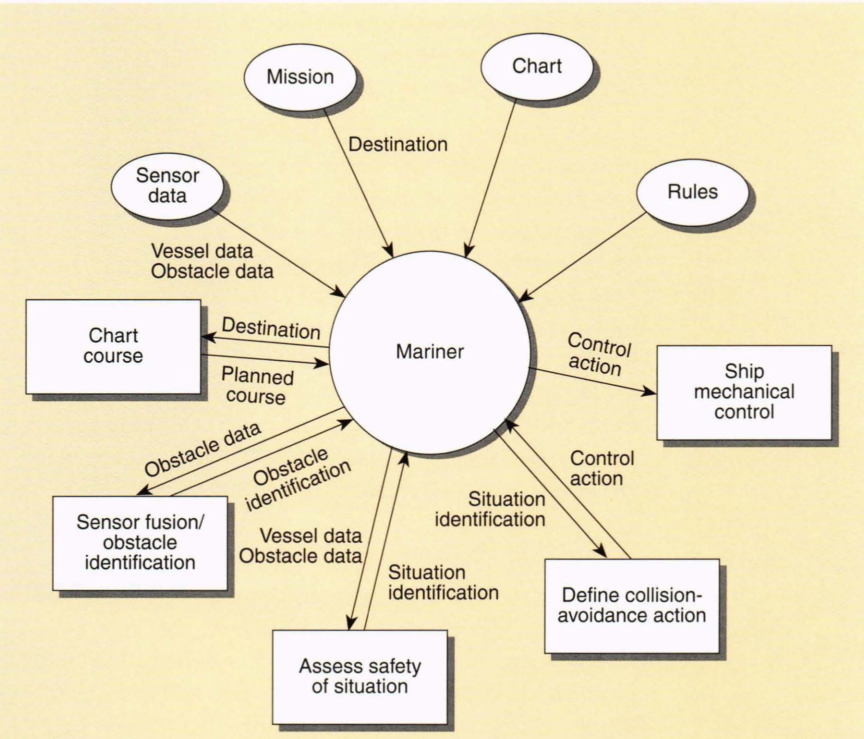


Figure 13. Context diagram describing the inputs and outputs used by the mariner in navigation and collision avoidance.

DA is the difference between the vessel's current course $\beta(t)$ and the desired course $\beta_{pc}(t)$. The *DA* and the sensor data are input into a fuzzy collision-avoidance decision system, which is described in the next section. This system computes a direction, based on fuzzy logic, to a safe course of travel for the next time step. The output is the rudder control angle $\theta(t)$. The simulation updates the vessel's position using the following simplified kinematic equations:

$$\beta(t+1)=\beta(t)+\theta(t),$$
$$x(t+1)=x(t)+v\sin[\beta(t+1)],$$
$$y(t+1)=y(t)+v\cos[\beta(t+1)].$$

Similarly, the obstacle positions are updated. The simulation continues until the vessel reaches the destination or the end of the stage boundaries.

Fuzzy Collision-Avoidance System

The fuzzy collision-avoidance system is an extension of an adaptive path execution system for a robot developed by Yen and Pflunger.[16] In this system, a collision-avoidance method for both stationary continuous obstacles, such as walls, and stationary discrete obstacles, such as tables, is considered. The primary differences between the adaptive path execution

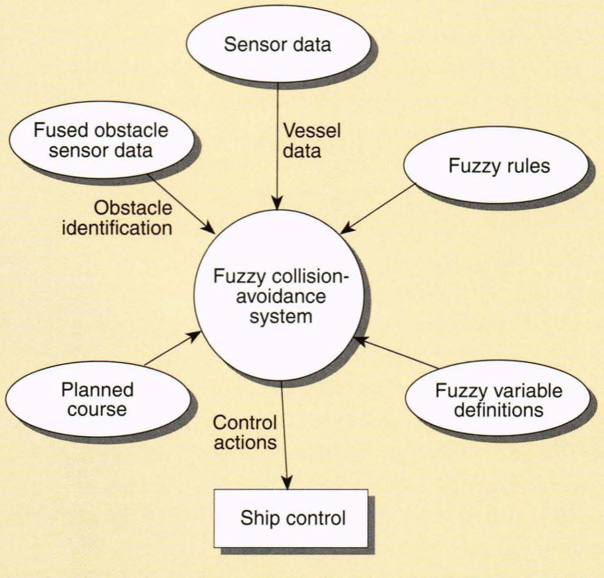


Figure 14. Context diagram for simulation of a fuzzy collision-avoidance solution.

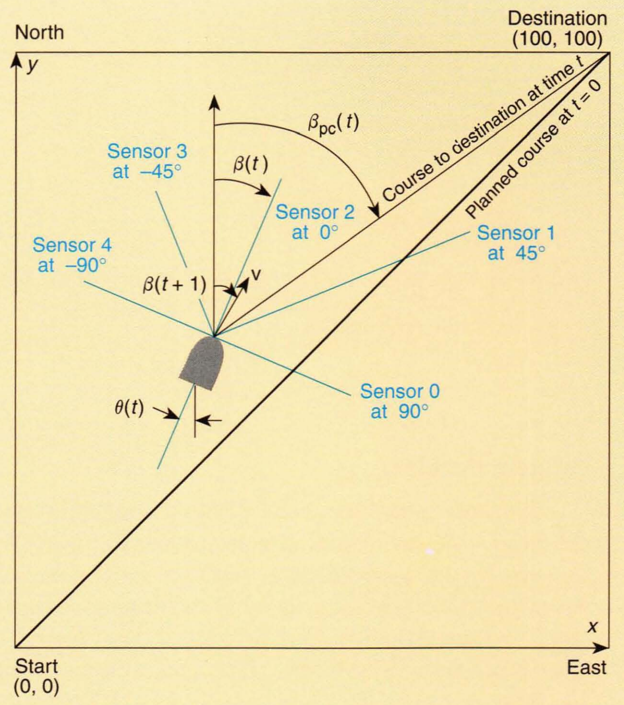


Figure 15. The geometry of the simulation problem, where $\beta(t)$ is the vessel heading angle at time *t*, $\beta_{pc}(t)$ is a heading angle (planned course) from the vessel to the destination at time *t*, and $\theta(t)$ is the rudder control angle for time *t* that defines the change in heading for time $t+1$ (v = constant speed variable).

Table 2. Sensor field of view.

Sensor number	Sensor angle[a]	Field of view[b] (deg)
0	90°	$[\beta(t) + 90.0$ to $\beta(t) + 67.5)$
1	45°	$[\beta(t) + 67.5$ to $\beta(t) + 22.5)$
2	0°	$[\beta(t) + 22.5$ to $\beta(t) - 22.5)$
3	−45°	$[\beta(t) - 22.5$ to $\beta(t) - 67.5)$
4	−90°	$[\beta(t) - 67.5$ to $\beta(t) - 90.0]$

[a]Sensor angle is relative to vessel heading $\beta(t)$.
[b]A bracket indicates the endpoint is inclusive. A parenthesis indicates the endpoint is noninclusive.

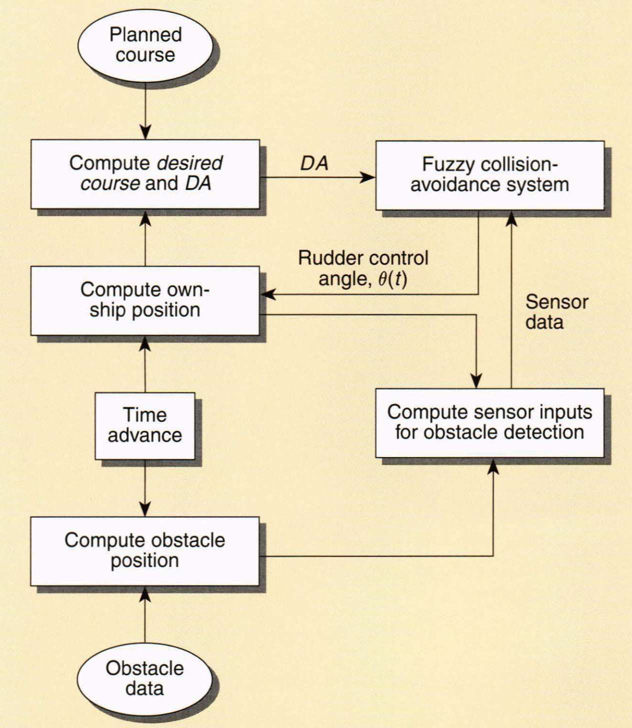

Planned course
Compute *desired course* and *DA*
DA
Fuzzy collision-avoidance system
Rudder control angle, $\theta(t)$
Compute own-ship position
Sensor data
Time advance
Compute sensor inputs for obstacle detection
Compute obstacle position
Obstacle data

Figure 16. Diagram of ship collision-avoidance simulation data.

system and the approach used here for the ship collision-avoidance system are that only the discrete obstacles are considered, obstacle motion is allowed, and the control solutions are modified accordingly. These obstacles represent other moving and stationary vessels. This description of the collision-avoidance system follows that of Yen and Pflunger.

Overview

The data flow diagram for the collision-avoidance system is shown in Fig. 17. The two inputs on the left, the sensor data and *DA*, are input from the simulation defined in Fig. 16. The sensor data consist of a closest obstacle distance and closing rate for each sensor. The *DA* is measured as the difference between the vessel's current course $\beta(t)$ and the desired course $\beta_{pc}(t)$. The output is the rudder control angle $\theta(t)$ at which the vessel travels to its next time position. These data are processed in the following steps:

1. Mapping sensor-input obstacle locations and motion to fuzzy sets that define a weighted *undesired direction* (*UD*)
2. Fuzzifying the *DA* and mapping it to a weighted general *desired direction* (*DD*)
3. Combining the *DD* and the *UD* of movement to yield a weighted *control direction* (*CD*)
4. Defuzzifying the resultant *CD* and determining the crisp rudder control angle $\theta(t)$

Determining Desired Direction

The *DD* is a fuzzy region with a domain over angles in the general direction of desired travel. The membership values indicate the weighted preference of a particular direction. As shown in Fig. 17, *DD* is determined by first fuzzifying the crisp *DA*. This *fuzzy input* is then used with the following rules to compute the *DD fuzzy output*:

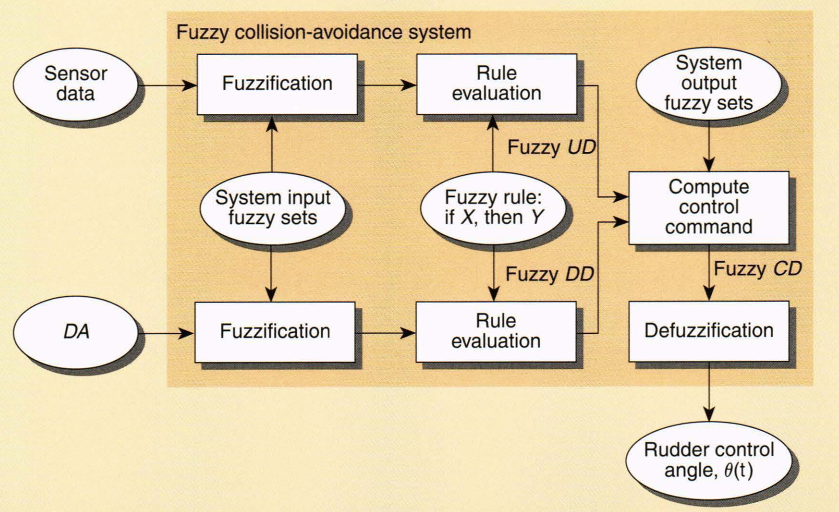


Figure 17. Data flow for the fuzzy collision-avoidance system. The sensor data and *desired angle* (*DA*) are input from the simulation. The sensor data consist of the closest obstacle's distance and closing rate for each sensor. The *DA* is the difference between the vessel's current course $\beta(t)$ and the desired course $\beta_{pc}(t)$. The output is the rudder control angle $\theta(t)$, which drives the simulation to the next step. (*UD* = *undesired direction*, *DD* = *desired direction*, and *CD* = *control direction*.)

If *DA* is BR, then *DD* is BR_{DD}.

If *DA* is SR, then *DD* is SR_{DD}.

If *DA* is ZE, then *DD* is ZE_{DD}.

If *DA* is SL, then *DD* is SL_{DD}.

If *DA* is BL, then *DD* is BL_{DD}.

Here, *DD* is categorized into the following five system output fuzzy sets: BIG RIGHT (BR_{DD}), SMALL RIGHT (SR_{DD}), ZERO (ZE_{DD}), SMALL LEFT (SL_{DD}), and BIG LEFT (BL_{DD}). These fuzzy sets and their membership functions are illustrated in Fig. 18. (The system input fuzzy sets for *DA* were illustrated in Fig. 7.) An example of the resultant fuzzy region defined by this mapping for a crisp *DA* of 38° is shown in Fig. 19, which illustrates a mapping of a fuzzily defined prescription of the motion direction to a more general (fuzzier) concept of a *DD*.

Determining Undesired Direction

The *UD* is a fuzzy region defining a weighted risk involved in choosing a direction of travel over the universe of possible travel directions. This directional risk is based on the *obstacle distance* (*OD*) and *obstacle closing rate* (*OC*) of the closest obstacle on the −90, −45, 0, 45, and 90° sensors. The *OD* and *OC* are fuzzified by mapping their crisp values to the system input fuzzy sets shown in Figs. 20a and 20b, respectively. These map into the fuzzy output *UD* using the following fuzzy rules:

If −90°*OD* is NEAR, then *UD* is BL_{UD}.

If −90° *OC* is CLOSING, then *UD* is BL_{UD}.

If −45°*OD* is NEAR, then *UD* is SL_{UD}.

If −45° *OC* is CLOSING, then *UD* is SL_{UD}.

If 0° *OD* is NEAR, then *UD* is ZE_{UD}.

If 0°*OC* is CLOSING, then *UD* is ZE_{UD}.

If 45°*OD* is NEAR, then *UD* is SR_{UD}.

If 45° *OC* is CLOSING, then *UD* is SR_{UD}.

If 90°*OD* is NEAR, then *UD* is BR_{UD}.

If 90° *OC* is CLOSING, then *UD* is BR_{UD}.

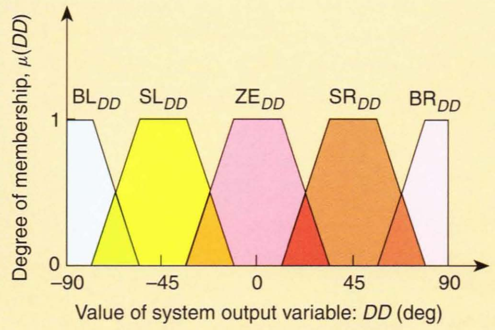


Figure 18. System output fuzzy sets for *desired direction* (*DD*), which is defined by five fuzzy sets: BIG LEFT (BL_{DD}), SMALL LEFT (SL_{DD}), ZERO (ZE_{DD}), SMALL RIGHT (SR_{DD}), and BIG RIGHT (BR_{DD}).

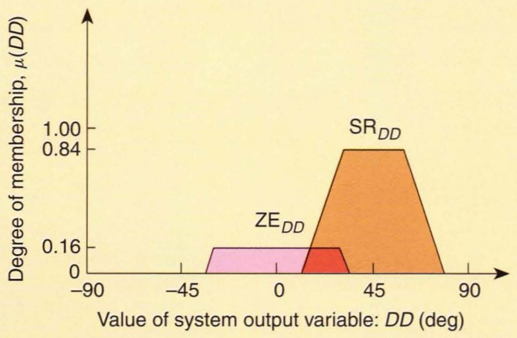


Figure 19. Fuzzy region defining *desired direction* (*DD*) for a crisp *desired angle* (*DA*) of 38°.

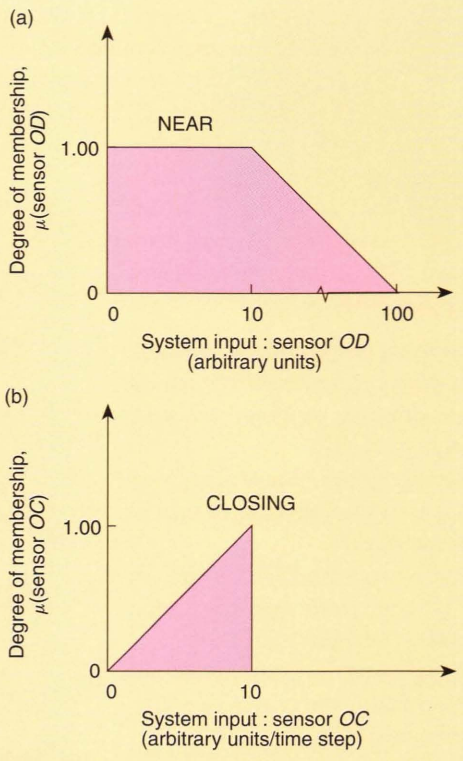


Figure 20. System input fuzzy sets for *obstacle distance* (*OD*) and *obstacle closing rate* (*OC*). (a) The *OD* is described by one fuzzy set NEAR with a universe of discourse [0, 100]. (b) The *OC* is described by one fuzzy set CLOSING with a universe of discourse [−10, 10].

Table 3. Example sensor obstacle data for *obstacle distance* (OD) and *obstacle closing rate* (OC) for the nearest obstacle detected on each sensor. The corresponding *fuzzy input* for each is listed. The last two columns are the mapping of these inputs to the output *undesired direction* (UD).

Sensor angle (deg)	*OD* (arbitrary units)	OC (arbitrary units/ time step)	μ_{NEAR} (degree of membership)	$\mu_{CLOSING}$ (degree of membership)	μ_{UD} (degree of membership)	*UD* fuzzy sets
−90	20	−2	0.89	0.00	0.89	BL
−45	10	0	1.00	0.00	1.00	SL
0	50	6	0.56	0.60	1.00	ZE
45	120	−5	0.00	0.00	0.00	SR
90	100	3	0.00	0.30	0.30	BR

Here, the *UD* is defined by the five fuzzy sets BIG RIGHT (BR_{UD}), SMALL RIGHT (SR_{UD}), ZERO (ZE_{UD}), SMALL LEFT (SL_{UD}), and BIG LEFT (BL_{UD}). The underlying membership functions are identical with those defined for *DD* in Fig. 18.

The rule evaluation process is illustrated in Table 3. For each of the five sensors, the nearest *OD* and *OC* (negative for opening) are listed. The next two columns labeled μ_{NEAR} and $\mu_{CLOSING}$ are the fuzzified *OD* and OC. The sixth column lists the *fuzzy output* for the *UD* using the fuzzy additive method for the rule evaluation, with corresponding fuzzy sets shown in the last column. The resulting fuzzy region is illustrated in Fig. 21a.

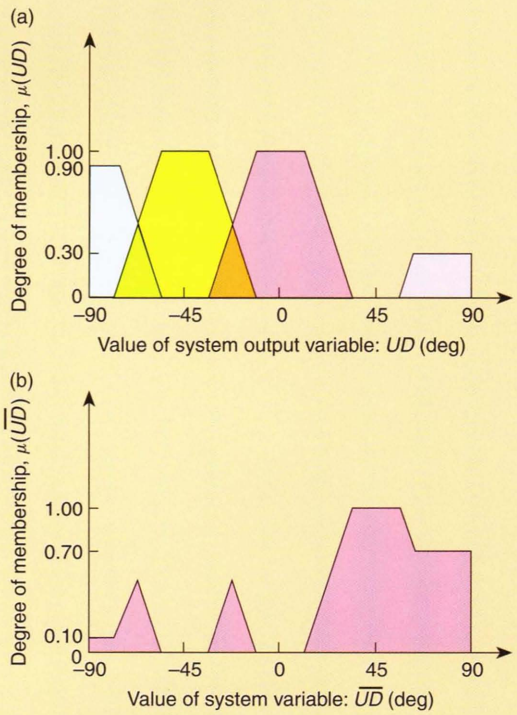


Figure 21. (a) Fuzzy region defining the *undesired direction* (*UD*) of travel for the illustative sensor obstacle data listed in Table 1. (b) Fuzzy region defining the complement of UD, $\overline{UD}$.

Determining Control Direction

The *CD* is a fuzzy region defined by combining the fuzzy *DD* and *UD*. This process is performed by first taking the complement of *UD* to obtain $\overline{UD}$. For example, the complement of *UD* in Fig. 21a is shown in Fig. 21b. The fuzzy *CD* is computed as the intersection of $\overline{UD}$ with *DD*. As defined in Fig. 5, the intersection is calculated by taking the minimum of the degree of membership of $\overline{UD}$ and *DD* at each angle element. This operation is illustrated in Fig. 22.

This procedure illustrates the evaluation of an unconditional rule, or one that is not qualified by an 'if' statement. The rule here is defined as '*CD* is *DD* and $\overline{UD}$,' which serves to restrict the solution space for the control direction to the maximum of the intersecting regions.

Defuzzifying the Results

A crisp *CD* is defined through defuzzification. Defuzzification is performed by calculating the COG of the fuzzy *CD* region, as illustrated in Fig. 22. If *CD* has disjoint regions, the COG is calculated for the region

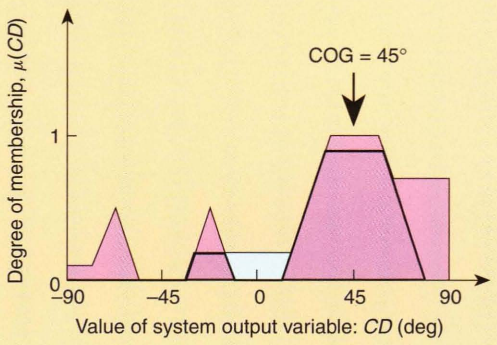


Figure 22. The fuzzy region defining *CD* is the intersection of the *DD* and $\overline{UD}$ fuzzy regions. Defuzzification computes a crisp *CD* through calculation of the COG of the area with the greatest mass.

with maximum mass. The rudder control angle $\theta(t)$ required to move in this direction is equal to the control angle.

DISCUSSION AND STATUS

The objectives of this project were the development of an algorithm employing fuzzy logic, the implementation of general-purpose fuzzy logic software, and the demonstration of a model. We developed an algorithm in this article for employing fuzzy logic, and general-purpose software has been implemented using the method described. The demonstration problem, the ship collision-avoidance problem, was defined, and a simplified simulation was devised for testing a fuzzy collision-avoidance solution. Further, a fuzzy solution for this problem was created. The fuzzy ship collision-avoidance solution has been implemented as defined, and its effectiveness in preventing ships from colliding with stationary and moving obstacles has been demonstrated through a simulation. The general software and collision-avoidance simulation are available on a diskette included with Ref. 17.

The modeled ship collision-avoidance fuzzy solution and simulation were greatly simplified in comparison with a real-world collision-avoidance system, but the real-world solution could be implemented using this model as a basis. For example, more realistic rules could replace the simplified rules here to account for factors such as obstacle's closest time of approach and closest distance of approach. Further, as suggested by Charles H. Sinex, a high-fidelity ship simulation could replace the simplified kinematic equations used in this model. A fuzzy model of the mariner's steering control could then be added to create a fuzzy steering control system that would account for the nonlinear effects in changing the rudder angle.

REFERENCES

1 Zadeh, L., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Syst. Man Cybern.* **SMC-3**(1), 28–44 (1973).
2 Zadeh, L., "Fuzzy Sets," *Inform. Contr.* **8**, 338–352 (1965).
3 Black, M., "Vagueness: An Exercise in Logical Analysis," *Philos. Sci.* **4**, 427–455 (1937).
4 Rescher, N., *Many-Valued Logic*, McGraw-Hill, New York, pp. 1–16, 22–28 (1969).
5 McNeill, D., and Freiberger, P., *Fuzzy Logic*, Simon & Schuster, New York, pp. 36–38 (1993).
6 Klir, G., and Folger, R., *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, NJ, pp. 37–61 (1988).
7 Viot, G., "Fuzzy Logic in C," *Dr. Dobb's J.* **18**(2), 40–49 (1993).
8 Cox, E., "Fuzzy Fundamentals," *IEEE Spectrum* **29**(2), 58–61 (1992).
9 Cox, E., *The Fuzzy Systems Handbook*, AP Professional, Cambridge, MA (1994).
10 Dove, M., Burns, R., and Stockel, C., "An Automatic Collision Avoidance and Guidance System for Marine Vehicles in Confined Waters," *J. Navig.* **39**(2), 180–190 (1986).
11 Coenen, F., Smeaton, G., and Bole, A., "Knowledge-Based Collision Avoidance," *J. Navig.* **42**(1), 107–116 (1989).
12 Jingsong, Z., Fengchen, W., and Zhao-lin, W., "The Development of Ship Collision Avoidance Automation," *J. Navig.* **45**(1), 107–113 (1992).
13 Iijima, Y., and Hagiwara, H., "Results of Collision Avoidance Manoeuvre Experiments Using a Knowledge-Based Autonomous Piloting System," *J. Navig.* **44**(2), 194–204 (1991).
14 James, M., "Modelling the Decision Process in Computer Simulation of Ship Navigation," *J. Navig.* **39**(1), 32–48 (1986).
15 Kong, S., and Kosko, B., "Adaptive Fuzzy Systems for Backing up a Truck-and-Trailer," *IEEE Trans. Neur. Net.* **3**(2), 211–223 (1992).
16 Yen, J., and Pflunger, N., "Designing an Adaptive Path Execution System," in *Proc. IEEE Int. Conf. Syst. Man Cybern.* **3**, 1459–1464 (1991).
17 Quaranta, T., *The Fuzzy Development Model*, JHU/APL FS-94-113 (1994).

ACKNOWLEDGMENTS: I would like to thank Charles H. Sinex for his help, advice, and ideas given to me unconditionally over the past year. His efforts and support have been invaluable to the development of this project.

THE AUTHOR

THERESE F. QUARANTA is an electrical engineer in the Fleet Systems Department. She obtained a B.S. in mechanical engineering from the Pennsylvania State University and an M.S. in electrical engineering from The Johns Hopkins University. She joined APL in 1990 after working for the Atlantic Research Corporation and David Taylor Research Center. Her interests include adaptive methods for control, signal processing, information fusion and management, prediction, and decision making. She is currently investigating potential projects at APL to continue her work on fuzzy systems.