Using Brightness and Saturation to Visualize Belief and Uncertainty

Joseph J. Pfeiffer, Jr.

Department of Computer Science
New Mexico State University
Las Cruces, NM 88003 USA
pfeiffer@cs.nmsu.edu

Abstract. In developing a visual language for mobile robots, it is necessary to represent the uncertainty present in robot position, obstacle position, and even obstacle presence. In developing a visualization of the robot's model of its environment, this uncertainty should be presented to the experimenter, in order to be able to evaluate the extent to which the robot's sensors and sensor fusion rules are providing consistent and reliable information.
In Isaac, a project developing a rule-based visual language for mobile robots, a time-varying diagram is used to represent the robot's current world model. Hue is used to represent object classes, and brightness is used to represent the degree of belief of an object's presence. A region in which there is confidence that no object is present is shown as white, while a region with high confidence in the presence of an object is represented with color. Saturation is used to represent confidence in the assessment of object presence (or absence): a totally unsaturated (*i.e.* grey) area represents an area in which there is no belief at all either in favor of or against the presence of any object; a fully saturated area represents an area in which there is high confidence in the region's classification. The combination of hue to distinguish between object classes with brightness and saturation for belief and confidence results in a three-dimensional color space for model visualization.
Sensor characteristics are encoded in belief functions; upon receiving sensor information, both belief functions and confidence levels can be modified. Belief functions in the presence and absence of obstacles in the model are maintained through Dempster-Shafer evidential reasoning.

1 Introduction

An important consideration in programming a mobile robot is having the capability of visualizing the robot's model of its environment. The visualization includes aspects such as the robot's location and orientation relative to environment features known *a priori*, new features and obstacles which have been discovered by the robot in the course of exploring and interacting with its environment, and pseudo-objects added to the environment model such as markers identifying paths which have been explored and do not need to be re-examined.

A well-known characteristic of a robot's environment model is that information is not perfect. Sensors return spurious values (indicating the presence of nonexistent objects), return inexact values (giving rise to incorrect estimates of either object or robot location), and fail to return values at all (failing to recognize the presence of objects). Modifications to the model must take these uncertainties into account, and a visualization should also display it. Also, since objects in the model may overlap, the visualization must display that as well. In this paper, we discuss the use of saturation to represent uncertainty in the Isaac mobile robotic environment. Following this Introduction, Section 2 will briefly describe the Isaac geometric reasoning language. Section 3 will review Dempster-Shafer theory and describe its application and visualization in Isaac. Section 4 describes related work, and Section 5 will present some preliminary conclusions and future work.

2 Isaac

Isaac is a rule-based visual language for geometric reasoning, intended for the control of mobile robots[1]. As usual in rule-based languages, a rule has a left-hand side containing preconditions (facts which must be present to enable the rule), and a right-hand side with postconditions (changes which will be made to the environment as a result of executing the rule). A typical Isaac rule, implementing obstacle avoidance, is shown in Figure 1.

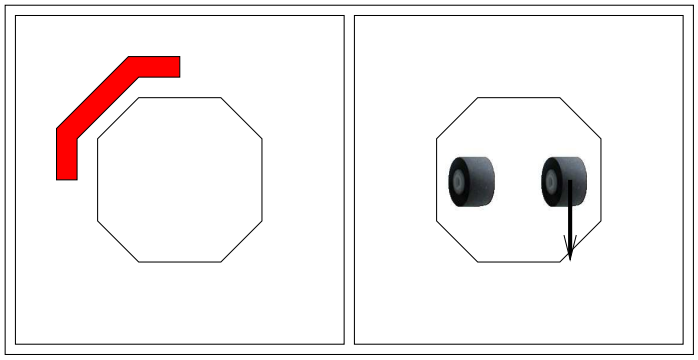


Fig. 1. Obstacle Avoidance Rule in Isaac

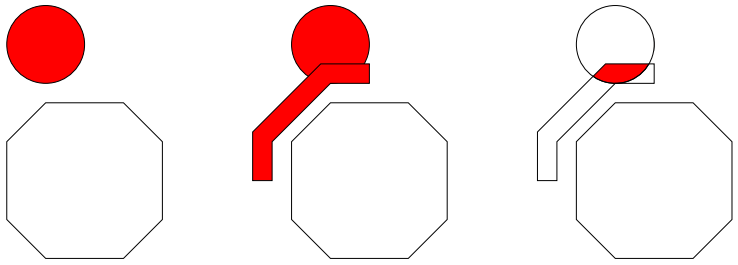
In this figure, the robot is shown as an octagon. The robot's direction of travel is upward in the figure. The region forward and to the left of the robot in the precondition is an avoidance region (in the actual environment and in the online version of this paper this region is colored red); if this region has a non-zero intersection with a similarly colored region in the robot's current environment model then the rule is enabled. The postcondition shows the result of activating

the rule: two new objects (icons representing wheels, with the left wheel stopped and the right wheel turning in reverse) will be inserted into the environment. These objects represent actions to be taken by the robot; these are specialized output objects which will actuate the motors as specified[2].

For purposes of this introduction, objects in the world model are represented as fully-saturated geometric objects. This will be generalized in Section 3.2; in addition to hue, objects will also have variable saturation and brightness.

2.1 Rule Enabling and Application

Figure 2 shows a typical application of the rule shown in Figure 1.



(a) Obstacle in Environment (b) Obstacle and Rule (c) Intersection Area

Fig. 2. Rule Enabling in Isaac

In Figure 2(a), the robot is shown approaching an obstacle, and Figure 2(b) shows the obstacle avoidance rule precondition in combination with the environment. Finally, Figure 2(c) shows the intersection of the precondition with the obstacle. As the intersection is non-empty, the rule is enabled. Color is significant for rule enabling; a rule precondition is intersected only with objects in the environment with the same hue as the precondition (in the actual system, and in the on-line version of this paper, the object and the precondition are both red).

2.2 Rule Combination

Situations frequently arise in which more than one rule is enabled simultaneously. In these circumstances, conflicts between the rules must be resolved in order to select a course of action.

Isaac uses a weighted average to combine the rules. Rules have weights associated with them; when several rules are activated simultaneously and have incompatible right-hand sides, the result is the weighted average of the right-hand sides of the enabled rules.

As an example, consider a situation in which a robot is following a planned path, but must avoid an obstacle. The situation is shown in Figure 3. The straight line is the planned path; it is in the model but not in the actual environment. The area with both path and obstacle is shown in a striped pattern.

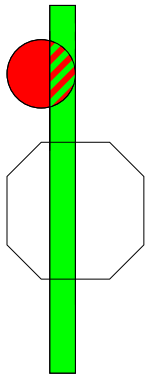


Fig. 3. Robot Following Path in Presence of Obstacle

In this situation, the robot has veered slightly to the right of the planned path, and the rule shown in Figure 4 will be activated to bring the robot back onto the path. This rule's postcondition stops the left motor while driving the right motor forward (in the actual implementation, and the on-line version of this paper, the path and corresponding object in the rule are shown in green).

The obstacle is also shown, as described above; consequently the object avoidance rule from Figure 1 is also active. These two active rules both call for the left motor to be stopped; the course correction rule calls for the right motor to go forward, while the obstacle avoidance rule calls for the right motor to go back.

In order to resolve the competing rule postconditions, weights are assigned to the rules.[1] A reasonable weighting for the two rules described here might be 1 for the line following and 10 for the obstacle avoidance; in this case, the net

[1] This is a change from previous descriptions of Isaac. In the original conception, rule weighting was defined in terms of the area of the matching rule precondition, as described in [1]. Our intent in using this definition was to avoid the necessity of defining rule weights; unfortunately, the effect was to make anticipating rule interaction nearly impossible.

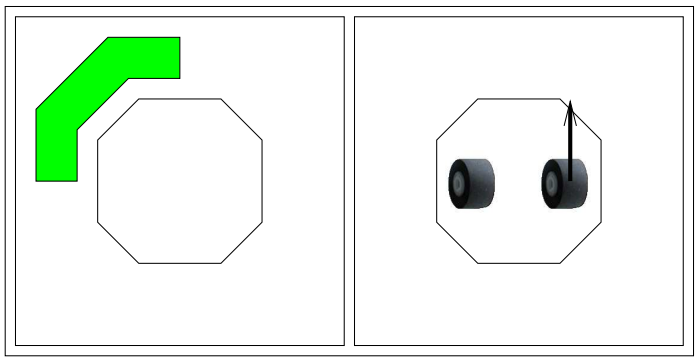


Fig. 4. Path-Following Rule

result would be that the left motor would stop (as both rules call for this), while the right motor would be given a value of $\frac{(1.0)(-1)+(.1)(1)}{1.0+.1} = -.82$

3 Uncertainty

The example in the previous section assumes perfect knowledge: knowledge of the location of the robot, and knowledge of both the presence and location of the obstacle. Neither of these is typically known in an actual environment. Instead, sensors provide data that is both inaccurate and unreliable. Sonar, in particular, is prone to false returns, failure to reliably generate echos on some substances, and spreading. Consequently, we can only regard sensor inputs as evidence (rather than certain knowledge) of possible features, and lack of input as evidence of lack of features. The solution to this will be to respond to sensor returns by putting objects in the model with a size and uncertainty derived from the characteristics of the sensors themselves. Multiple sensor returns will be fused to form a coherent picture of the environment.

3.1 Dempster-Shafer Belief Functions

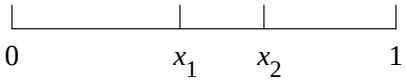
Dempster-Shafer theory, and particularly Dempster's rule of combination, provides a means of explicitly maintaining uncertainty and combining evidence from multiple sources[3]. In Dempster-Shafer theory, propositions are represented as subsets from a set Θ of mutually exclusive alternatives for a parameter (referred to as a *frame of discernment*). In our case, the only two alternatives are the presence or absence of an object of a given color at a location in the world model, and the subsets are $\{\phi, T, F, \Theta\}$ where ϕ is the empty set, T and F are the presence or absence of an object, and Θ is $T \cup F$. A belief function $Bel\,(\theta)$ is used to represent the degree of belief in each of the subsets, where $Bel\,(\theta)$ must satisfy the following conditions:

1. $Bel\,(\phi) = 0$

2. $0 \leq Bel(T), Bel(F) \leq 1$
3. $Bel(\Theta) = 1$

A point which is implicit in this definition are that the sum of the belief in the two alternatives can be no greater than 1 (as $Bel(\Theta) = 1$), but can be less than 1, allowing some portion of the total belief to be "unallocated."

Dempster's rule of combination provides a means of updating these belief functions in the presence of new evidence. The interval $[0, 1]$ is divided into three parts according to the belief functions:



where $x_1 = Bel(T)$ and $x_2 - x_1 = Bel(F)$. The remainder of the interval is the unallocated portion of the belief. To combine two belief functions, the line segments representing their belief functions are combined as in Figure 5. The relative areas of the nine regions of the figure provide the updated belief function.

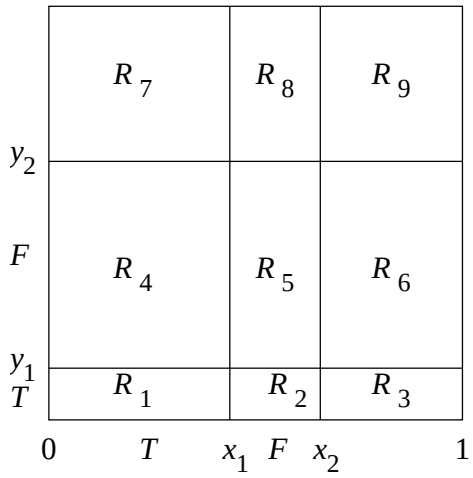


Fig. 5. Orthogonal Combination of Belief Functions

Upon combining the two functions, we have a total of nine regions. In the result, region R_9 reflects that part of the belief function which remains unallocated. The combined areas of $R_1 + R_3 + R_7$ represents the new belief assigned to T; the combined areas of $R_5 + R_6 + R_8$ represents the new belief assigned to F. Regions R_2 and R_4 are incompatible assignments (they represent a belief that both of the mutually exclusive alternatives are true), so the new values are normalized by dividing them by the total area of the compatible assignments, $1 - (R_2 + R_4)$.

As an example, consider a situation in which we presently believe an area to be clear of obstacles with certainty 0.7. For compactness, we will represent a belief assignment with a tuple $(Bel(T), Bel(F), 1-(Bel(T)+Bel(F)))$. The third component in the tuple is redundant, however we prefer to show the unallocated belief explicitly. The belief assignment in this case is $(0, 0.7, 0.3)$. Now assume a sensor input is received indicating that there is an obstacle in the region; we have a confidence of 0.8 in this sensor, so its belief assignment is $(0.8, 0, 0.2)$. Combining the sensor input with our previous belief function, the new belief assignment is $(0.14, 0.54, 0.32)$.

3.2 Visualizing Uncertainty

In visualizing the belief function, we map $Bel(T)$ to saturation and $Bel(F)$ to brightness. The $Bel(T)$ mapping uses the full range of saturation from 0 to 1; the $Bel(F)$ mapping only uses brightnesses from 0.5 to 1, in order that darker lines can be used for emphasis. Combined with hue for object classification, this results in a three-dimensional (H, S, B) color space. The mapping of belief and uncertainty to a color is shown in Figure 6.

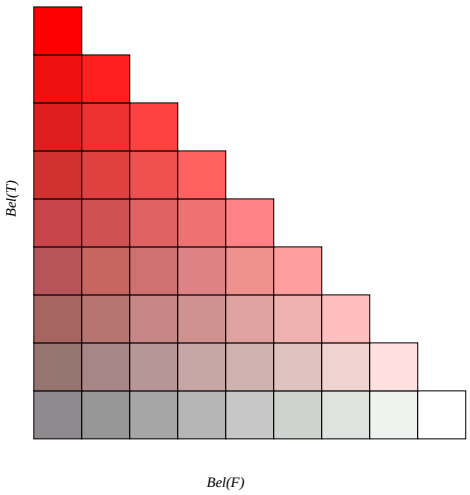


Fig. 6. Visualization of Belief and Certainty for Object

In this figure, the three corners have the following interpretation: the lower left corner is the visualization of a region whose contents are completely unknown. The saturation is 0, and the brightness is 0.5. The top left corner, with $Bel(T) = 1$ and $Bel(F) = 0$, has a saturation of 1 and brightness of 0.5. Finally, the lower right corner, with $Bel(T) = 0$ and $Bel(F) = 1$, has a brightness of 1. The figure is triangular, reflecting the fact that $Bel(T) + Bel(F) \leq 1$. In the on-line version of this paper, the upper corner is red.

The saturation of an object with a brightness of 1 is always 0, and its hue is undefined. This does not cause an inconsistency in Isaac, as Isaac rule preconditions can only be conditioned on the presence of an object and never on its absence.

3.3 Sensors and Sensor Rules

The rules exhibited in Section 2 are one of the three types of rules available in Isaac: actuator rules. Sensor rules also exist for adding objects to the environment model as a result of sensor input, and deduction rules are able to use objects currently in the environment to perform operations such as path planning.

Isaac's response to a sensor input is the invocation of an "input rule" to place an object in the robot's world model at a location determined by the robot's position and the sensor's parameters[2][4]. Due to the uncertainty in the sensor return, the object's location and extent are also uncertain. Consider a rangefinding sensor such as sonar. A reading from such a sensor indicates two things: that there is an object at the indicated distance and location, and that there is no object closer (in that direction). A rule representing this for a sensor with a 30° spread, and with a belief assignment of $(0, .5, .5)$ for the close region and $(.5, 0, .5)$ for the obstacle region would appear as in Figure 7.

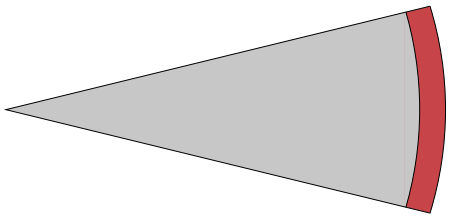


Fig. 7. Example Sensor Rule

A simulation has been developed to show the results of a series of sonar readings taken as a robot traverses a model room. The room, diagrammed in Figure 8, is a simulated 5 meters on a side with a single 1.25 meter square obstacle in its center. In this simulation, the robot proceeds across the left-hand wall while taking readings to the right. The robot moves one centimeter forward after taking each reading.

For purposes of this simulation, we use a belief assignment of (0, .1, .9) for the close region and (.1, 0, .9) for the obstacle region. For a first example, the sensor return is assumed to be perfect; the range returned is exactly the range to the nearest object within the sensor's spread. The results of the simulation are displayed in figure 9.

Initially, we assume no information regarding the room contents, so the environment model is a uniform grey representing a belief assignment of $(0, 0, 1)$.

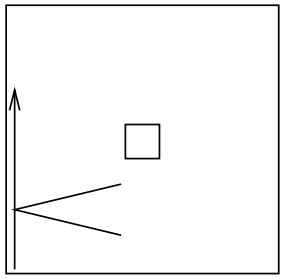
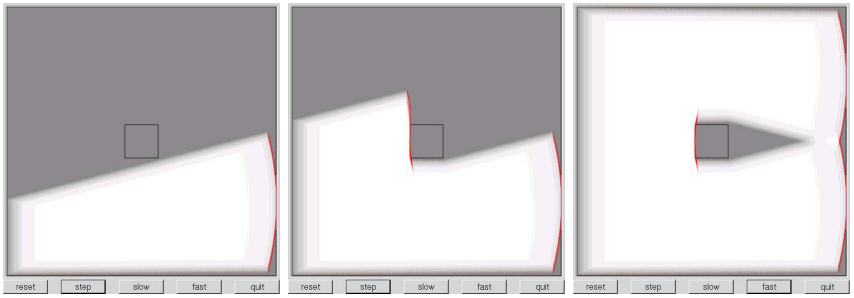


Fig. 8. Room to be Explored



(a) Room Model After 150 Scans

(b) Model After 300 Scans

(c) Final Room Model

Fig. 9. Simulation of robot exploring room

After each sonar reading, the environment model is modified according to Dempster's Rule, and the robot proceeds forward. Following 150 readings, the appearance of the room model is as shown in Figure 9(a). In the interest of clarity, the outlines of the room and the obstacle are included in this Figure (even though it is not actually part of the model). In the simulation, and in the on-line version of this paper, a red arc is visible at the far extent of the white area, coincident with the far wall.

After another 150 readings, the room model will have changed as shown in Figure 9(b). At this point, the obstacle has also been located. More importantly from the perspective of this work, the extent to which the robot has belief in the presence or absence of obstacles, and the areas which have not yet been explored, are clearly visible.

The process continues as the robot makes its way across the room; the final model is shown in Figure 9(c). As the overlapping sensor returns have been fused, the system shows higher confidence in the lack of obstacles in those regions which have returned "empty" several times, and likewise shows greater confidence in the presence of obstacles in the regions that have had multiple echo returns.

3.4 Uncertainty and Rule Activation

In Section 2.1, rule enabling is presented as the intersection between the robot's environment model and the preconditions of a rule. In the presence of uncertainty, this is modified: a rule is enabled to the extent that an object is believed to be present in the area defined by a rule's precondition. Objects in which the robot has only a slight belief have only a slight effect on the robot's behavior.

4 Related Work

The previous work most closely related to this is Anderson's Inter-Diagrammatic Reasoning[5][6]. As with this work, Anderson's diagrams divide the plain into polygons with common properties (he refers to this as tesselating the plain); the properties of these tesserae are represented using color. In his most developed version of the theory, these colors are selected from a cyan-magenta-yellow subtractive color space. Operations are defined for manipulating diagrams; these are gemerally similar in nature to fuzzy logic operations (for instance, the intersection of two diagrams is defined by taking the minimum of each of the three color coordinates at each point in the plain).

Inter-diagrammatic reasoning is more purely a diagrammatic reasoning system than Isaac's uncertainty visualization. Anderson defines a semantics of operations on diagrams based on color, while this work's use of color is only as a visualization tool. It would be possible to redefine Dempster's Rule in terms of operations on the saturation and brightness of the polygons, however, this would not be fruitful.

Dempster-Shafer theory has been used in robot localization (with a much more detailed sonal model than described here) by [7]. This paper contains figures representing belief functions from sonar returns; the authors use a series of figures (one figure for presence, one for absence, and one for unallocated belief) for a single belief assignment, with intensity representing a component of the belief assignment.

5 Conclusions

We have described a visualization of belief functions for uncertain geometric reasoning. This visualization is well-suited for use with Isaac's rule processing mechanism, as it extends the previously one-dimensional color space used by Isaac (hue) into a three-dimensional space better making use of the capabilities of the display device.

Our present work is focused on creating more detailed simulations of Isaac's behavior on mobile robots, developing models of our sensors in order to define rules using them, and merging models produced by several robots in a single, global model.

References

1. Pfeiffer, Jr., J.J.: A language for geometric reasoning in mobile robots. In: Proceedings of the IEEE Symposium on Visual Languages, Tokyo, Japan (1999) 164–171
2. Pfeiffer, Jr., J.J., Vinyard, Jr., R.L., Margolis, B.: A common framework for input, processing, and output in a rule-based visual language. In: Proceedings of the IEEE Symposium on Visual Languages, Seattle, Washington, USA (2000) 217–224
3. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press (1976)
4. Vinyard, Jr., R.L., Pfeiffer, Jr., J.J., Margolis, B.: Hardware abstraction in a visual programming environment. In: Proceedings of the International Multiconference on Systemics, Cybernetics, and Informatics, Orlando, Florida, USA (2000)
5. Anderson, M., McCartney, R.: Inter-diagrammatic reasoning. In: Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, Canada (1995)
6. Anderson, M., Armen, C.: Diagrammatic reasoning and color. In: Proceedings of the 1998 AAAI Fall Symposium on Formalization of Reasoning with Visual and Diagrammatic Representations, Orlando, Florida (1998)
7. Hughes, K., Murphy, R.: Ultrasonic robot localization using dempster-shafer theory. In: SPIE Neural and Stochastic Methods in Image and Signal Processing, San Diego, CA, Society of Photo-Optical Instrumentation Engineers, Society of Photo-Optical Instrumentation Engineers (1992) 2–11