AN ERROR ANALYSIS OF THE MOTION OF A VEHICULAR ROBOT IN 2-D

by

Gordon C. Fossum

Memorandum No. UCB/ERL M81/13

13 March 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

AN ERROR ANALYSIS OF THE MOTION OF A VEHICULAR ROBOT IN 2-D

by Gordon C. Fossum

INTRODUCTION.

In order to do interesting things with a self-propelled robot, it would be nice if we could predict where the robot might stop after executing a movement, with the hope that the area thus defined is somehow "close" to where the robot is supposed to be.

This paper defines a robot with a single executable instruction with two degrees of freedom and describes what is presumed to be a reasonably universal set of mutually independent sources of errors. The result of each error is investigated; then these independent results are gradually combined to yield a composite closed curve which encloses all possible stopping points of the robot after executing its instruction.

The curve just described is dependent on several factors: the pre-instruction position and orientation of the robot, the two independent parameters of the instruction itself, and the limiting values of each error source. The "reserved" variables used in this paper, in order of appearance, are: t, d, e1, k1, e2, k2, e3, k3, and c.

ROBOT DEFINITION.

The robot may be viewed as a creature capable of moving in straight lines and "turning on a dime" so that, in executing a turn, its center of mass remains stationary (all assertions regarding where the robot "is" or "is not" refer to its center of mass, so the presumption of a "point robot" may be appropriate). The only instruction the robot can execute is of the form

ROTATE t AND MOVE d METERS"

where t and d are real numbers satisfying $-180<t\le 180$ and $d>0$.

ERRORS.

There are seven error sources considered in this paper. They are treated as being independent of each other (such that changing the "severity" of any one of them has no effect on the severity of the others). They are examined singly (presuming that only one of them at a time has a non-zero value) to determine the "error area" each one can spawn. The errors are

ABSOLUTE ROTATIONAL SKID
RELATIVE ROTATIONAL SKID
ABSOLUTE ANGULAR ERROR
RELATIVE ANGULAR ERROR
ABSOLUTE PATH LENGTH ERROR
RELATIVE PATH LENGTH ERROR
PATH NON-LINEARITY ERROR.

In all figures contained in this paper, the "correct" translational motion of the robot is toward the top of the page, which is to say that whatever the original orientation and angle of rotation prior to the translational motion, the "clever cameraperson" chose the appropriate angle to shoot from.

1. ABSOLUTE ROTATIONAL SKID. Variable name=e1 Units=meters.

The first maneuver in the execution of the instruction is a rotation. In any real machine, the act of starting and stopping the rotation will cause the center of mass to move. The variable e1 represents the maximum (radial) distance it can "skid" due to starting and stopping, independent of the magnitude of the rotation. If this error acts alone, the robot, upon completion of its entire maneuver would stop somewhere in a circle of radius e1, centered on its intended destination (see Figure A).

2. RELATIVE ROTATIONAL SKID. Variable name=k1 Units=meters/degree.

Some skidding is caused by the rotation itself and will be proportional to the magnitude of the rotation (unlike the previous error). The variable k1 is the maximum constant of proportionality which can be encountered. This error, acting alone, will cause the robot to stop inside a circle of radius (k1)(|t|) centered on the intended destination (see Figure B).

3. ABSOLUTE ANGULAR ERROR. Variable name=e2 Units=degrees.

The act of starting and stopping the rotation will cause some loss in what the true heading of the robot should be. Roundoff error in calculating the rotation angle will also be included here. The variable e2 bounds these errors. Acting alone, e2 causes the robot to stop on (not inside) the circle of radius d, centered on the starting point, such that it is no more than e2 degrees of arc away from its intended destination (see Figure C).

4. RELATIVE ANGULAR ERROR. Variable name=k2 (unitless).

Some inaccuracy is to be expected somewhere in the linkage of the rotational mechanism. The variable k2 bounds this proportional inaccuracy. This error causes the robot to stop on the same circle as that of the previous error, such that it is no more than (k2)(|t|) degrees of arc away from its intended destination (see Figure D).

The first four errors above were caused by the rotation. The last three are caused by the "translation," or linear motion of the robot.

5. ABSOLUTE PATH LENGTH ERROR. Variable name=e3 Units=meters.

The act of starting and stopping the linear motion will cause some error along the direction of motion. Roundoff error in calculating the path length will also be included here. The variable e3 bounds these errors. Acting alone, e3 causes the robot to stop on the line connecting the starting and stoppng points within e3 meters of the intended stopping point (see Figure E).

6. RELATIVE PATH LENGTH ERROR. Variable name=k3 (unitless).

Some inaccuracy will exist in the linkages of the translational mechanism. This error, proportional to the path length d, is bounded by the constant of proportionality k3, so that the robot will stop on the same line as that of the previous error within (k3)(d) of the intended stopping point (see Figure F).

7. PATH NON-LINEARITY ERROR. Variable name=c Units=1/meters.

A variety of factors will cause the robot to follow some path which is not, in fact, linear. The variable c is the maximum curvature which the robot's path can exhibit. The reciprocal of c is the radius of the circle which the robot would trace if it exhibited worst case behavior. Thus if the value of c were 0.05, the robot could conceivably move in a circle of radius 20 meters. (Throughout the following paragraph, refer to Figure G.)

This error is the tough one to analyze because it allows the robot such freedom. The robot's path can be visualized as a rope of length d, immovably anchored at one end (the starting point) and projecting initially in a certain direction, along some flat floor. The rope is only slightly flexible, such that it can be bent to a curvature of magnitude c at any point, but not more. the job is to describe the closed curve in 2-space such that points on and inside the curve curve represent possible positions of the free end of the rope, and points outside the curve simply can't be reached by the free end of the rope. Three extreme points can be shown to be on the curve immediately. The intended destination is on the curve, because, in some sense, you can't go "farther" than that point from the starting point. Two other points on the curve are those arrived at by proceeding "hard to port" or "hard to starboard" from the starting point, for the entire path length d. The "hard to port" point is located (are you ready?) on a circle of radius 1/c whose center is at a distance 1/c from the starting point, to the "left", along the line perpendicular to the intended path of the robot which contains the starting point such that the length of the arc along this circle from the starting point up to the "hard to port" point is d. The "hard to starboard" point is found in an analogous fashion.

These three points can be connected by a curve which can be viewed as being obtained by swinging the rope from "hard to port" through the intended destination to the "hard to starboard" point, keeping it as taut as possible at all times. A little reflection leads one to conclude that a snapshot of the rope at an arbitrary point on this journey will show the rope starting on a circular arc to some point and then continuing as a straight line (see Figure H). This curve forms the outer boundary to the error area, as there is no way for the robot to go beyond it (if all other errors are zero). Another curve is found by "pushing" the rope (perhaps by attaching a spring to connect the two ends of the rope) so as to minimize the radial distance between the two ends. A rope compressed in this fashion will assume an S shape built of two circular arcs, each of radius 1/c, as no other permitted configuration is as "short" in radial length. (At this point, however, a proof of this conjecture remains elusive.) This curve is not symmetric, however, and we must superimpose two versions of it (one arrived at by "peeling" the rope from left to right, the other from right to left) and take, as our inner boundary to the error area those portions "closest" to the starting point (see Figure I). These curves, then, define the area comprising the PATH NON-LINEARITY ERROR.

ERROR COMPOSITION.

The absolute and relative angular errors can be combined, resulting in an arch which is wider than either of them. Similarly, the absolute and relative path length errors can be combined. If all four of these errors are considered together, the resulting error area resembles the swath of a windshield wiper (see Figure J). This error area represents the final answer if we could assume that the robot really could "turn on a dime" and really did travel in straight lines.

For the most general picture, though, it's best to start with the path non-linearity error area (Figure I), and "fold in the other ingredients carefully."

First, add the path length errors in by considering that the path non-linearity error presumes a path of exact length. If we substitute for the "correct" path length the longest path length permitted by the two path length errors (taken together), we get a slightly larger path non-linearity error area which is positioned slightly farther away from the starting point. An analogous, smaller, closer area is derived from the shortest path length permitted by the path length errors. Now, recall that the "hard to port" and "hard to starboard" points were on the circles of tightest curvature that the robot could travel, but the curvature itself is independent of (and therefore constant throughout) the just-completed construction. Thus, as the path length varies from longest to shortest, these two points sweep out circular arcs, which, taken in union with the outer (upper) curve of the larger path nonlinearity error area and the inner (lower) curves of the smaller path non-linearity error area, define the new composite error area which specifies where the robot could be if paths of inexact length and imperfect linearity were permitted, but all rotations were presumed to be executed perfectly (see Figure K).

The angular errors can next be incorporated into this composite error area easily by just rotating the entire area about the starting point clockwise and counterclockwise to the maximum angle permitted by the angular errors taken together (-e2-(k2)(¦t¦) to +e2+(k2)(¦t¦)) (see Figure L).

The final composite error area is achieved by incorporating the "skid" errors. This is accomplished by stating simply that the new error area consists of all points which are within a distance of (e1+(k1)(¦t¦)) of some point on the previous error area. This production can be roughly visualized as painting a border of this width, with rounded corners, around the previous error area (see Figure M).

A MODIFICATION.

It's clear that the most "insidious" error here is the path nonlinearity error. In severe cases (in which the radius of curvature is significantly less than the path length "d"), the path nonlinearity error can cause highly undesirable results, which in the extreme, will cause the error area (the area in which the robot can stop) due to this error alone to be a circle of radius d centered on the starting point.

One way to limit this "wanderlust" is to install a compass in the robot. This instrument can be of two types: smart or dumb.

Type 1 (smart compass): This compass is kept informed of all rotation commands, and maintains an up-to-date record of where the robot SHOULD be headed and ensures that, between the rotational motion and the translational motion, any needed rotational corrections are made so that the robot's heading is initially correct to within the error of the compass itself (a small, constant error due to the operation of the compass). This compass also commands "mid-course corrections" whenever the robot veers from its initial heading by mote than some small angular tolerance in either direction (see Figure N).

Type 2 (dumb compass): This compass merely takes a reading at the instant the robot starts its translational motion (after the rotation is completed) and then commands midcourse corrections just like the smart compass. It has no knowledge of what the true course should have been, because it doesn't read the rotation commands.

Both of these compasses bound the path nonlinearity error "c" to a narrow range of possible headings determined by the pre-programmed tolerance of the compass and its error. (see Figure O. The error area shown in Figure O is only an estimate, due to the difficulties in incorporating "mid-course corrections.") Note, however, that while the dumb compass effectively adds this modified path nonlinearity error to the angular errors e2 and (k2)(|t|), the smart compass negates the effect of the angular errors by compensating for them as needed, and thus REPLACES them with this modified path nonlinearity error.

ACNOWLEDGEMENTS.

I would like to express my sincere appreciation to Manuel Blum, Professor, EECS Department, UC Berkeley, for his spirited interest and valuable guidance in this project.

My gratitude also goes to Dave Zielke, a student in the Math Department at Berkeley, whose constructive criticism resulted in non-trivial modifications to this paper.

Research sponsored by the National Science Foundation Grant MCS-79-03767-A01.

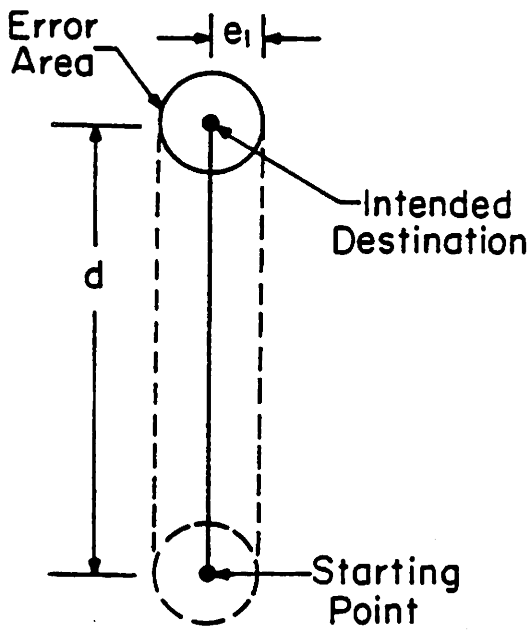

FIGURE A

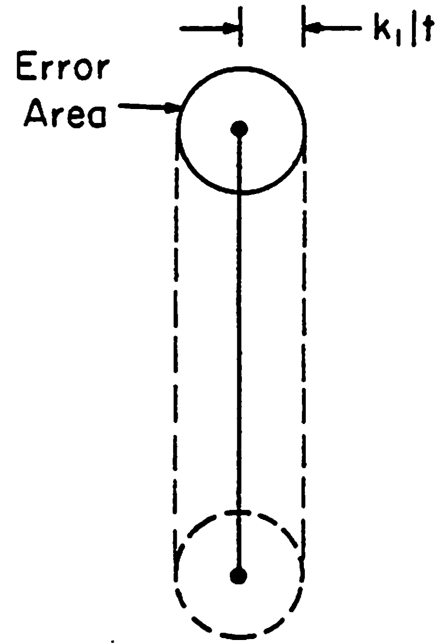

FIGURE B

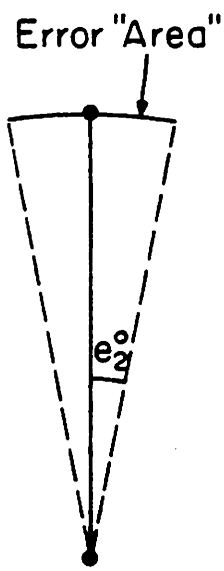

FIGURE C

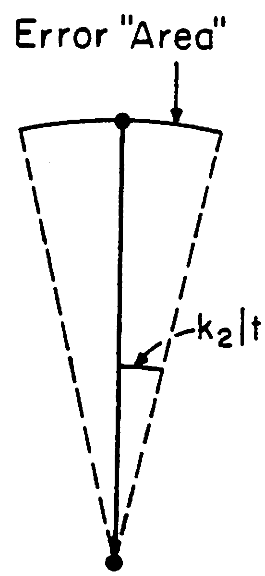

FIGURE D

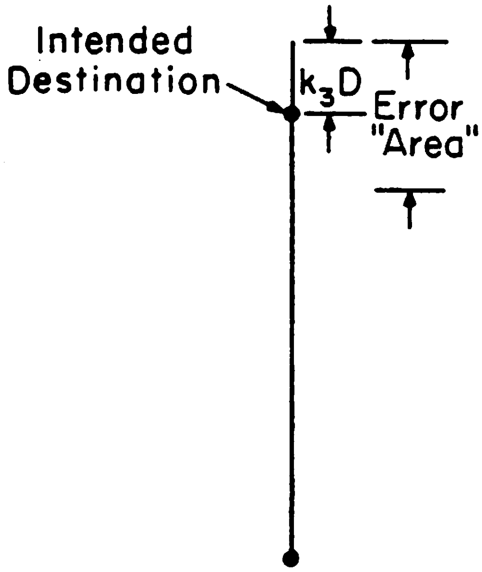

FIGURE F

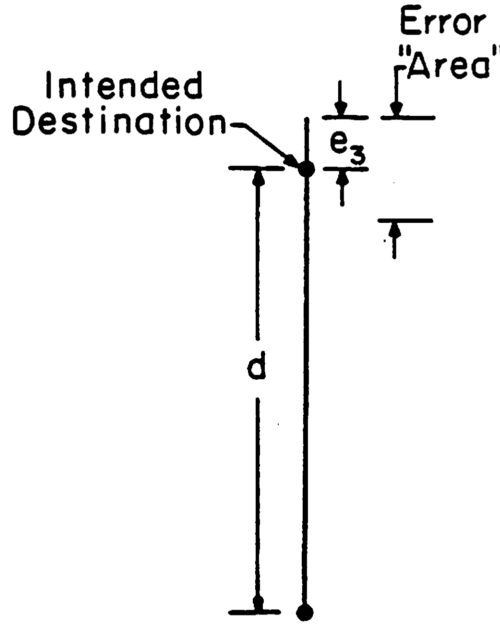

FIGURE E

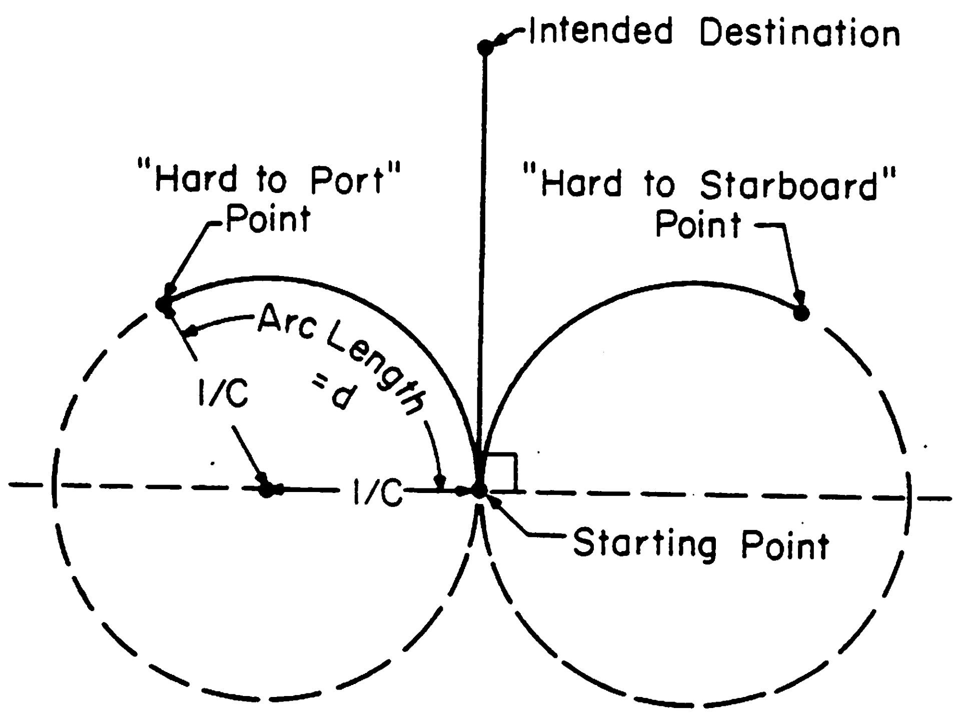

FIGURE G

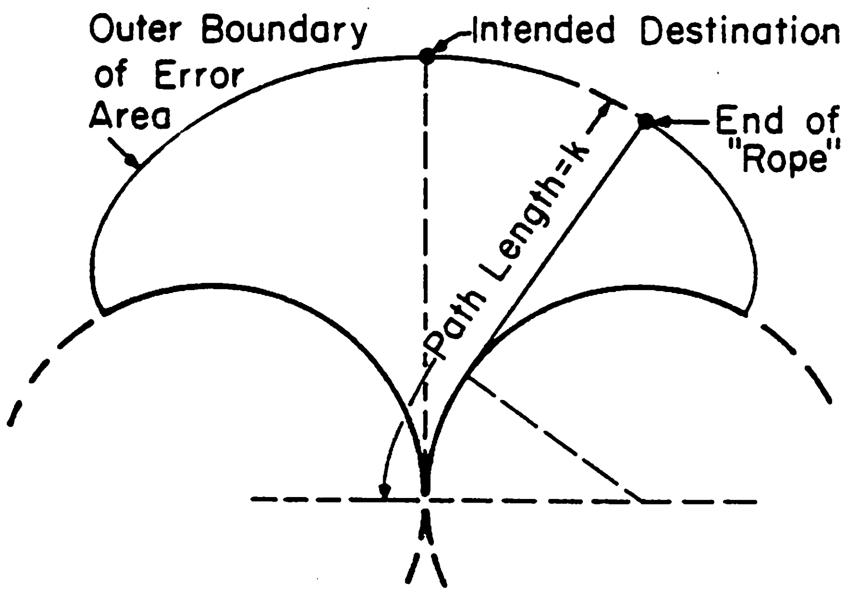

FIGURE H

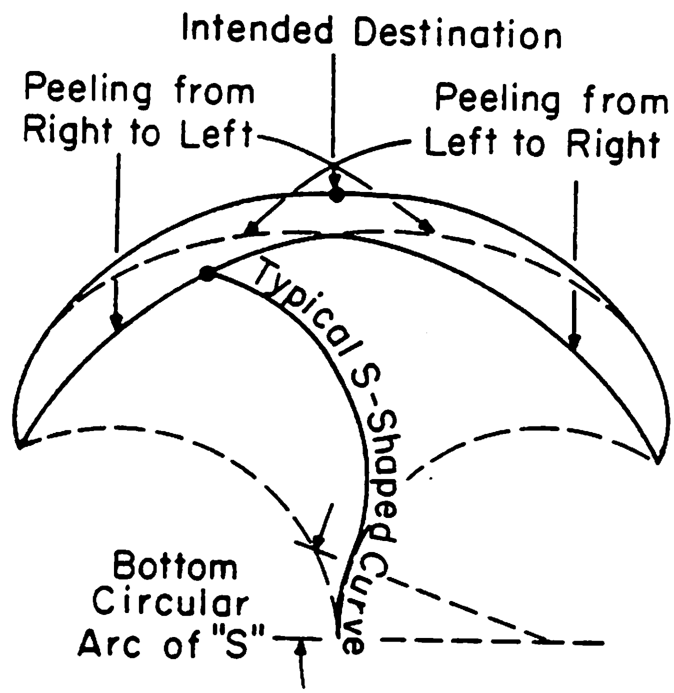

FIGURE I

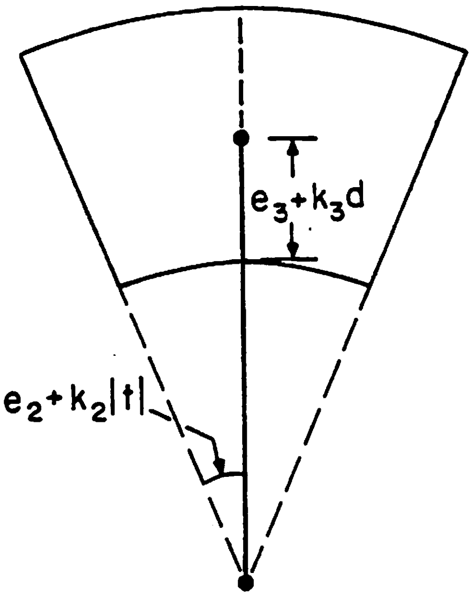

FIGURE J

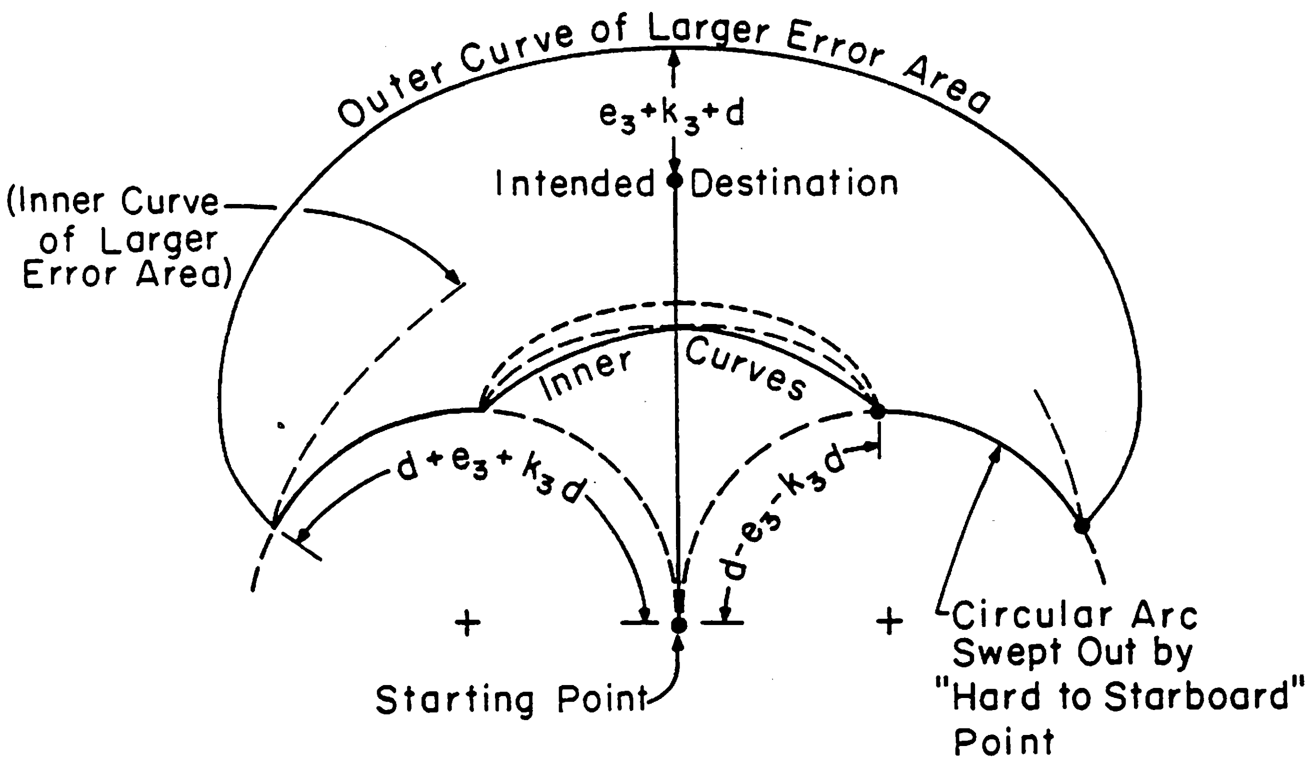

FIGURE K

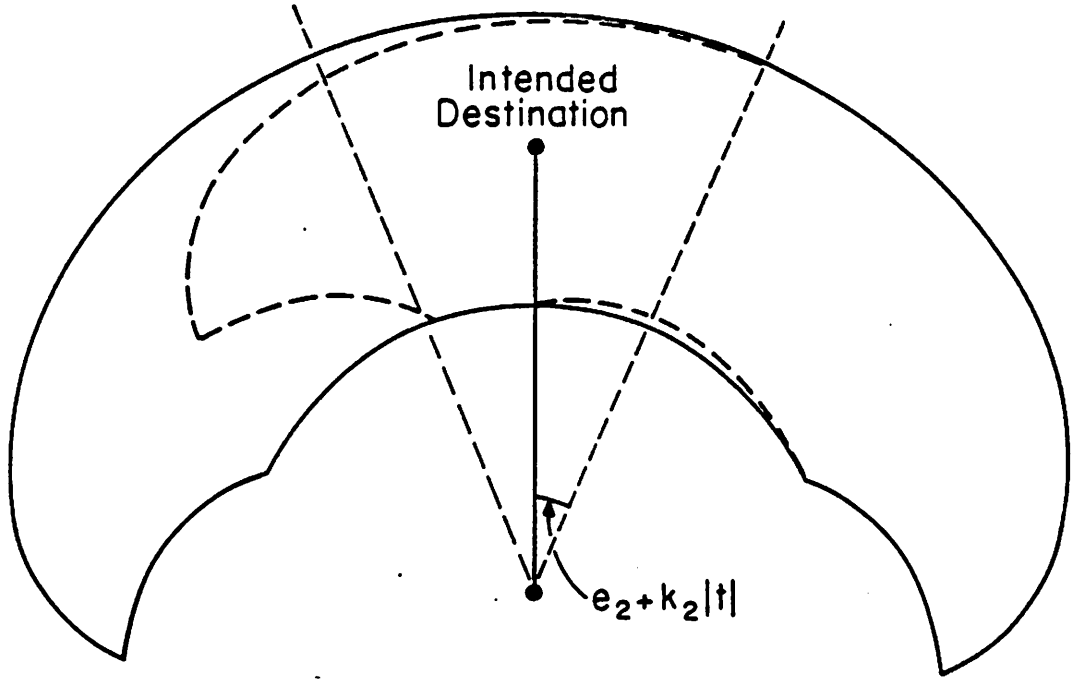

FIGURE L

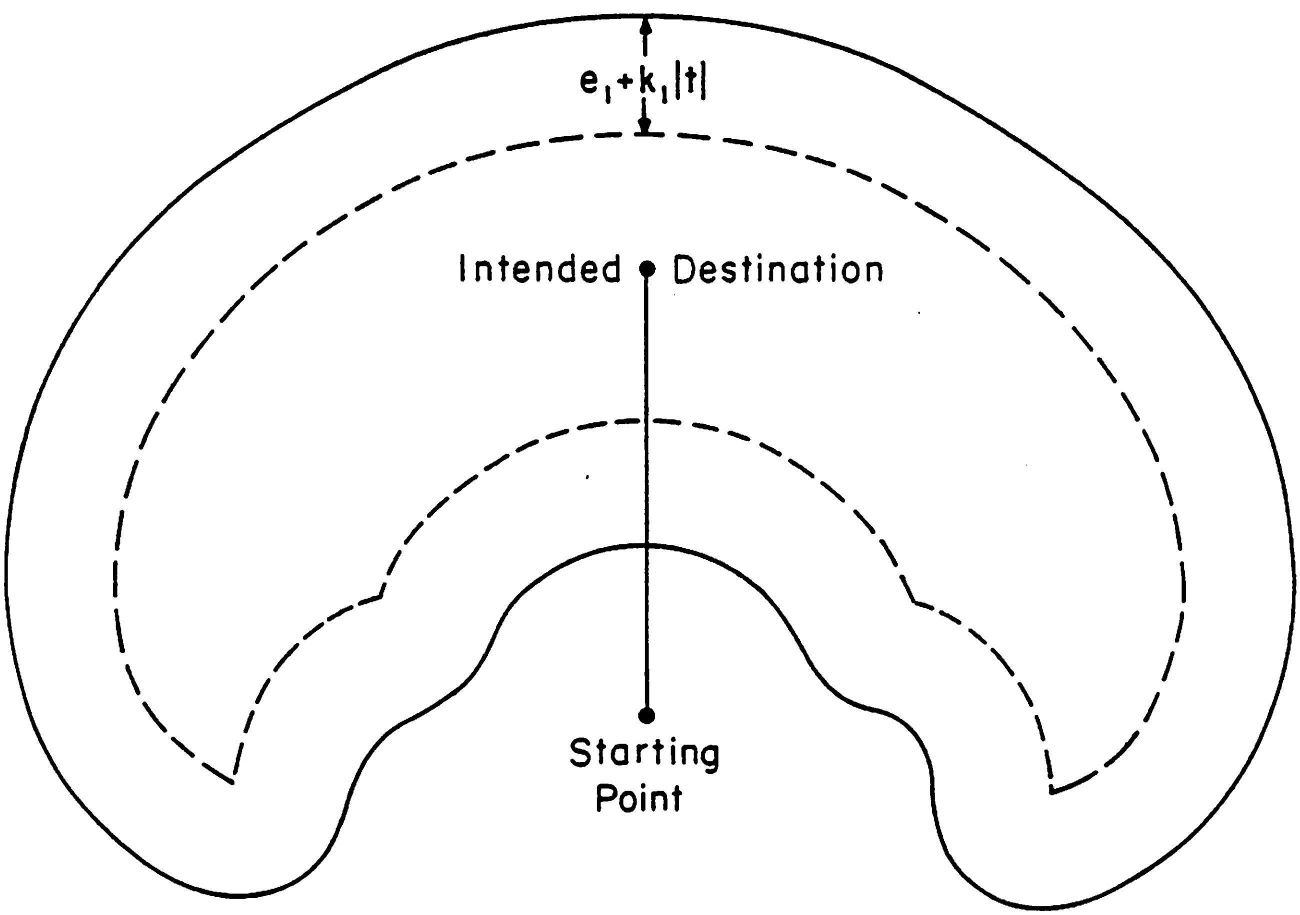

FIGURE M

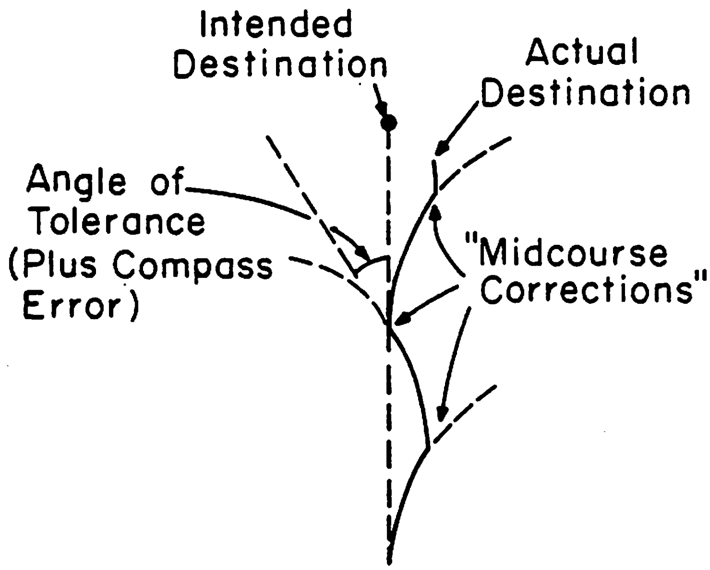

FIGURE N

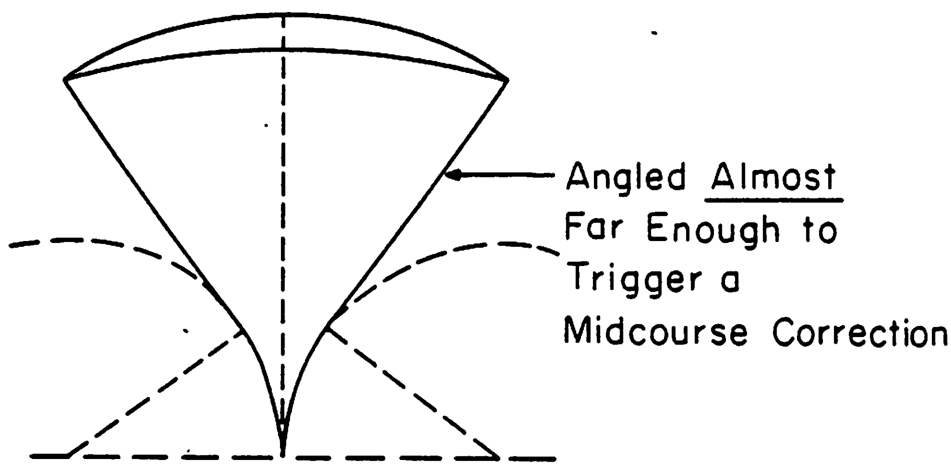

FIGURE O