The Robot Builder

Volume Eleven Number Nine September 1999

Notices

- Introductory Mobile Robotics Class - 10:00AM - 12:00PM
- Business Meeting - 12:30 - 1:00
- General Meeting - 1:00 - 3:00

Distribution

If you would like to receive The Robot Builder via e-mail, contact the editor at:

apendragn@earthlink.net

Inside this Issue

A Circular Navigation System 1

Symbolic Artificial Intelligence and First Order Logic 4

A Circular Navigation System [Part 1]

Jim Ubersetzig

This paper, originally published in December 93 and reprinted with permission, covers the theory required to understand the construction article in next month's newsletter. The construction article will describe the robot navigation system Jim Ubersetzig demonstrated at the April 99 RSSC meeting.

At last ! An accurate navigation system for use inside a room, suitable for use on an autonomous mobile vehicle. Simple, on-vehicle electronics and simple computations provide accurate knowledge of position and orientation within the room.

Equipment required is three wall mounted targets, a vehicle mounted spinning sensor, and a vehicle mounted microcontroller.

Targets

The wall mounted targets should be of the type that reflects light back to it's source. An example is the corner reflector.

One way to build a corner reflector is to cut out the bottom corner of a cardboard box. Notice that the three cardboard surfaces meet at right angles. That property is essential for a corner reflector. Glue mirrors to the inside of the three cardboard surfaces and your corner reflector is complete.

As an experiment tape the corner reflector to a wall and turn out the lights. A flashlight can be used to demonstrate corner reflector action. When the light from the flashlight strikes the corner reflector, most of the reflected light shines back at the flashlight.

Although corner reflectors made from a cardboard box can be used for the targets of the navigation system, there is a better way. Cardboard corner reflectors are flimsy and easily get out of alignment. The rear reflector from a bicycle has many corner reflectors molded in a piece of plastic. Buy one, saw it into pieces, and glue them to a wall.

Spinning Sensor

The vehicle mounted spinning sensor is illustrated in figure 1.

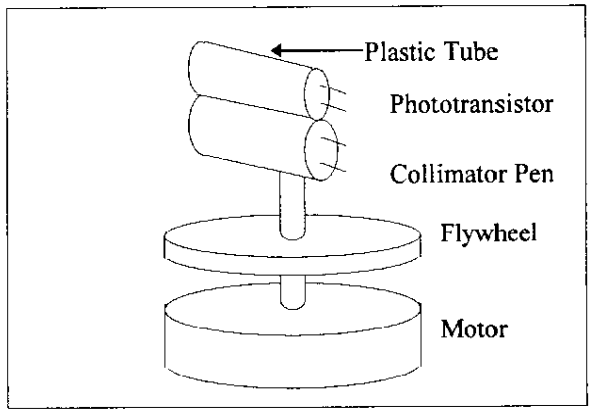

Figure 1 - Basic layout of sensor scanner

The motor and flywheel mount in the vehicle. The purpose of the flywheel is to even out the rotation and provide a smooth spin. Note that the optics rotate above the vehicle. The collimator pen emits a narrow beam of light that bounces off the wall target and returns to the phototransistor, which produces an electrical pulse. The plastic tube excludes light from other sources in the room which might distract the spinning sensor.

The collimator pen is a cylinder shaped part about 1/2 inch diameter and an inch long.

(see Spinning Sensor on page 2)

Spinning Sensor from Page 1

There are wires at one end and red light comes out the other end. Electrical power requirement is 2.5 volts @ 150 mA. Optical power output is 2.5 mW. Specification indicate the light beam should make a 2 inch spot at 100 yards.

Microcontroller

The vehicle-mounted microcontroller should have an internal timer register which counts using an accurate oscillator. This is to accurately measure time. The idea is to record the timer register when targets are detected.

The circuitry shown in figure 2 causes the microcontroller to execute an interrupt routine each time a wall target is detected. The interrupt routine is a very small computer program which stores the time in a memory buffer. When the memory buffer contains enough time values, the microcontroller calculates the vehicle position.

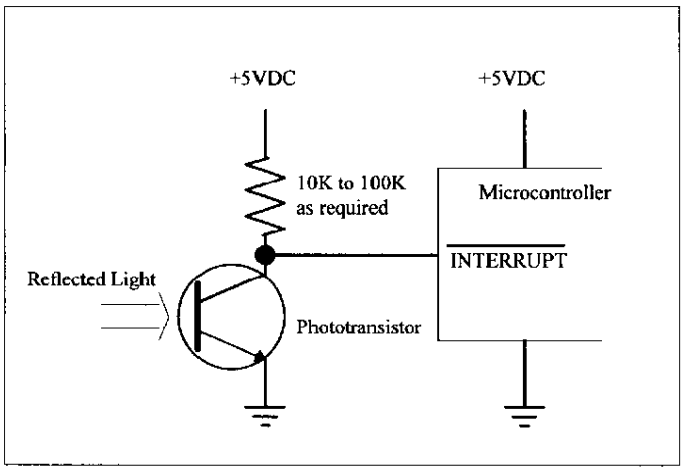

Figure 2 - Phototransistor circuit

Target Positions

A diagram of the room, in figure 3, shows the vehicle and the targets. When the targets are installed, get a tape measure and record target positions L01, L12, L23 and room dimensions L and W. These numbers must be known to the vehicle's microcontroller. So either put them in the source code for the software or provide a keyboard on the vehicle for typing in the values.

Angle Measurement

Looking at figure 4, the navigation software needs to know angles A12 and A23 but all it has is a list of

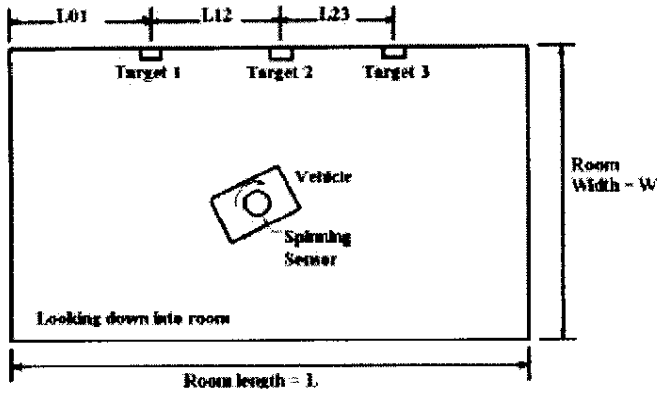

Figure 3 - Room layout

the times T[i] when targets are detected. So the first task is to determine which times are for target 1, which times are for target 2 and target 3.

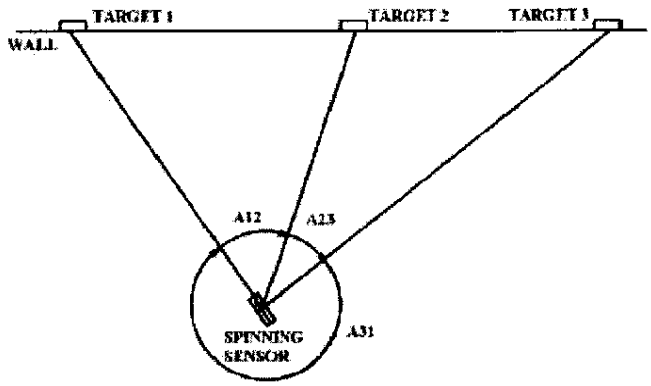

Figure 4 - Angle Measures

Graphically the timing is shown in figure 5.

Figure 5 - Timing diagram

Because the targets are on one wall only, most of the scan detects nothing, and the targets are grouped together. So it is easy to identify times for each target. The navigation software calculates the angles:

$$A12 = \frac{t2 - t1}{\text{next } t1 - \text{prior } t1} \times 360 \text{ degrees}$$

$$A23 = \frac{t3 - t2}{\text{next } t1 - \text{prior } t1} \times 360 \text{ degrees}$$

(see Spinning Sensor on page 3)

where t1 is the time when target 1 was detected, t2 is the time when target 2 was detected, etc.

Position Determination

The vehicle location can be determined from the angles if we remember a property of circles. " Given any two points P1, P2 on a circle, all the angles A are the same, and are 1/2 the angle measured from the center of the circle. " This is shown in figure 6.

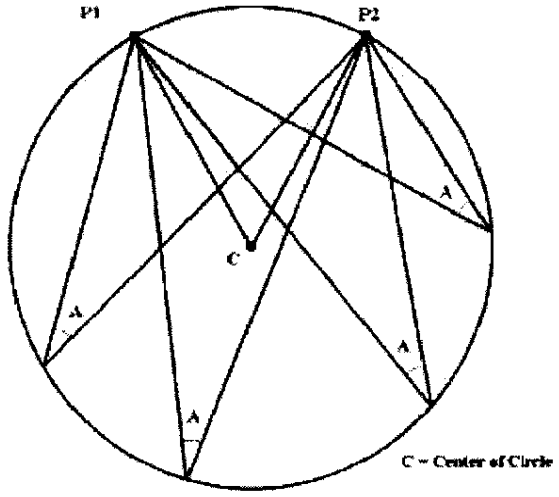

Figure 6 - Properties of the circle

This property of circles is applied by realizing that if P1 and P2 are wall target locations then A is the angle measured by the spinning sensor. Since the angle A is the same for all locations on the rim of a circle, it follows that the sensor position is somewhere on the rim of the circle.

There are really two circles involved because both A12 and A23 have been measured. The centers of the circles lie on center lines CL12 and CL23 which are perpendicular bisectors of lines L12 and L23.

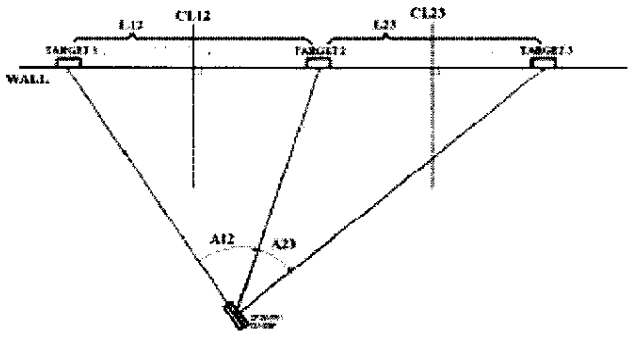

Figure 7 - Finding the angles

Since we know line L23 and the angle from target 2 to target 3 measured from the center of the circle = 2 x A23, we have enough information to draw the circle.

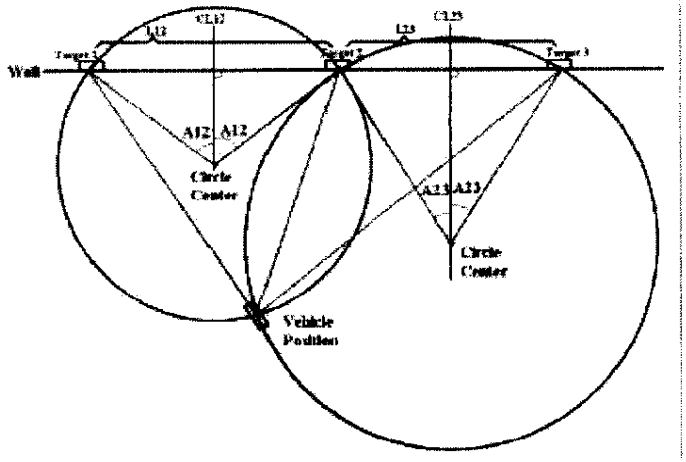

Figure 8 - Two circles delineating target position

Drawing both circles, we note that the vehicle position is where the circles cross. It is easy for a microcontroller to determine the vehicle distance from the wall, and the distance along the wall.

Option - Heading Determination

Adding the equipment shown in figure 9 will enable the vehicles microcontroller to determine the vehicles heading. If the beam interrupter is adjusted to detect when the sensor points forward on the vehicle, the microcontroller can measure the time between pulses from the interrupter module (in figure 9) and pulses from the target detecting phototransistor (in figure 1). The angle can be determined from the time difference. After the position of the vehicle has been calculated, the bearing angle to a target can be determined. Adding the angle from the paragraph above gives the vehicle heading.

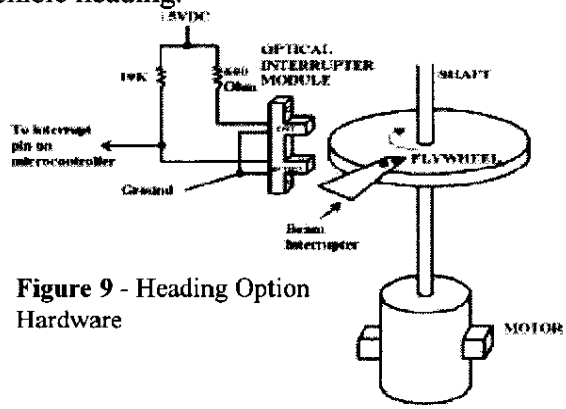

Figure 9 - Heading Option Hardware

Symbolic Artificial Intelligence and First Order Logic

By Arthur Ed LeBouthillier

There are many different approaches to creating artificial intelligence in computers and, for our purposes, intelligent robots. Since the founding of the field of Artificial Intelligence (AI), symbolic approaches have persisted. Symbolic approaches represent problems as logical propositions between objects and use rules of logic to perform deductions. It is believed that by representing these relationships and reasoning about them, we can make computers perform similar mental feats as humans. This differs from non-symbolic techniques such as neural nets where often no consideration is given to how a problem is represented or solved as long as it is solved. Other trends away from symbolic AI approaches are some behavioral methods where there is no attempt to model the world internally. This article covers some of the basic ideas underlying symbolic AI; understanding these ideas is required to understand how more sophisticated AI programs work and eventually implementing them AI techniques in robots.

First Order Logic

Classical logic has a long history and is well understood. With this historical basis, early AI researchers created representations of logic that would allow computers to perform logical reasoning. First Order Logic was one of these developments.

First Order Logic provides a method to store declarations about the world, the robot and everything it knows. There are limits to what it can represent, but you can go a long way before running into them. The limits it has are similar to the limits that exist on any programming language. Any given language can do what any other language can do, but sometimes it is harder to do some tasks in a given language. Rather than detail the theory in a mathematical way, let's look at a simple problem using First Order Logic (FOL). Before that, let's cover the basics.

First Order Logic Basics

The most basic elements of first order logic are a bunch of symbols. These symbols represent objects and relationships in the world. Examples of symbols are: Bob, Sam, Person, Hungry, Socrates, is-a, and a-kind-of. I think you're already beginning to see the significance of these symbols. They are like words in a sentence.

The second element of first order logic is the way that symbols can be brought together to form propositions. A proposition is a kind of sentence in the language of First Order Logic. First Order Logic describes a way to combine words (symbols) into sentences to make meaningful propositions. It specifies that a meaningful proposition has the structure of:

Proposition = Predicate(Arg_1, Arg_2, ..., Arg_n)

The predicate describes the relationship between each of its arguments. The first argument in a proposition is, by convention, the subject of the sentence and the second argument is usually the object of the sentence (if it exists).

Therefore, we could translate various sentences in the English language into propositions in First Order Logic:

English	FOL
Bob is hungry	hungry(Bob)
Socrates is a man	is-a(Socrates, Man)
Man is mortal	mortal(Man)

The representational power of First Order Logic is very great and allows you to translate virtually any idea you can express in a sentence as a proposition. There are some problems with the ability to represent time-based changes, but there are often tricks one can perform to alleviate them.

Variables and Universal Qualifiers

We can also put variables into propositions,

(see FOL on page 5)

allowing a single proposition to stand for a whole class of propositions. A variable is merely a symbol which is designated as being a variable. For our purposes, a variable will be represented as a symbol preceded by a question mark (i.e. ?X).

A Universal Qualifier is a compact notation for asserting new facts from old ones. It represents a special technique one may apply to a list of propositions to develop new facts. It is usually used in conjunction with variable-based propositions so that whole classes of new statements can be made. In classical logic, a Universal Qualifier has the structure of:

For All X, Proposition1 -> Proposition2

This is read as "For All X, Proposition one implies Proposition two." Therefore, we could say that all *Mammals are Vertebrates* in a compact rule by asserting:

For All X, mammal(?X) -> Vertebrate(?X)

Using our First Order Logic representation and substituting "universally" for the "For All" part, we could make the same statement:

Universally(mammal(?X), Vertebrate(?X))

Logical Combiners

We can also use And and Or modifiers to make more sophisticated statements. For example, *a bird is a mammal which uniquely has feathers.* This fact could be established by this rule:

Universally(mammal(?X) AND has-feathers(?X), bird(?X))

Deduction

Using the simple techniques outlined above, we can perform a function similar to deductive reasoning. A common example of deductive reasoning is: Knowing that *Socrates is a Man* and *All men are mortal*, we can conclude that *Socrates is Mortal.* By applying the universal rule listed below, we could deduce this in First Order Logic:

Start:
is-a(Socrates,Man)
Universally (is-a(?X, Man), Mortal(?X))

Result:
is-a(Socrates, Man)
Mortal(Socrates)

Deduction constitutes the major tool used in Expert Systems. By building up a list of propositions (known as the Knowledge Base) with a list of rules (known as the rule base), expert systems are able to deduce new facts from what they already know. By including extensions for input and output, they allow the Expert System to interact with the world. Although expert systems are limited, they have proven themselves to be extremely useful in certain applications.

A Detailed Example

A more complete example of how to represent a complex description can be seen in the Cyc knowledge base. The Cyc project is a multi-year attempt to encapsulate common-sense knowledge using a First Order Logic-like language, called CycL. It represents the most advanced attempt to make a knowledge base for artificial intelligence.

In figure 1 is an example of relations from the concept "person." The Cyc knowledge base has two major distinction related to sets. A thing that represents a subset of a set "generalizes" it and its relation predicate is **genls**. An element of a set uses the **isa** predicate.

The distinction between a set, a subset and an element of a set is an important thing to distinguish when reasoning about the world. If Fred is a person and a person is a collection (i.e. a collection of all people), Fred is not a collection. Fred, the element of the set Person is not equivalent to a set. Fred is an element of a set, not a set. There are not multiple instances of Fred walking around. If there were, they would be Fred1 and Fred2, elements of set Freds which would generalize person. Using **genls** and **isa** allows us to make these distinction.

We can see that Cyc (and thus First Order Logic) is able to represent many varied distinctions and traits that we understand about people (i.e. a person generalizes primate ... generalizes mammal ... generalizes vertebrate ... generalizes animal). We can also understand that a person is an agent and that a person exists with some temporal properties (i.e. exists for a duration of time).

By applying various rules like deduction, we are able to resolve new facts that don't explicitly exist in the database. Cyc, using the CycL language, provides a whole suite of rules and functions which allow the basic propositions to resolve a much wider breadth of knowledge.

Knowledge and Robots

A robot using a complex knowledge base like Cyc or First Order Logic would be able to reason about many different aspects of the world. It could reason from first principles, reason about its goals, reason about interactions between its actions in the world and plan appropriately. Examples of working robots doing these things do exist in research labs. The Flakey robot of Stanford Research Institute has demonstrated some advanced reasoning capabilities. The NASA's Remote Agent on the Deep Space One spacecraft is another.

Of course, nothing we have discussed here addresses the issue of learning about the world or how that information is brought into the robot, but the mechanism to reason with knowledge is well understood. Learning is an ongoing part of AI research and future robots should be able to convert sensory information into symbolic representations of the world that they would then be able to reason with.

Summary

First Order Logic is an important tool for symbolic AI. It provides a mechanism to represent and manipulate knowledge about the world in a computer usable form. Cyc is an example of a complex database using an extended version of First Order Logic.

```
;;; Agent
isa(Agent ExistingObjectType)
genls(Agent CompositeTangibleAndIntangibleObject)

;;; Animal
isa(Animal BiologicalKingdom)
genls(Animal PerceptualAgent)
genls(Animal AnimalBLO)
genls(Animal SolidTangibleThing)
genls(Animal Organism-Whole)

;;; BiologicalKingdom
isa(BiologicalKingdom BiologicalTaxonType)
genls(BiologicalKingdom BiologicalTaxon)

;;; BiologicalLivingObject
isa(BiologicalLivingObject ExistingObjectType)
genls(BiologicalLivingObject OrganicStuff)
genls(BiologicalLivingObject CompositeTangibleAndIntangibleObject)

;;; BiologicalOrder
isa(BiologicalOrder BiologicalTaxonType)
genls(BiologicalOrder BiologicalTaxon)

;;; CompositeTangibleAndIntangibleObject
isa(CompositeTangibleAndIntangibleObject ExistingObjectType)
genls(CompositeTangibleAndIntangibleObject PartiallyTangible)
genls(CompositeTangibleAndIntangibleObject PartiallyIntangible)
genls(CompositeTangibleAndIntangibleObject SomethingExisting)

;;; ExistingObjectType
isa(ExistingObjectType Collection)
genls(ExistingObjectType TemporalStuffType)
genls(ExistingObjectType ObjectType)

;;; IndividualAgent
isa(IndividualAgent ExistingObjectType)
genls(IndividualAgent Agent)

;;; Mammal
isa(Mammal BiologicalClass)
genls(Mammal Vertebrate)

;;; Omnivore
isa(Omnivore ExistingObjectType)
genls(Omnivore Organism-Whole)

;;; OrganismClassificationType
isa(OrganismClassificationType SiblingDisjointCollection)
genls(OrganismClassificationType ConventionalClassificationType)
genls(OrganismClassificationType ExistingObjectType)

;;; Organism-Whole
isa(Organism-Whole ExistingObjectType)
genls(Organism-Whole BiologicalLivingObject)

;;; PerceptualAgent
isa(PerceptualAgent ExistingObjectType)
genls(PerceptualAgent IndividualAgent)

;;; Person
isa(Person OrganismClassificationType)
genls(Person LegalAgent)
genls(Person Primate)
genls(Person HumanOccupationConstructResident)
genls(Person Omnivore)

;;; Primate
isa(Primate BiologicalOrder)
genls(Primate Mammal)
genls(Primate TerrestrialOrganism)

;;; Vertebrate
isa(Vertebrate BiologicalTaxon)
genls(Vertebrate Animal)
```

Figure 1 - An example of First Order Logic from Cyc

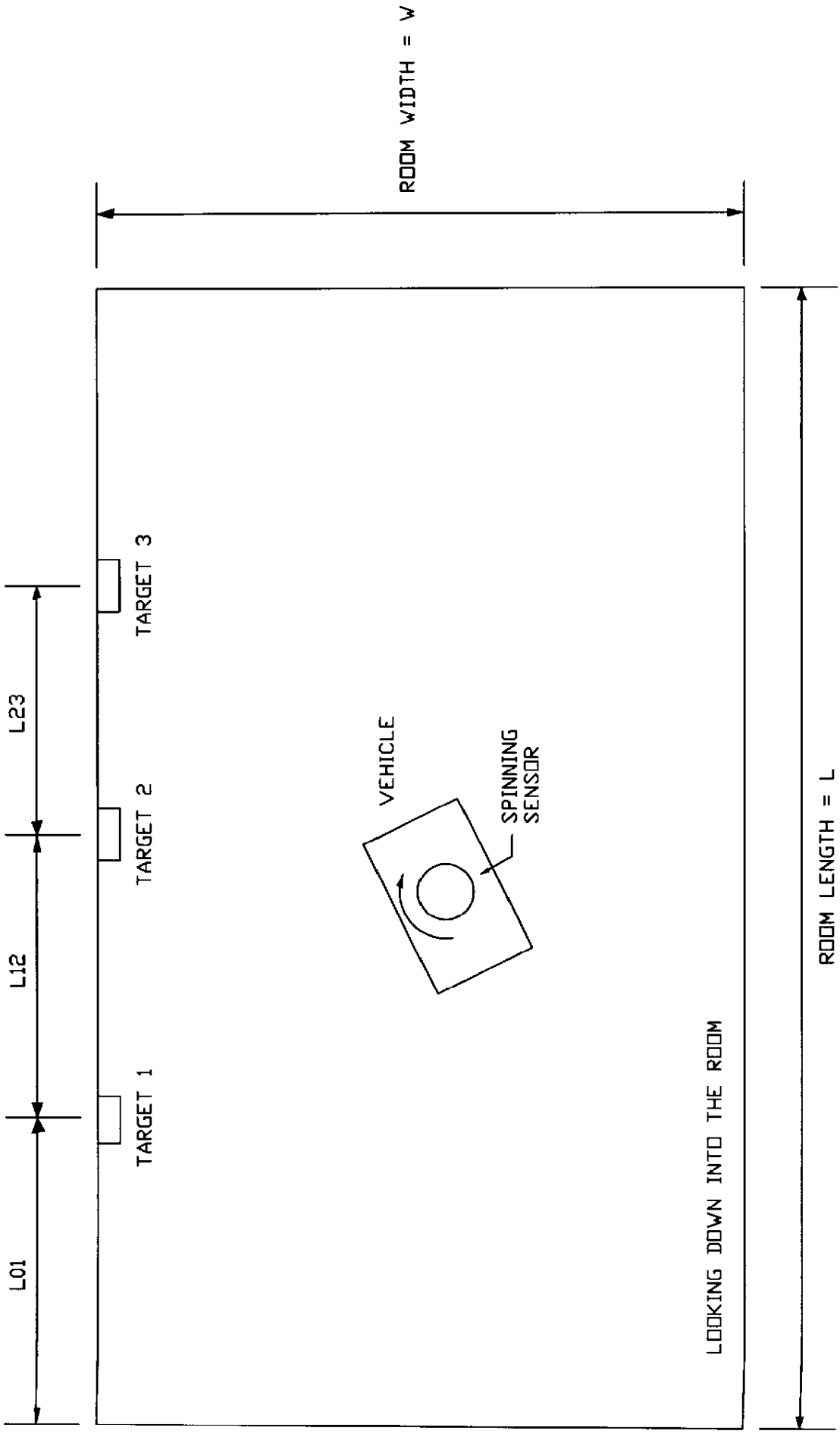
ROOM WIDTH = W
TARGET 3
TARGET 2
TARGET 1
L23
L12
L01
VEHICLE
SPINNING
SENSOR
LOOKING DOWN INTO THE ROOM
ROOM LENGTH = L

FIGURE 3

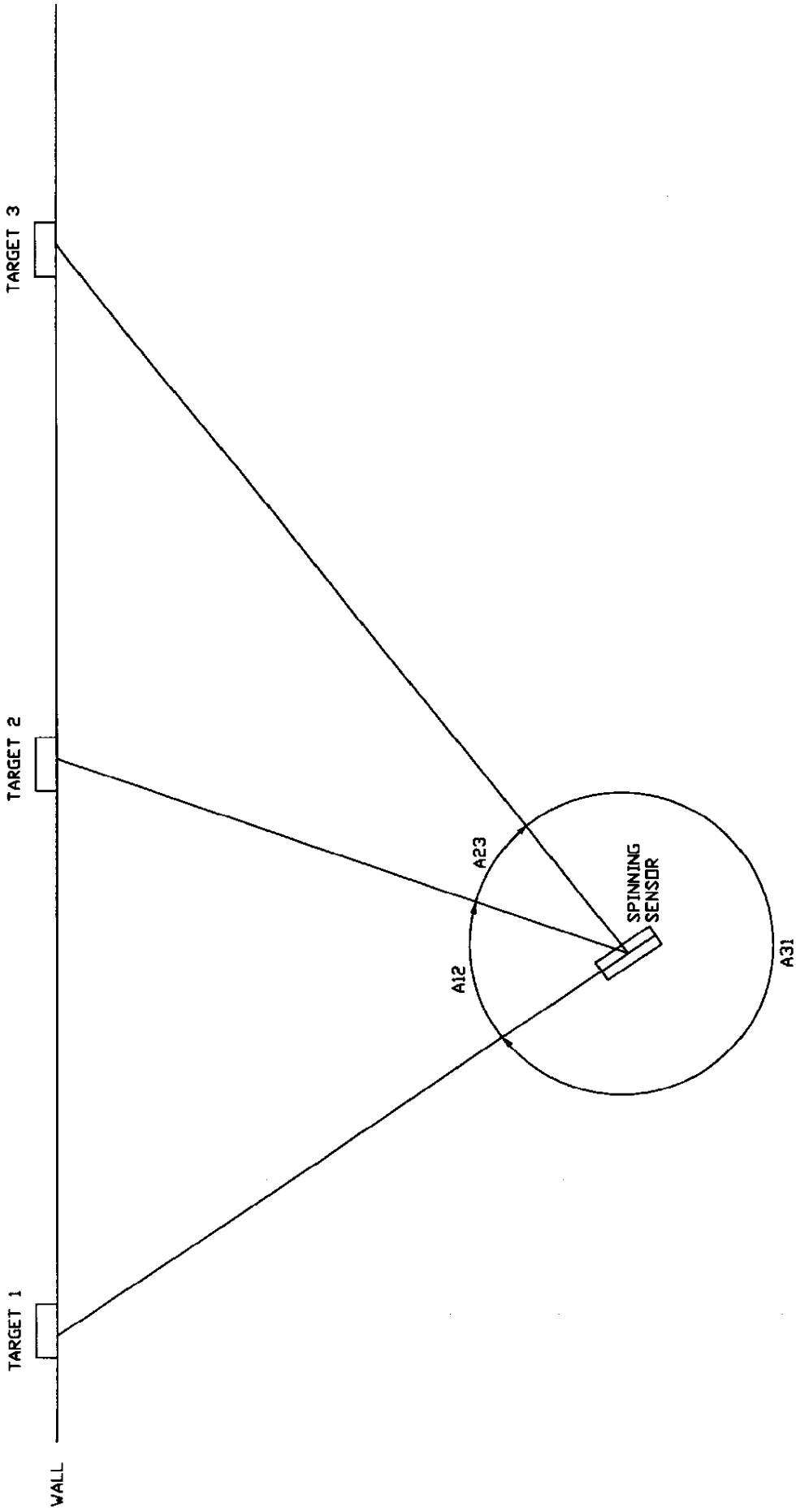

FIGURE 4

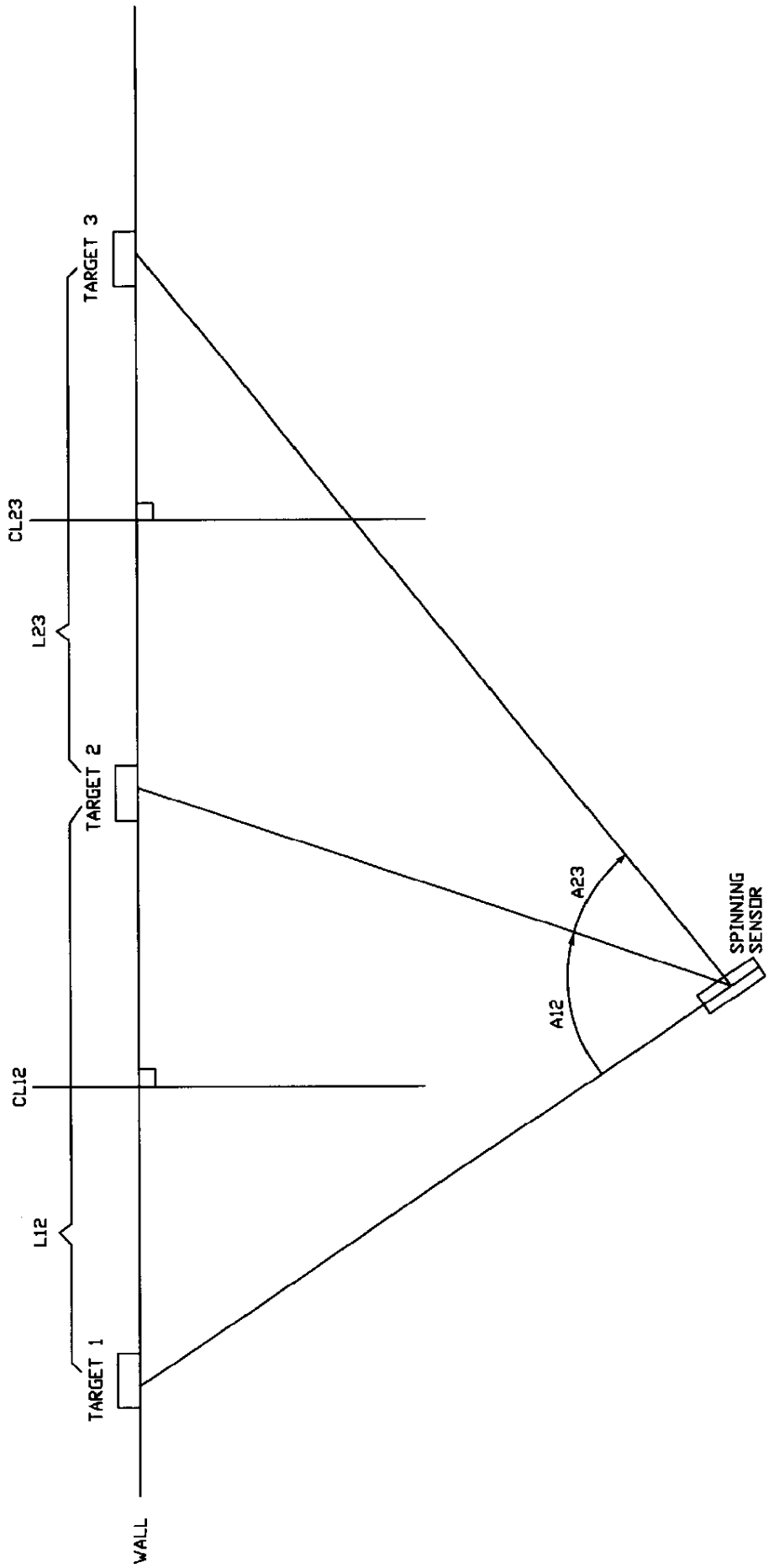
WALL
TARGET 1
TARGET 2
TARGET 3
L12
L23
CL12
CL23
A12
A23
SPINNING SENSOR

FIGURE 7

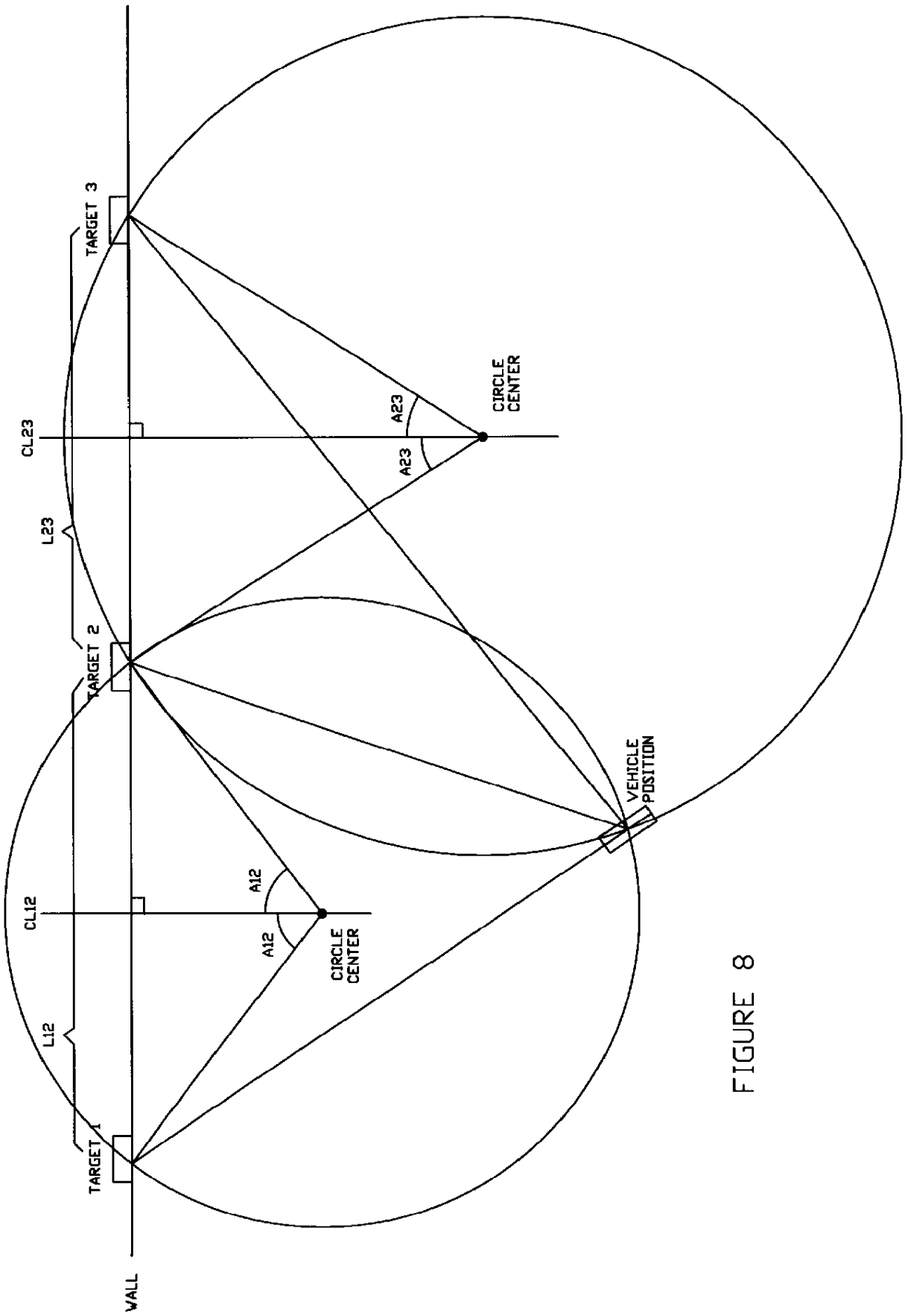
WALL
TARGET 1
TARGET 2
TARGET 3
L12
L23
CL12
CL23
A12
A12
A23
A23
CIRCLE CENTER
CIRCLE CENTER
VEHICLE POSITION

FIGURE 8