# Cone Detection Algorithm

## Background

### Bayes Theorem

* Pr (location | data) = Pr (data | location) \* Pr (location) / Pr (data)
* Pr (data) can be treated as a normalizing constant

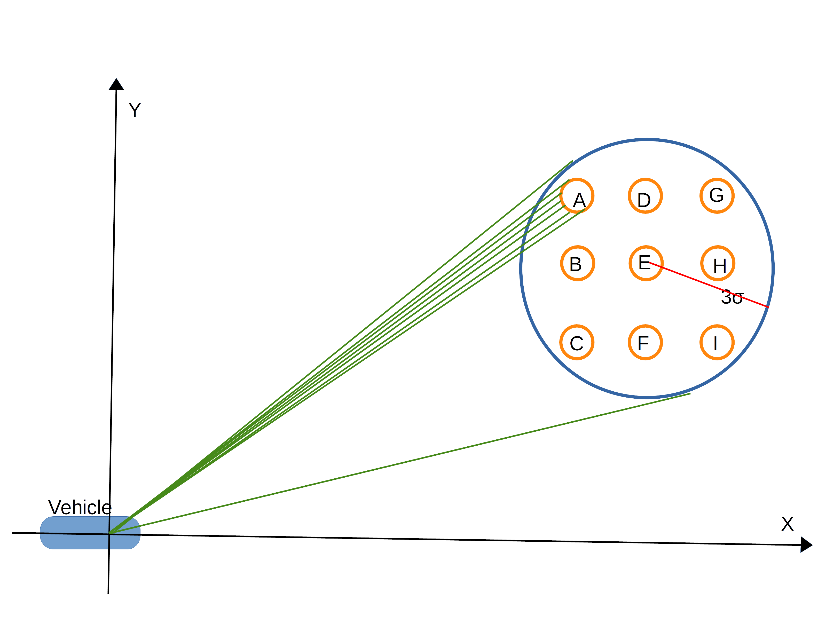
### World coordinate system

* The location of the cone is known with high accuracy
* The location of the vehicle has uncertainty which we characterize by the standard deviation, σ.
* Vehicle is within a circle of radius 3 σ

### Vehicle coordinate system

* By definition, the vehicle is at the origin
* Cone is within a circle of radius 3 σ
* Decide on the resolution to which we want to detect cone: say from 1.0 to 0.05 m
* Form a grid in the cone circle of this size

### Geometry



* Pr (location) can be found from the normal distribution. For example:
* **Location;** **σ from center** **Likelihood** **Pr (location)**
* A 2.5 0.03 0.027
* B 1.5 0.15 0.134
* C 2.5 0.03 0.027
* D 1.5 0.15 0.134
* E 0 0.4 0.357
* F 1.5 0.15 0.134
* G 2.5 0.03 0.027
* H 1.5 0.15 0.134
* I 2.5 0.03 0.027
* Total 1.12 1
* For each possible cone location, treat its location as a given and compute the ideal data for   
  Pr (data| location)
* Use inaccuracies of instrument to compute possible noisy data
* Find Pr(location|data)

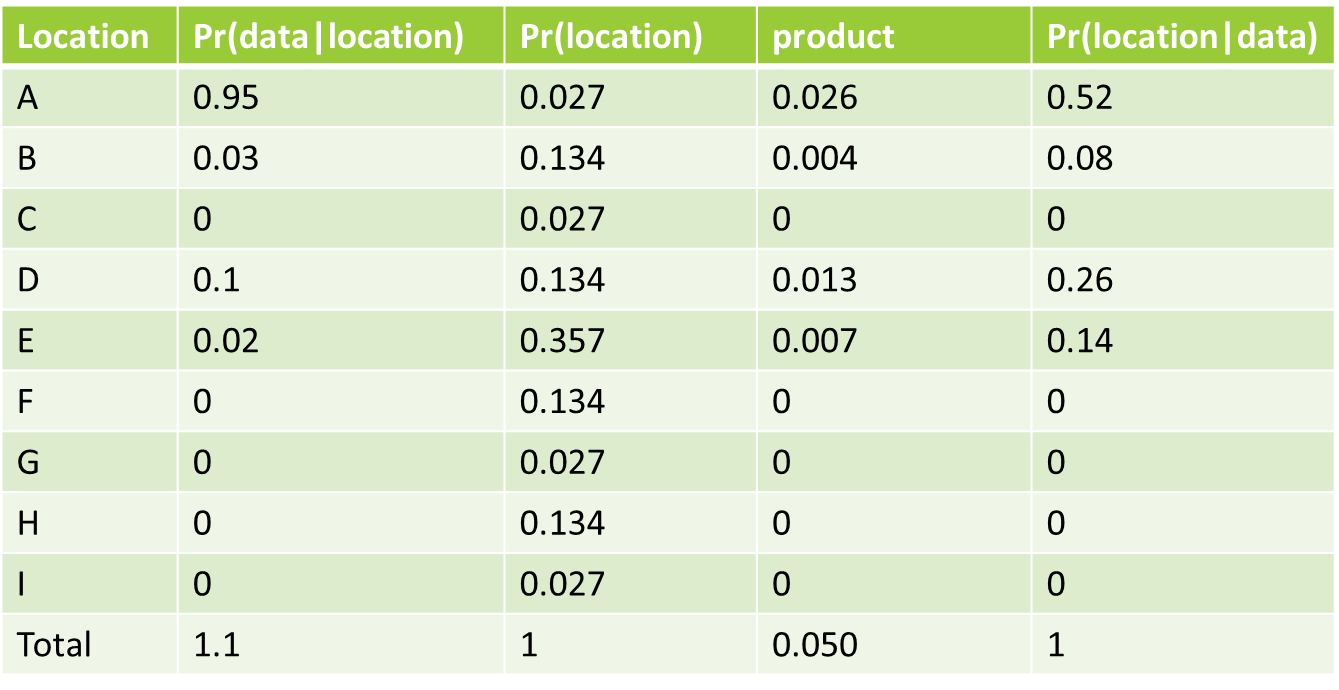
### Ideal data | A

* For example, consider the case where it is 14.5 m to the center of the circle of possible cones.
* Assume σ for vehicle location is 1 m, making a diameter of 6 m for the circle of possible cones.
* Assume that there are 16 data rays that intersect the circle and that the center of cone A is at 13.9 m. The data that we expect with no error is
* [-,13.5,13.2,13.5,-,-,-,-,-,-,-,-,-,-,-,-]

### Pr(data | A)

* Suppose that the data that we have received is
* [30, 13.1, 13.0, 13.4, 20, 25, 25, 28, 30, 30, 10, 26, 24, 24, 30, 30]
* Given that the cone is at location A, this is likely data.
* We might assign Pr(data | A) = 0.95.
* If we hypothesize that the cone is at location F, Pr(data|F)=0.
* We might assign Pr(data|D)=0.1, Pr(data|B)=0.03, Pr(data|E)=0.02
* These probabilities are based on analyzing the lidar uncertainty.

### Pr(location | data)

* Pr(A|data) = Pr(data|A)\*Pr(A)/Pr(data) = 0.95 \* 0.27 / Pr(data)
* 

### New vehicle location

* In the vehicle coordinate system, we would weight cone locations A, B, D and E according to their probabilities. This becomes the center of the new cone location.
* The new circle diameter will shrink, perhaps producing a new σ = 0.5 m. In the world coordinate system, the vehicle position shifts by the amount that the center of the cone locations has shifted.
* The uncertainty in vehicle location is reduced to the new σ

# Assumptions:

All cones are the same size.

No error in cone position

Uncertainty in vehicle location is circularly symmetric.

Laser range data scanner is functional outdoors and not overwhelmed by ambient light

Laser data is an array of ranges to nearest obstacle, ordered by angular bearing to object, where the bearing is monotonic. There is no interaction between bearing and range. (Some data seem to contradict this assumption).

Laser data lies in a plane whose intersection with a cone is a circle (not an ellipse). Only the near half of the circle is visible.

There is no shadowing of the returned signal, or if there is it will be indicated by low signal intensity.

Speed of vehicle has a negligible impact on the range data.

No effect from rain, snow, dust, fog, daylight or artificial lights.

No interference from other lasers.

# Algorithm

**Inputs:** Estimated vehicle GPS location (GPS\_E, GPS\_N) in degrees; standard deviation of location (σ) in meters; list of exact landmark (cone) positions (LM\_GPS\_E[], LM\_GPS\_N[] in degrees; diameter of a cone (ConeDiam) in meters at the height of the laser scanner; desired spacings between possible cone locations (ConeSpacing) in meters, radius of earth (EARTH\_RADIUS) in meters, range of Lidar (LidarRange) in meters; data array from laser scan.

**Outputs**: New estimate of vehicle location (NewGPS\_E, NewGPS\_N); new estimate of standard deviation (Newσ); probability that a cone was found (PrDetect).

NewGPS\_E = GPS\_E;

NewGPS\_N = GPS\_N;

Newσ = σ;

PrDetect = 0;

// Vehicle is at (0,0) in vehicle coordinate system

For each cone // vehicle coordinates  
 locX[cone] = EARTH\_RADIUS \* (LM\_GPS\_E[cone] – GPS\_E) \* π / 180 \* cos(GPS\_N \*π /180)  
 locY[cone] = EARTH\_RADIUS \* (LM\_GPS\_N[cone]– GPS\_N) \* π / 180  
 Dist\_Squared[cone] = locX[cone]\* locX[cone] + locY[cone]\* locY[cone]

Select the cone for which Dist\_Squared is minimum. From now on, this will be referred to as (LocX, LocY).

If minimum Dist\_Squared > LidarRange\* LidarRange, return.

Construct an array of cone locations centered on (LocX, LocY) where grid spacing is ConeSpacing and maximum distance of any location from (LocX, LocY) is < 3\* σ.

For each location in the grid  
 Compute likelihood of location based on ConeDist - the distance from (LocX, LocY), assuming a normal distribution. This is Pr(location)  
 Compute visual angle of cone as: A = 2\*tan( (ConeDiam/2) / ConeDist)\*180/pi  
 Compute number of points on cone as: N = Integer(#Points per degree from Lidar \* A)  
 Construct an array of ideal data as: P = ConeDist - sin(i\*180/(N-1))\*(ConeDiam/2) (if N is odd: i = 0,1,...,(N-1)/2,…, 1,0; if N is even: i = 0,1,…,(N-2)/2,(N-2)/2,…,1,0.)  
 Look at N points of actual data centered on the index for the location midpoint.  
 Evaluate how well actual data matches ideal data and assign Pr(data | location)

Take center of ideal cone. For each point in the ideal data, compute the distance from the ideal center to the actual data point: act\_data. Form deviation = act\_data - ideal data. Take sum of all deviations. If that sum is zero, we have a perfect fit! Otherwise, take average of deviation for this cone position. This is a likelihood measure for this cone.

Assign Pr(data | location) based on these likelihood measures.

Pr (location | data) = Pr (data | location) \* Pr (location)

Find the location for which Pr (location | data) is highest; if too low, return

Assign PrDetect based on highest Pr (location | data).

Set (NewConeX, NewConeY) to this location.

Find the minimum Radius for which a circle centered on (NewConeX, NewConeY) contains no location for which Pr (location | data) is substantially more than zero.

Newσ = Radius / 3

VehicleX = (NewConeX- LocX)

VehicleY = (NewConeY- LocY)

NewGPS\_N = GPS\_N + VehicleY \* 180 / (π \* EARTH\_RADIUS)

NewGPS\_E = GPS\_E + VehicleX \* 180 / (π \* EARTH\_RADIUS \* cos(GPS\_N \*π /180) )

return