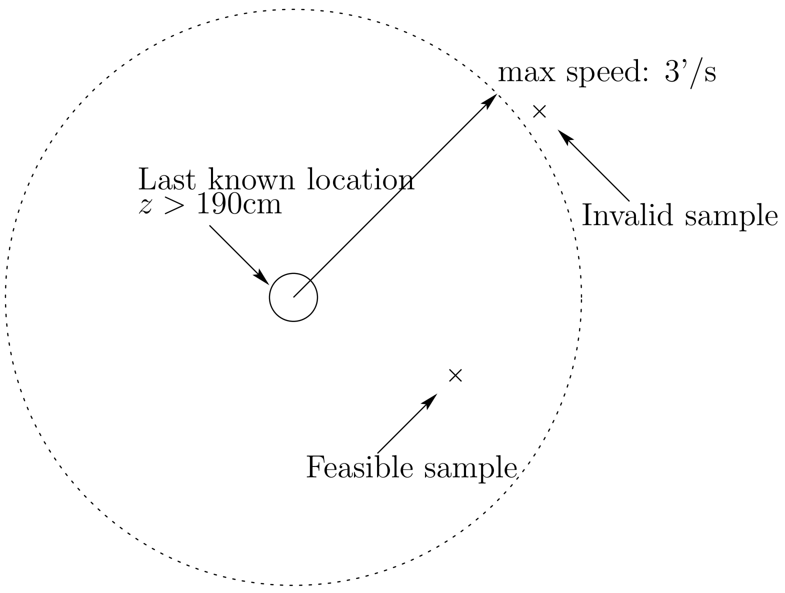

Figure 1: Valid robot positions on the floor

1 Filtering out erroneous readings

Discard $|d - h_{\text{ceiling}}| > \epsilon$

By sitting still long enough, the robot can get a good approximation to its location. To do this, perform the least-squares calculation until you have two positions with a valid height (currently, $z > 190$). If they are close, then your location is known. Now that the listener's location is known, we may discard clearly erroneous readings.

Since the robot has a maximum speed, we may safely discard any beacon readings which would require the robot to travel at a rate exceeding this fixed speed. The resulting circle is illustrated in Figure 1. Given the robot's heading, the valid region can be further restricted to an ellipse inside this circle, but we do not consider this for now. Either way, this bounding region can be used to obtain bounds on valid beacon readings.

Figure 2 shows how radial distances on the floor translate into valid beacon distances. The beacon distance from the last known position, h, satisfies the relation $h^2 = (r_b - r_0)^2 + \bar{h}^2$. The extreme values satisfy $h^2_{\text{min}} = (r_b - r_{\text{min}})^2 + \bar{h}^2$ and $h^2_{\text{max}} = (r_b - r_{\text{max}})^2 + \bar{h}^2$. In general, $h_{\text{max}} - h \geq h - h_{\text{min}}$; however, the difference is not great, and so we take $\Delta h = h_{\text{max}} - h$ and use

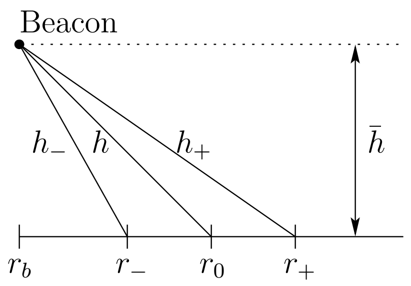

Figure 2: Valid beacon readings

the bound $h - \Delta h < h_{\text{valid}} < h + \Delta h$.

Ceiling height: $\bar{h} = B_z - r_z$.

Calculating Δh: Multiply the additive conjugate pair to obtain $(h_+ + h)(h_+ - h) = h_+^2 - h^2$. By the definitions of h, we also have that $2h \leq h_+ + h \leq 2h_+$. Then use this inequality to obtain an upper bound on Δh.

$$\Delta h \leq h_+ - h = \frac{h_+^2 - h^2}{h_+ + h} \leq \frac{h_+^2 - h^2}{2h}$$

$$\begin{aligned}
\Delta h &\leq \frac{[(r_+ - r_b)^2 + \bar{h}^2] - [(r_0 - r_b)^2 + \bar{h}^2]}{2h} \\
&= \frac{(r_+ - r_b)^2 - (r_0 - r_b)^2}{2h} \\
&= \frac{(r_+^2 - 2r_+r_b + r_b^2) - (r_0^2 - 2r_0r_b + r_b^2)}{2h} \\
&= \frac{r_+^2 - r_0^2 + 2(r_0 - r_+)r_b}{2h} \\
&= \frac{(r_0 + dr)^2 - r_0^2 - 2r_bdr}{2h} \\
&= \frac{(r_0^2 + 2r_0dr + dr^2) - r_0^2 - 2r_bdr}{2h} \\
&= \frac{dr^2 + 2(r_0 - r_b)dr}{2h}
\end{aligned}$$

Optionally, add in acknowledgement of normal measurement variance to get the bound $\Delta h \leq [dr^2 + 2(r_0 - r_b)dr]/2h + 2\sigma_r$. Note that the radial difference is $dr = k_v dt \approx (3'/s)dt = (91cm/s)dt = (0.09cm/ms)dt$.

2 Moving System Model

The receiver has a current position and recent average velocity. These will agree with observed measurements.

Beacon positions: $B^i = [B^i_x, B^i_y, B^i_z]$; i indicates which beacon was heard
Current time: t
Present position: $\vec{p} = [\vec{p}_x, \vec{p}_y, \vec{p}_z]$
Recent velocity: $\vec{v} = [\vec{v}_x, \vec{v}_y, \vec{v}_z]$

Note that $\vec{v}_z \equiv 0$ since the robots stay at a fixed height from the floor plane.

At a time t_1 in the recent past, we have that $p(t_1) + (t - t_1)\vec{v} = \vec{p}$; thus $\vec{p}(t_n) = \vec{p} - (t - t_n)\vec{v}$ for all recent times $t_n \approx t$.

3 Moving Least-Squares Fit

Each reading from the cricket system gives us a time, distance pair, (t_n, d_n). For each of these readings, we may associate our predicted distance, $h_n = ||B^n - p(t_n)||$, based on the known beacon location, when the reading occurred, and the state variables $\vec{p}$ and $\vec{v}$. To obtain an optimal fit on the cricket data, we want to minimize a cost function of the form $J = \sum_n (h_n - d_n)^2$.

However, this cost function contains square roots in h_n and is therefore hard to solve for. Since we really only need $h_n = d_n$ and both h_n and d_n are positive, our new least-squares cost function is $J = \sum_n (h_n^2 - d_n^2)^2$. This cost is almost the same as the optimal cost, but it weighs errors slightly differently. Most importantly, squaring h_n before subtraction d_n eliminates the difficult square roots.

By setting $h_n^2 - d_n^2 = 0$ for each beacon measurement received, we obtain N equations of the form

$$\begin{aligned} f_n : \; & h_n^2 - d_n^2 = \\ & B^n \cdot B^n - 2B^n \cdot \vec{p} + 2(t - t_n)B^n \cdot \vec{v} \\ & + \vec{p} \cdot \vec{p} - 2(t - t_n)\vec{p} \cdot \vec{v} + (t - t_n)^2 \vec{v} \cdot \vec{v} - d_n^2 = 0 \end{aligned} \quad (1)$$

Unfortunately, these equations still contain the nonlinear terms $\vec{p} \cdot \vec{p}$, $\vec{p} \cdot \vec{v}$, and $\vec{v} \cdot \vec{v}$. By taking difference between f_n for different readings, these nonlinearities may be cancelled out. First, re-arrange f_n to separate out the nonlinearities:

$$\begin{aligned} f_n :\ & \vec{p}\cdot\vec{p} - 2(t-t_n)\vec{p}\cdot\vec{v} + (t-t_n)^2\vec{v}\cdot\vec{v} \\ & = d_n^2 - B^n \cdot B^n + 2B^n\cdot\vec{p} - 2(t-t_n)B^n\cdot\vec{v} \end{aligned} \tag{2}$$

Taking the difference between two of these equations will drop out the $\vec{p}\cdot\vec{p}$ term. For example,

$$\begin{aligned} f_2 - f_1 :\ & [\vec{p}\cdot\vec{p} - 2(t-t_2)\vec{p}\cdot\vec{v} + (t-t_2)^2\vec{v}\cdot\vec{v}] \\ & - [\vec{p}\cdot\vec{p} - 2(t-t_1)\vec{p}\cdot\vec{v} + (t-t_1)^2\vec{v}\cdot\vec{v}] \\ & = [d_2^2 - B^2\cdot B^2 + 2B^2\cdot\vec{p} - 2(t-t_2)B^2\cdot\vec{v}] \\ & - [d_1^2 - B^1\cdot B^1 + 2B^1\cdot\vec{p} - 2(t-t_1)B^1\cdot\vec{v}] \end{aligned} \tag{3}$$

Which simplifies to

$$\begin{aligned} & 0{+}2(t_2-t_1)\vec{p}\cdot\vec{v} + (t_2^2 - 2t(t_2-t_1) - t_1^1)\vec{v}\cdot\vec{v} \\ & = (d_2^2 - d_1^2) - (B^2\cdot B^2 - B^1\cdot B^1) \\ & + 2(B^2 - B^1)\cdot\vec{p} - 2((t-t_2)B^2 - (t-t_1)B^1)\cdot\vec{v} \end{aligned} \tag{4}$$

The weighted difference of two such differences would then drop out the $\vec{p}\cdot\vec{v}$ terms; and this could be repeated to eliminate $\vec{v}\cdot\vec{v}$. However, such a process may amplify the variance inherent in each individual reading. Instead, we use all the measurements to calculate estimates of the nonlinear terms as linear functions of $\vec{p}$ and $\vec{v}$. To start, the following sums will be useful.

$$\begin{aligned} N &= \sum_n (1) \\ N_1 &= \sum_n (t-t_n) \\ N_2 &= \sum_n (t-t_n)^2 \\ N_3 &= \sum_n (t-t_n)^3 \\ N_4 &= \sum_n (t-t_n)^4 \end{aligned}$$

These appear in the following weighted sums of f_n, which express the nonlinear terms as linear functions of the unknowns.

$$
\begin{aligned}
S_{pp} =& \sum_n (t-t_n)^2 f_n \qquad (5)\\
=& N_2 \vec{p}\cdot\vec{p} - 2N_3\vec{p}\cdot\vec{v} + N_4\vec{v}\cdot\vec{v}\\
=& \sum_n (t-t_n)^2(d_n^2 - B^n\cdot B^n) + 2\sum_n (t-t_n)^2(B^n\cdot\vec{p}) - 2\sum_n (t-t_n)^3(B^n\cdot\vec{v})\\
S_{pv} =& \sum_n (t-t_n) f_n \qquad (6)\\
=& N_1 \vec{p}\cdot\vec{p} - 2N_2\vec{p}\cdot\vec{v} + N_3\vec{v}\cdot\vec{v}\\
=& \sum_n (t-t_n)(d_n^2 - B^n\cdot B^n) + 2\sum_n (t-t_n)(B^n\cdot\vec{p}) - 2\sum_n (t-t_n)^2(B^n\cdot\vec{v})\\
S_{vv} =& \sum_n f_n \qquad (7)\\
=& N \vec{p}\cdot\vec{p} - 2N_1\vec{p}\cdot\vec{v} + N_2\vec{v}\cdot\vec{v}\\
=& \sum_n (d_n^2 - B^n\cdot B^n) + 2\sum_n (B^n\cdot\vec{p}) - 2\sum_n (t-t_n)(B^n\cdot\vec{v})
\end{aligned}
$$

Notice that there are two ways of expressing each of these sums; one is linear, and the other is nonlinear. These may be expressed in matrix form as

$$
\begin{bmatrix} S_{pp} \\ S_{pv} \\ S_{vv} \end{bmatrix} = \mathcal{N} \begin{bmatrix} \vec{p}\cdot\vec{p} \\ -2\vec{p}\cdot\vec{v} \\ \vec{v}\cdot\vec{v} \end{bmatrix} = \mathcal{L} \begin{bmatrix} 1 \\ 2\vec{p} \\ -2\vec{v} \end{bmatrix} \qquad (8)
$$

where

$$
\mathcal{N} = \begin{bmatrix} N_2 & N_3 & N_4 \\ N_1 & N_2 & N_3 \\ N & N_1 & N_2 \end{bmatrix} \qquad (9)
$$

and

$$
\mathcal{L} = \begin{bmatrix} \sum_n (t-t_n)^2(d_n^2 - B^n\cdot B^n) & \sum_n (t-t_n)^2 B^n & \sum_n (t-t_n)^3 B^n \\ \sum_n (t-t_n)(d_n^2 - B^n\cdot B^n) & \sum_n (t-t_n) B^n & \sum_n (t-t_n)^2 B^n \\ \sum_n (d_n^2 - B^n\cdot B^n) & \sum_n B^n & \sum_n (t-t_n) B^n \end{bmatrix} \qquad (10)
$$

These linear equations are easily solved to obtain

$$\begin{bmatrix} \vec{p}\cdot\vec{p} \\ -2\vec{p}\cdot\vec{v} \\ \vec{v}\cdot\vec{v} \end{bmatrix} = \mathcal{N}^{-1}\mathcal{L} \begin{bmatrix} 1 \\ 2\vec{p} \\ -2\vec{v} \end{bmatrix} \tag{11}$$

By defining $T = [1,\ t - t_n,\ (t - t_n)^2]$, the nonlinearities can now be removed from f_n (2) since

$$\vec{p}\cdot\vec{p} - 2(t-t_n)\vec{p}\cdot\vec{v} + (t-t_n)^2\vec{v}\cdot\vec{v} = T\cdot \begin{bmatrix} \vec{p}\cdot\vec{p} \\ -2\vec{p}\cdot\vec{v} \\ \vec{v}\cdot\vec{v} \end{bmatrix} = T\cdot\mathcal{N}^{-1}\mathcal{L} \begin{bmatrix} 1 \\ 2\vec{p} \\ -2\vec{v} \end{bmatrix} \tag{12}$$

Thus a least-squares fit to the position-velocity model can be obtained by solving N equations of the form

$$\left(\begin{bmatrix} 0 & B^n & (t-t_n)B^n \end{bmatrix} - T\cdot\mathcal{N}^{-1}\mathcal{L} \right) \begin{bmatrix} 0 \\ 2\vec{p} \\ -2\vec{v} \end{bmatrix} = -d_n^2 + B^n\cdot B^2 + T\cdot\mathcal{N}^{-1}\mathcal{L} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

3.1 Algorithm

Here is the proposed algorithm for the above result. To represent the formulas for S_{pp}, S_{pv}, and S_{vv}, we write them as

$$\begin{aligned}
S_{pp} =& \sum_n (d_n^2 - B^n \cdot B^n) + 2\sum_n (B^n) \cdot \vec{p} + 2\sum_n ((t-t_n)B^n) \cdot \vec{v} \\
=& \text{s1c} + 2([\text{s1px, s1py, s1pz}] \cdot \vec{p} + [\text{s1vx, s1vy, s1vz}] \cdot \vec{v}) \\
S_{pv} =& \sum_n \frac{d_n^2 - B^n \cdot B^n}{t-t_n} + 2\sum_n \frac{B^n}{t-t_n} \cdot \vec{p} + 2\sum_n (B^n) \cdot \vec{v} \\
=& \text{s2c} + 2([\text{s2px, s2py, s2pz}] \cdot \vec{p} + [\text{s2vx, s2vy, s2vz}] \cdot \vec{v}) \\
S_{vv} =& \sum_n \frac{d_n^2 - B^n \cdot B^n}{(t-t_n)^2} + 2\sum_n \frac{B^n}{(t-t_n)^2} \cdot \vec{p} + 2\sum_n \frac{B^n}{t-t_n} \cdot \vec{v} \\
=& \text{s3c} + 2([\text{s3px, s3py, s3pz}] \cdot \vec{p} + [\text{s3vx, s3vy, s3vz}] \cdot \vec{v})
\end{aligned}$$

To represent the formula for $C(\vec{p}, \vec{v})$, we shall write it as

$$\begin{aligned}
C(\vec{p}, \vec{v}) &= C_c + C_p\vec{p} + C_v\vec{v} \\
&= \begin{bmatrix} \text{ccx} \\ \text{ccy} \\ \text{ccz} \end{bmatrix} + \begin{bmatrix} \text{cpxx} & \text{cpxy} & \text{cpxz} \\ \text{cpyx} & \text{cpyy} & \text{cpyz} \\ \text{cpzx} & \text{cpzy} & \text{cpzz} \end{bmatrix} \vec{p} + \begin{bmatrix} \text{cvxx} & \text{cvxy} & \text{cvxz} \\ \text{cvyx} & \text{cvyy} & \text{cvyz} \\ \text{cvzx} & \text{cvzy} & \text{cvzz} \end{bmatrix} \vec{v}
\end{aligned}$$

```
t - array of times
bd - array of beacon distances
bx, by, bz - arrays of beacon coordinates
N:=number of readings
N1:=0
N2:=0
N_1:=0
N_2:=0
s1* = s2* = s3* :=0

// Generate the sums
for(n=0; n<N; n++)
{
  dt:= t[N-1]-t[n]
  dt2:= dt*dt
  N1+= dt
  N2+= dt2
  N_1+= 1/dt
  N_2+= 1/dt2

  cn:= bd[n]*bd[n] - bx[n]*bx[n] - by[n]*by[n] - bz[n]*bz[n]
  s1c+= cn
  s2c+= cn/dt
  s3c+= cn/dt2

  s1px+= bx[n]
  s2px+= bx[n]/dt
  s3px+= bx[n]/dt2
  // repeat for py and pz

  s1vx+= bx[n]*dt
  s2vx+= bx[n]
  s3vx+= bx[n]/dt
  // repeat for vy and vz
}
```

```
// Compute the inverse matrix B such that A*B=I
den:= N*N*N + N1*N1*N_2 + N2*N_1*N_1 - N*(2*N1*N_1 + N2*N_2)
b11:= N*N - N1*N_1
b12:= N2*N_1 - N1*N
b13:= N1*N1 - N2*N
b21:= N1*N_2 - N*N_1
b22:= N*N - N2*N_2
b23:= b12
b31:= N_1*N_1 - N*N_2
b32:= b21
b33:= b11

// Calculate C(p,v)=B*[Spp Spv Svv]'
ccx:=    (b11*s1c  + b12*s2c  + b13*s3c )/den
cpxx:= 2*(b11*s1px + b12*s2px + b13*s3px)/den
cpxy:= 2*(b11*s1py + b12*s2py + b13*s3py)/den
cpxz:= 2*(b11*s1pz + b12*s2pz + b13*s3pz)/den
cvxx:= 2*(b11*s1vx + b12*s2vx + b13*s3vx)/den
cvxy:= 2*(b11*s1vy + b12*s2vy + b13*s3vy)/den
cvxz:= 2*(b11*s1vz + b12*s2vz + b13*s3vz)/den

ccy:=    (b21*s1c  + b22*s2c  + b23*s3c )/den
cpyx:= 2*(b21*s1px + b22*s2px + b23*s3px)/den
cpyy:= 2*(b21*s1py + b22*s2py + b23*s3py)/den
cpyz:= 2*(b21*s1pz + b22*s2pz + b23*s3pz)/den
cvyx:= 2*(b21*s1vx + b22*s2vx + b23*s3vx)/den
cvyy:= 2*(b21*s1vy + b22*s2vy + b23*s3vy)/den
cvyz:= 2*(b21*s1vz + b22*s2vz + b23*s3vz)/den

ccz:=    (b31*s1c  + b32*s2c  + b33*s3c )/den
cpzx:= 2*(b31*s1px + b32*s2px + b33*s3px)/den
cpzy:= 2*(b31*s1py + b32*s2py + b33*s3py)/den
cpzz:= 2*(b31*s1pz + b32*s2pz + b33*s3pz)/den
cvzx:= 2*(b31*s1vx + b32*s2vx + b33*s3vx)/den
cvzy:= 2*(b31*s1vy + b32*s2vy + b33*s3vy)/den
cvzz:= 2*(b31*s1vz + b32*s2vz + b33*s3vz)/den
```

```
// Set up the least-squares fit ‘Ax=B’
LSQA:= allocate space for an Nx6 matrix - coefficients of p and v
LSQx:= allocate space for a 6x1 matrix  - solution for p and v
LSQB:= allocate space for an Nx1 matrix - constants

for(n=0; n<N; n++)
{
  dt:= t[N-1]-t[n]
  dt2:= dt*dt

  // 2*B[n] - T*Cp
  LSQA[i,0]:= 2*bx[n] - cpxx - dt*cpyx - dt2*cpzx
  LSQA[i,1]:= 2*by[n] - cpxy - dt*cpyy - dt2*cpzy
  LSQA[i,2]:= 2*bz[n] - cpxz - dt*cpyz - dt2*cpzz

  // -2*(t-tn)*B[n] - T*Cv
  LSQA[i,3]:= -2*dt*bx[n] - cvxx - dt*cvyx - dt2*cvzx
  LSQA[i,4]:= -2*dt*by[n] - cvxy - dt*cvyy - dt2*cvzy
  LSQA[i,5]:= -2*dt*bz[n] - cvxz - dt*cvyz - dt2*cvzz

  LSQB[i]:= bx[n]*bx[n] + by[n]*by[n] + bz[n]*bz[n] - d[n]*d[n]
            + c1c + dt*c2c + dt2*c3c
}

// Perform the least-squares fit
```

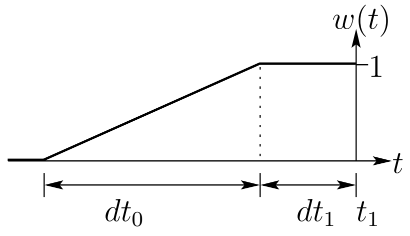

Figure 3: Hanning-like weighting function. Sample values: $dt_1 = 0.5$s, $dt_0 = 2.5$s.

4 Weighted Least-Squares Fit

Since the robot can accelerate, we want to give recent measurements more weight than older measurements. Thus the weighting function of Figure 3 is used. This function is given by:

$$w(t) = \begin{cases} 1 & t_1 - dt_1 \le t \\ (t - dt_1)/dt_0 & t_1 - (dt_0 + dt_1) < t < t_1 - dt_1 \\ 0 & \text{else} \end{cases} \tag{14}$$

This adjusts our cost function to become $J = \sum_n w_n(h_n^2 - d_n^2)$.

5 Extended Kalman Filter

Variable	Size	Meaning
X_k^-	6x1	Estimated system state $(x, \vec{v})$
X_k^+	6x1	EKF corrected system state
K_k	6x1	Kalman gain matrix
P_k	6x6	Covariance matrix
R_k	1x1	Estimated distance variance
d_k	1x1	Reported distance measurement

6 Fixed Gain Filter

Assume we have a good estimate of our last state $(x_{k-1}^+, \vec{v}_k)$. Use it to predict our current state, $x_k^- = x_{k-1}^+ + \vec{v}_k \Delta t$. Now look at the beacon's distance. This projects a sphere of radius d_k around the beacon's origin, B_k. Moving along

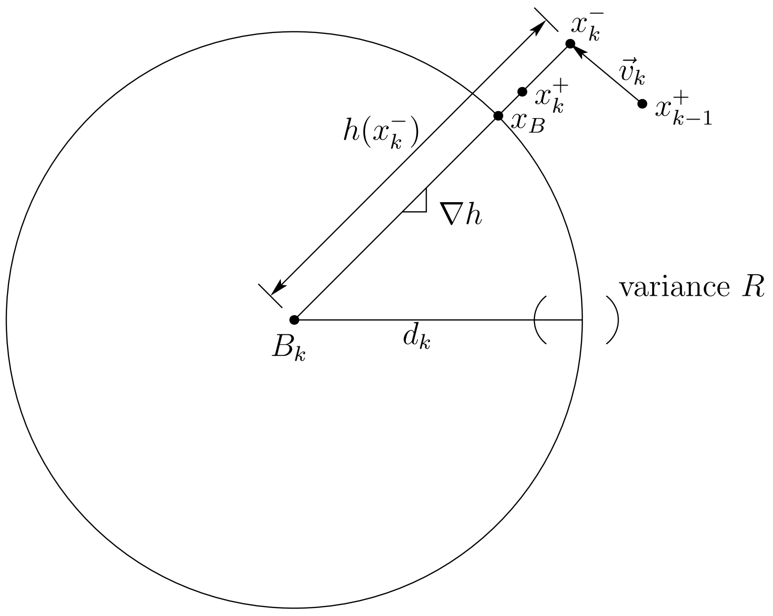

Figure 4: The basic Cricket filter design

the radius between x_k^- and B_k, the nearest point indicated by the beacon is denoted as x_B.

Since the signals are noisy, there is some variance R in the distance measurement. Ignoring this variance, we would just set $x_k^+ = x_B$
$= x_k^- + (d_k - h(x_k^-))\nabla h$. Since we have variance, we set $x_k^+ = x_k^- + \theta(d_k - h(x_k^-))\nabla h$ for some $0 < \theta < 1$.

Our velocity estimate for the next time step is then $\vec{v}_{k+1} = (x_k^+ - x_{k-1}^+)/\Delta t$. Writing this in terms of the current estimate, this is $\vec{v}_{k+1} = \vec{v}_k + (x_k^+ - x_k^-)/\Delta t$.

The distance estimate $h(x, v)$ is defined as
$h(x, v) = \sqrt{((x_k^-)_x - B_x)^2 + ((x_k^-)_y - B_y)^2 + ((x_k^-)_z - B_z)^2}$

Thus, $\frac{\partial h}{\partial x} = \frac{(x_k^-)_x - B_x}{h}$, $\frac{\partial h}{\partial y} = \frac{(x_k^-)_y - B_y}{h}$, and $\frac{\partial h}{\partial z} = \frac{(x_k^-)_z - B_z}{h}$.