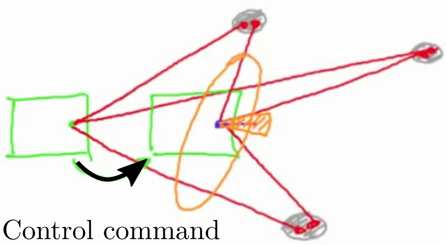
SLAM - Unit G: Particle Filter SLAM (FastSLAM).

Claus Brenner
claus.brenner@ikg.uni-hannover.de



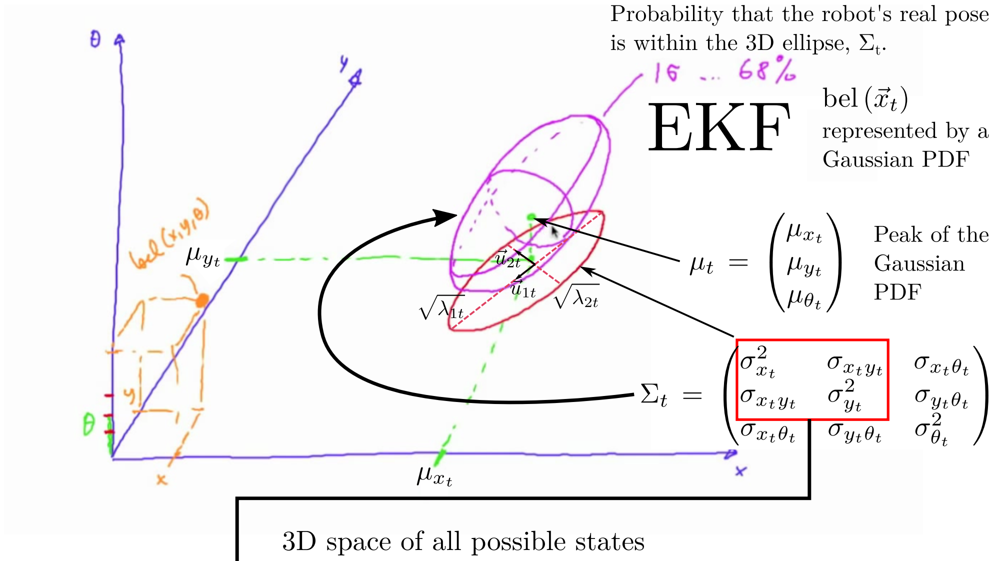
Control command

$$\vec{u}_t = \begin{pmatrix} l_t \\ r_t \end{pmatrix}$$

Extended Kalman Filter (EKF)

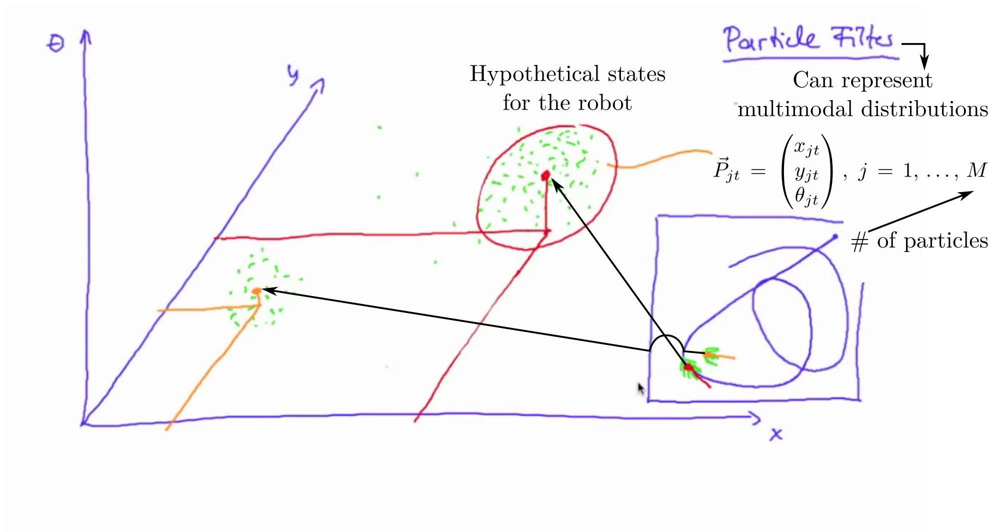
$$\mu_t = \begin{pmatrix} \mu_{x_t} \\ \mu_{y_t} \\ \mu_{\theta_t} \end{pmatrix}$$ Estimated state or corrected state

$$\Sigma_t = \begin{pmatrix} \sigma_{x_t}^2 & \sigma_{x_t y_t} & \sigma_{x_t \theta_t} \\ \sigma_{x_t y_t} & \sigma_{y_t}^2 & \sigma_{y_t \theta_t} \\ \sigma_{x_t \theta_t} & \sigma_{y_t \theta_t} & \sigma_{\theta_t}^2 \end{pmatrix}$$

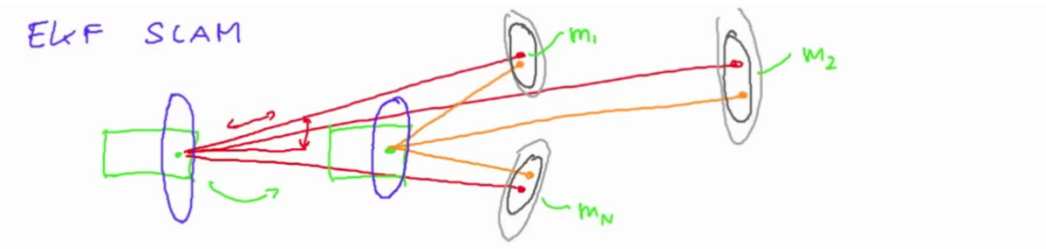


$$
\begin{pmatrix} \sigma_{x_t}^2 & \sigma_{x_t y_t} \\ \sigma_{x_t y_t} & \sigma_{y_t}^2 \end{pmatrix} = U \cdot D \cdot U^T
$$

$$
= \underbrace{\left(\vec{u}_{1t},\, \vec{u}_{2t}\right)}_{\substack{\text{Eigenvectors} \\ \text{(Ellipse axes' directions)}}} \cdot \underbrace{\begin{pmatrix} \lambda_{1t} & 0 \\ 0 & \lambda_{2t} \end{pmatrix}}_{\text{Eigenvalues}} \cdot \begin{pmatrix} \vec{u}_{1t}^T \\ \vec{u}_{2t}^T \end{pmatrix}
$$



The landmarks' positions were given in advanced in the EKF lecture and in the PF lecture, and also they were fixed values.



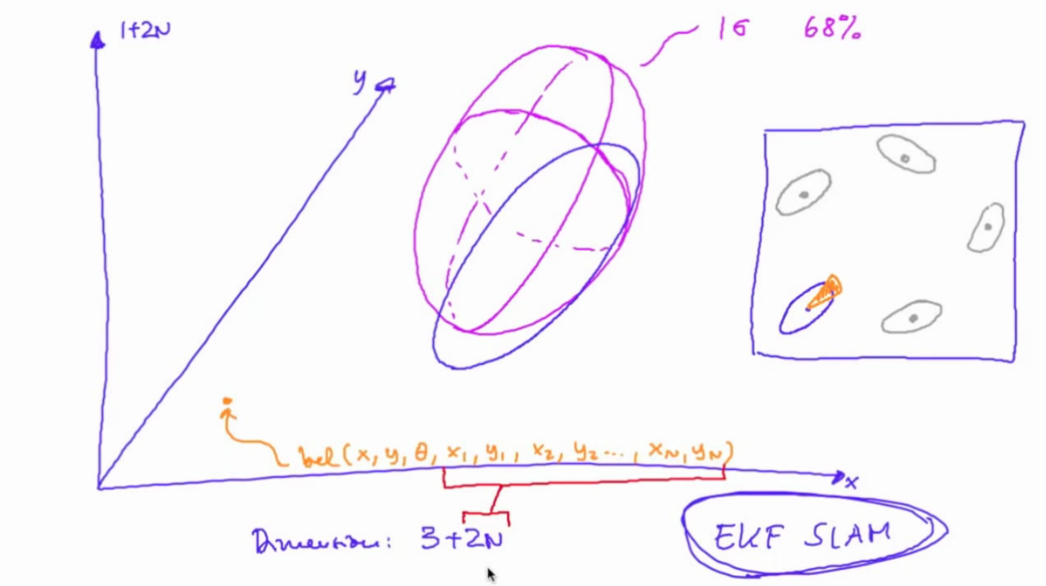
$$\vec{\mu}_t = \begin{pmatrix} \mu_{x_t} \\ \mu_{y_t} \\ \mu_{\theta_t} \\ \mu_{x_{L_1}} \\ \mu_{y_{L_1}} \\ \vdots \\ \mu_{x_{L_j}} \\ \mu_{y_{L_j}} \\ \vdots \\ \mu_{x_{L_N}} \\ \mu_{y_{L_N}} \end{pmatrix} \begin{array}{l} 3 \\ 2N \end{array} \qquad \Sigma_t = \begin{pmatrix} \sigma^2_{x_t} & \sigma_{x_t y_t} & \sigma_{x_t \theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \sigma_{x_t y_t} & \sigma^2_{y_t} & \sigma_{y_t \theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ \sigma_{x_t \theta_t} & \sigma_{y_t \theta_t} & \sigma^2_{\theta_t} & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{x_{L_0}} & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{y_{L_0}} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \sigma^2_{x_{L_j}} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & \sigma^2_{y_{L_j}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & \sigma^2_{x_{L_N}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \sigma^2_{y_{L_N}} \end{pmatrix} \begin{array}{l} 3 \\ 2N \end{array}$$

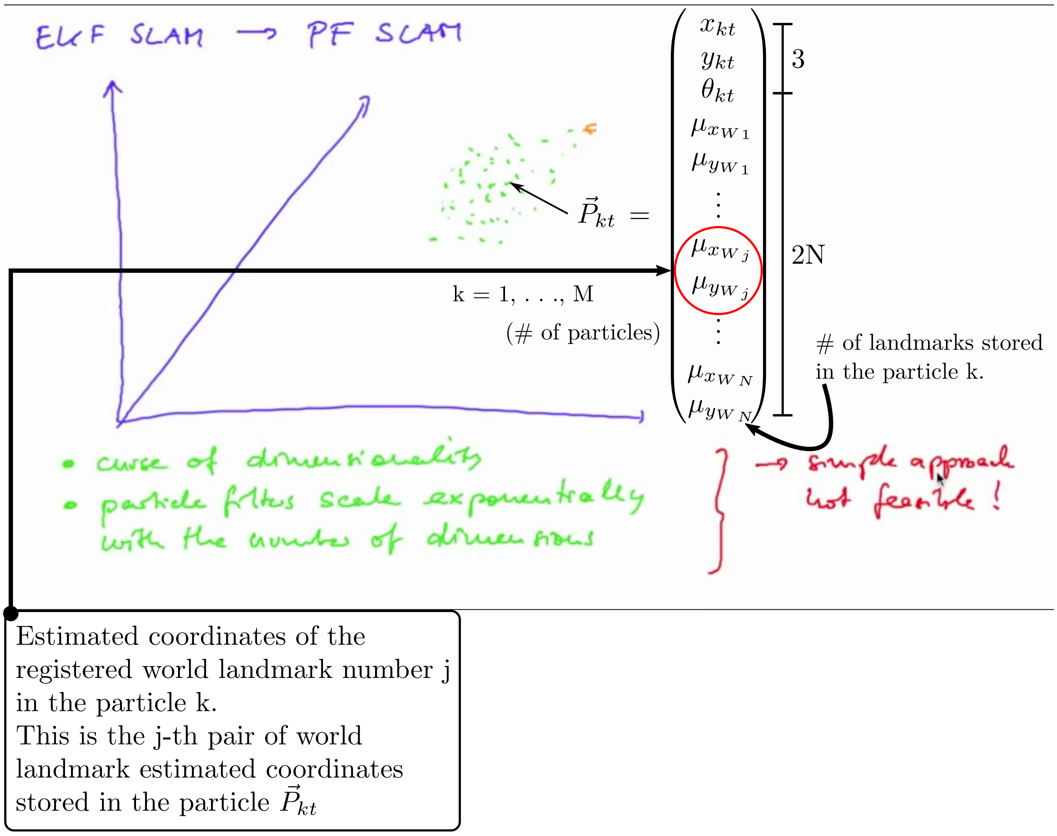
$$\underbrace{\hphantom{000}}_{3}\underbrace{\hphantom{0000000000000}}_{2N}$$

Landmarks' estimated coordinates

Landmarks' variances

In this lecture the robot observes landmarks whose positions are not given in advanced. The landmarks' positions are stocastic as well as the robot's pose.





Curse of dimensionality:

The particle filter scales exponentially with the number of dimensions. This means that if the dimension of the search space increase, the number of particles that are necessary to represent the posterior distribution scales exponentially. Therefore, this approach is unfeasible.

The new approach will be**the Rao-Blackwellized Particle Filter (RBPF)**, where the posterior PDF is factorized as:

Posterior PDF of the
Full-SLAM problem

Represented with particles

$$p\left(X_{1:t},\, \text{Map} \mid Z_{1:t},\, U_{1:t}\right) = p\left(\text{Map} \mid X_{1:t},\, Z_{1:t},\, U_{1:t}\right)\, p\left(X_{1:t} \mid Z_{1:t},\, U_{1:t}\right)$$

Entire path
$X_{1:t} = \{\vec{x}_1,\, \vec{x}_2,\, \ldots,\, \ldots,\, \vec{x}_{t-1},\, \vec{x}_t\}$

$$= \left(\prod_{j=1}^{N} p\left(\vec{p}_{W\,j} \mid X_{1:t},\, Z_{1:t}\right)\right) p\left(X_{1:t} \mid Z_{1:t},\, U_{1:t}\right)$$

Conditional independence:
If the path $X_{1:t}$ is known, then the
positions of the landmarks are independent!

PDF of the landmark position given
the entire path and all the measurements

$$\vec{p}_{W\,j} = \begin{pmatrix} x_{W\,j} \\ y_{W\,j} \end{pmatrix}, \quad j = 1,\, \ldots,\, N$$

The PDF $p\left(\vec{p}_{W\,j} \mid X_{1:t},\, Z_{1:t}\right)$ is represented as a Gaussian PDF.

An independent EKF is used
for each registered world landmark

The term

$$\vec{p}_{W\,j} = \begin{pmatrix} x_{W\,j} \\ y_{W\,j} \end{pmatrix}$$

$j = 1,\, \ldots,\, N$, represents the random coordinates of the registered world landmark number$j$ within the state vector$\vec{x}_t$.
The term

$$\vec{\mu}_{L_j} = \begin{pmatrix} \mu_{x_{W\,j}} \\ \mu_{y_{W\,j}} \end{pmatrix}$$

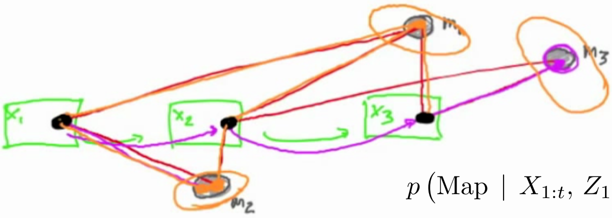
$j = 1,\, \ldots,\, N$, represents the estimated coordinates of the registered world landmark number$j$ within a specific particle,$\vec{P}_{kt},\ k = 1,\, \ldots,\, M$.
The index where the estimated$x$ coordinate for the registered world landmark$j$,$\mu_{x_{W\,j}}$, is stored within a specific particle,$\vec{P}_{kt}$, is[1]:

$$i = 3 + 2\,j - 1$$

The**Rao-Blackwellized Particle Filter (RBPF)** is given without a proof!
Summarize: The PDF of the robot's pose is represented using a particle filter. For each particle, the

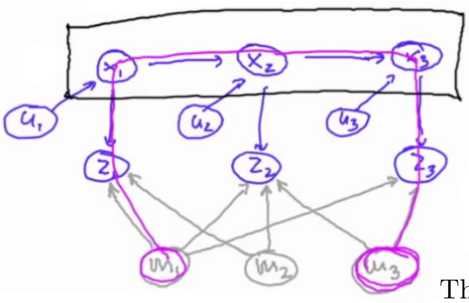
[1]The same idea applies for the random the random coordinate$x_{W\,j}$ within the state vector$\vec{x}_t$

remaining part of the posterior PDF is represented by an independent Gaussian PDF, which is the result of an independent EKF.



$$p\left(\text{Map} \mid X_{1:t},\, Z_{1:t},\, U_{1:t}\right) \;=\; \prod_{j=1}^{N} p\left(\vec{p}_{W\,j} \mid X_{1:t},\, Z_{1:t}\right)$$

Dynamic Bayes network generated from the above process



The term $\vec{p}_{W\,j} = \begin{pmatrix} x_{W\,j} \\ y_{W\,j} \end{pmatrix}$

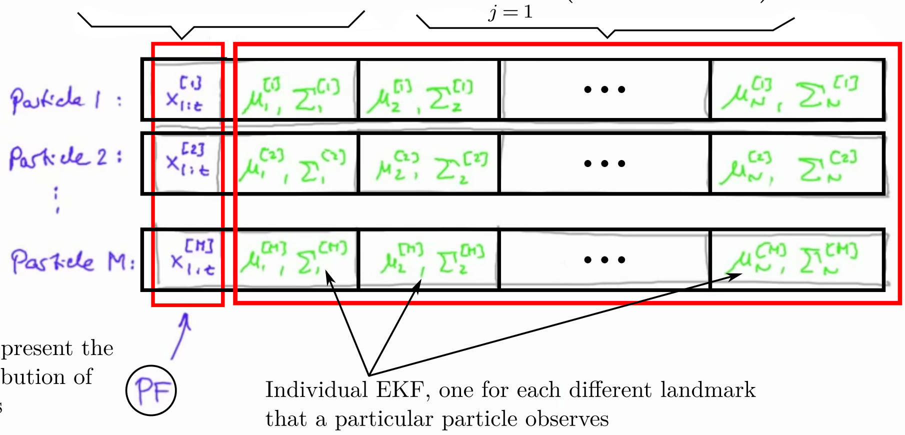
$j = 1, \ldots, N$, represents the random coordinates of the registered world landmark number j within the state vector $\vec{x}_t$.

The index where the random x coordinate for the registered world landmark $j, x_{W\,j}$, is stored within the state vector $\vec{x}_t$ is:

$$i \;=\; 3 \,+\, 2\,j \,-\, 1$$

$p\left(X_{1:t},\, \text{Map} \mid Z_{1:t},\, U_{1:t}\right) \;=$

$$\underbrace{p\left(X_{1:t} \mid Z_{1:t},\, U_{1:t}\right)} \qquad \underbrace{\prod_{j=1}^{N} p\left(\vec{p}_{W\,j} \mid X_{1:t},\, Z_{1:t}\right)}$$

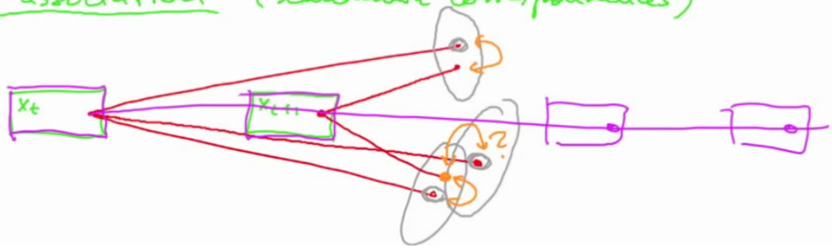


To represent the distribution of paths

Individual EKF, one for each different landmark that a particular particle observes

Fast-SLAM solves not only the FULL-SLAM problem but also the ONLINE-SLAM problem

Data association (landmark correspondences)



$$p\left(X_{1:t},\,\text{Map}\mid Z_{1:t},\,U_{1:t},\,C_{1:t}\right)\;=\;p\left(\text{Map}\mid X_{1:t},\,Z_{1:t},\,U_{1:t},\,C_{1:t}\right)\,p\left(X_{1:t}\mid Z_{1:t},\,U_{1:t},\,C_{1:t}\right)$$

$$=\left(\prod_{j=1}^{N} p\left(\vec{p}_{W\,j}\mid\,X_{1:t},\,Z_{1:t},\,C_{1:t}\right)\right)\,p\left(X_{1:t}\mid Z_{1:t},\,U_{1:t},\,C_{1:t}\right)$$

- Each particle maintains individual data associations!
- FastSLAM maintains posterior over <u>multiple</u> data associations!

The term $\vec{p}_{W\,j}=\begin{pmatrix}x_{W\,j}\\ y_{W\,j}\end{pmatrix}$
$j\,=\,1,\,\ldots,\,N$, represents the random coordinates of the registered world landmark number j within the state vector $\vec{x}_t$.

FastSLAM solves <u>full SLAM problem</u> ...

⋮

Particle i

$x^{(i)}_{\boxed{1:t}}$	$\mu^{(i)}_1,\ \Sigma^{(i)}_1$	$\mu^{(i)}_2,\ \Sigma^{(i)}_2$	...

↑ full path

... and the <u>online SLAM problem</u>!

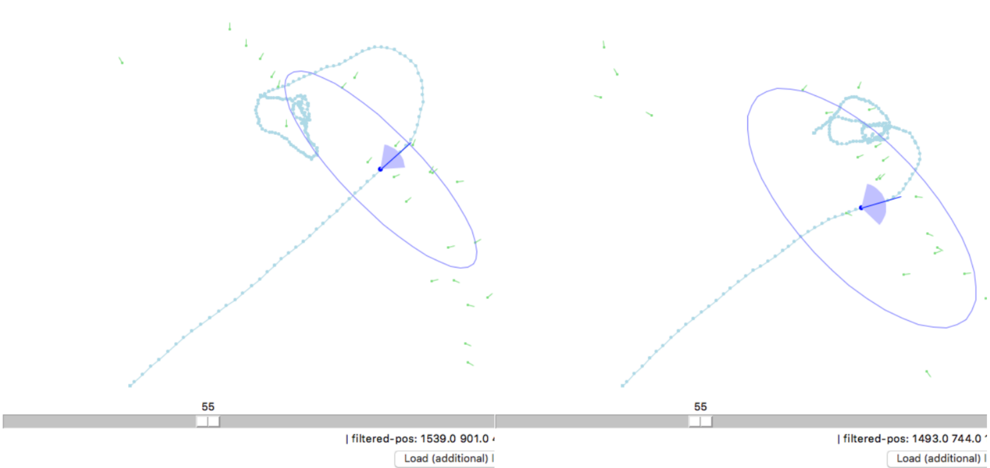
Particle i

$x^{(i)}_{\boxed{t}}$	$\mu^{(i)}_1,\ \Sigma^{(i)}_1$	$\mu^{(i)}_2,\ \Sigma^{(i)}_2$	

↑ only last pose

$\Rightarrow$ so we use it as a filter!

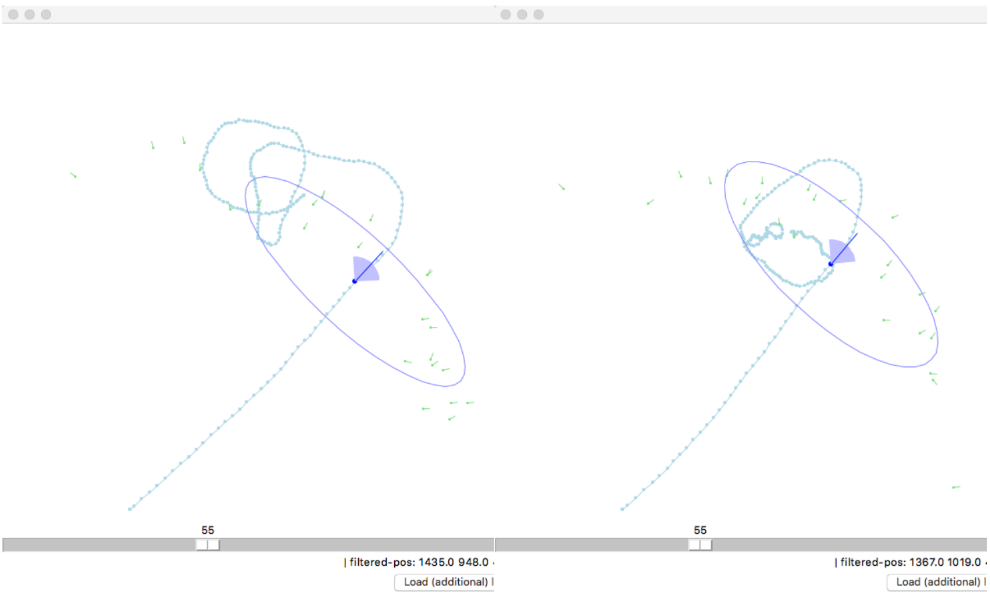
Prediction step only
(4 different tests)



Different trajectories due to the
error in the robot's motion

$$\sigma_{l_t}^2 = (p_1\, l_t)^2 + \left(p_2\,(l_t - r_t)\right)^2$$
$$\sigma_{r_t}^2 = (p_1\, r_t)^2 + \left(p_2\,(l_t - r_t)\right)^2$$

$$\text{left\_command} \sim \mathcal{N}\left(l_t,\, \sigma_{l_t}^2\right)$$
$$\text{right\_command} \sim \mathcal{N}\left(r_t,\, \sigma_{r_t}^2\right)$$

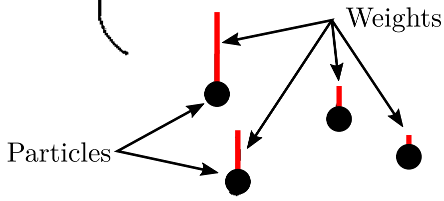


In class FastSLAM:

Function in the Fast_SLAM class

```
correct(cylinders)
    weights = update_and_compute_weights(cylinders)
    resample(weights)
```

Function programmed in the Particle Filter lecture




```
update_and_compute_weights(cylinders)
    weights = []
    for p in particles:
        weight = 1.0
        for m in measurement_list:
            w = p.update_particle(m)
            weight = weight * w
        weights.append(weight)
    return weights
```

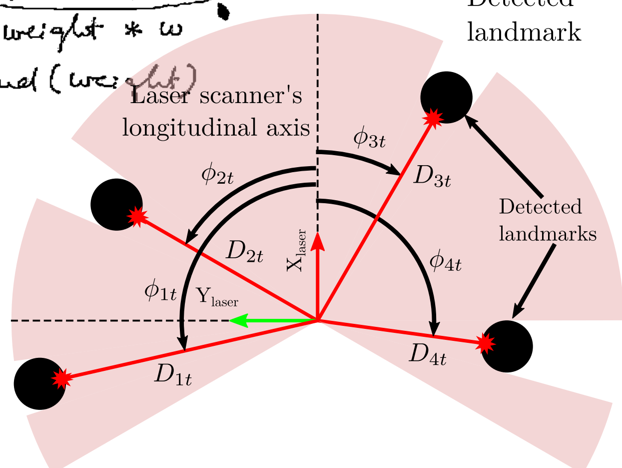
FastSLAM

Present all the measurements to all the particles

Function programmed in the Particle class

Measurement

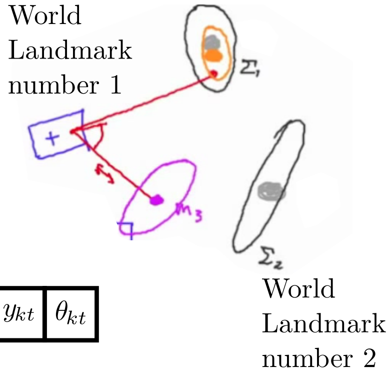
$$\vec{z}_t = \begin{pmatrix} D_t \\ \phi_t \end{pmatrix}$$



In class **Particle**:

$$\texttt{update\_particle(}\overbrace{(D_t,\ \phi_t)}^{\vec{z}_t\ \text{(measurement)}}\texttt{):}$$

A) Compute likelihood of correspondence and compare it to a threshold.
Depending on the result:
B) Initialize a new landmark
or
C) Update landmark j (EKF)

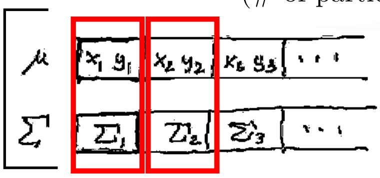


$\vec{P}_{kt}$

x_{kt}	y_{kt}	θ_{kt}

$k = 1, \ldots, M$
(# of particles)

For each particle



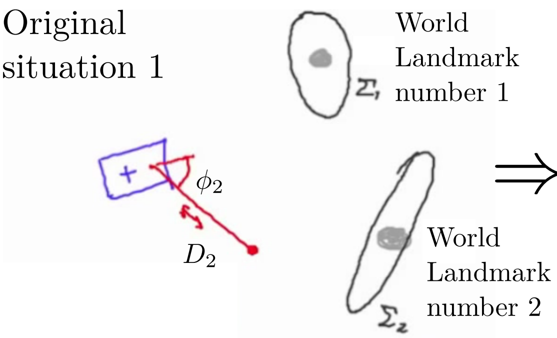
EKF_1 EKF_2

In the situation 1 the algorithm computes the likelihood between the measurement $\vec{z}_t$ and the landmark number 1.
In the situation 2 the algorithm computes the likelihood between the measurement $\vec{z}_t$ and the landmark number 2.

One EKF for each world landmark that a particular particle has registered.
Each EKF consists on two elements:
1º. Vector with the estimated coordinates for a registered world landmark.
2º. Covariance matrix with uncertainties.

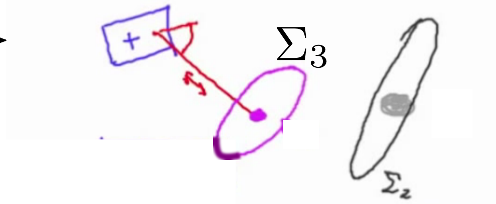
$\Downarrow$

Original situation 1

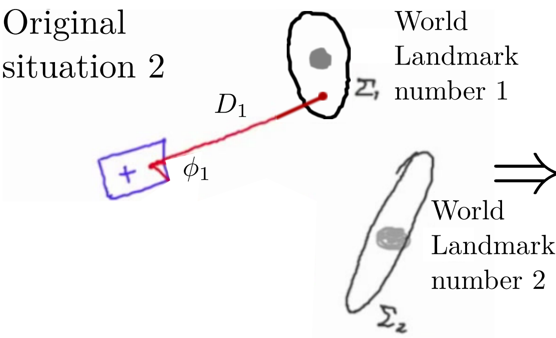


$\Rightarrow$

B stage
Register a new world landmark with the number 3.
Add to the particle: $(\mu_{x_{W3}}, \mu_{x_{W3}})$ and Σ_3

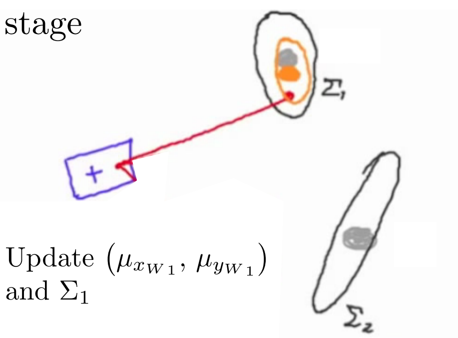


Original situation 2



$\Rightarrow$

C stage



Update $(\mu_{x_{W1}}, \mu_{y_{W1}})$ and Σ_1

$$h(\ldots) \begin{cases} D_{jt} = \sqrt{(x_{L_j}-x_\ell)^2 + (y_{L_j}-y_\ell)^2} \\ \phi_{jt} = \operatorname{atan}\left(\dfrac{y_{L_j}-y_\ell}{x_{L_j}-x_\ell}\right) - \theta \end{cases}$$

$$\underset{jt}{H} = \begin{bmatrix} \frac{\partial D}{\partial x_{L_j}} & \frac{\partial D}{\partial y_{L_j}} \\ \frac{\partial \phi}{\partial x_{L_j}} & \frac{\partial \phi}{\partial y_{L_j}} \end{bmatrix} \quad 2\times 2,\ \mathbb{R}$$

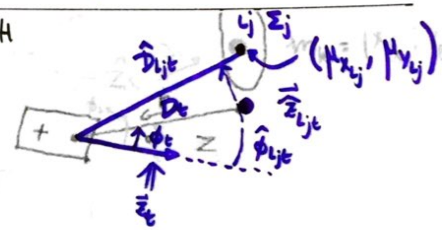
$$\frac{\partial D}{\partial x_{L_j}} = \frac{x_{L_j}-x_\ell}{\sqrt{q}} \qquad \frac{\partial D}{\partial y_{L_j}} = \frac{y_{L_j}-y_\ell}{\sqrt{q}}$$

$$\frac{\partial \phi}{\partial x_{L_j}} = -\frac{(y_{L_j}-y_\ell)}{q} \qquad \frac{\partial \phi}{\partial y_{L_j}} = \frac{x_{L_j}-x_\ell}{q}$$

$$\sqrt{q} = \sqrt{(x_{L_j}-x_\ell)^2 + (y_{L_j}-y_\ell)^2}$$

Ⓐ compute likelihoods.

↑ In slam lecture: $2\times(3+2N)$ H



Expected measurement:

$j = 1 \ldots N$, # of landmarks

$k = 1 \ldots M$, # of particles.

$$\begin{bmatrix} \hat{D}_{jt} \\ \hat{\phi}_{jt} \end{bmatrix} = \hat{\vec{z}}_{jt} = h\big(\overbrace{(x_{it}, y_{it}, \theta_{it})}^{\text{Particle } P_{it}},\ j\big)$$

$$H = \frac{\partial h}{\partial \text{landmark}}\bigg|_{(x_{it}, y_{it}, \theta_{it}, j)} \qquad (2\times 2)$$

$x_{it}, y_{it}, \theta_{it}$

= Particle i, at time t, P_{it}, $i = 1 \ldots M$

μ	...	$x_k\ y_k$	...
Σ_i	...	Σ_k	...

list of landmark coordinates the particle P_i has.

Covariance of the term: $(\vec{z}_t - \hat{\vec{z}}_{L_jt})$

$$Q_{jt} = \underbrace{H_{jt}\, \Sigma_{jt}\, H_{jt}^T}_{\text{measurement variance due to the landmark variance}} + \underbrace{Q}_{\text{variance due to actual measurement } \vec{z}_t}$$

aka (Q_ℓ)

Σ_{jt}: Variance of landmark k (in x, y)

$$Q = \begin{bmatrix} \sigma_D^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}$$

In the SLAM-EKF lecture we used $H_{tk} \in \mathbb{R}^{2\times(3+2N)}$ and $\bar{\Sigma}_t \in \mathbb{R}^{(3+2N)\times(3+2N)}$

and now, in this lecture, we use

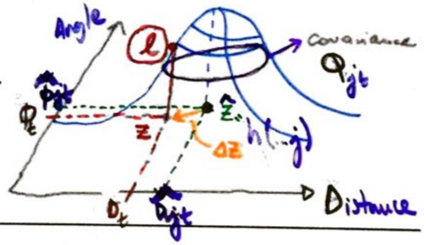
$H_{tk} \in \mathbb{R}^{2\times 2}$ and $\Sigma_k \in \mathbb{R}^{2\times 2}$

$$\Delta z_{jt} = z_t - \hat{z}_{jt}$$

Likelihood that the measurement $\vec{z}_t$ is due to the observation of Landmark L_j $(\mu_{x_{L_j}}, \mu_{y_{L_j}})$

$$\ell = \frac{1}{2\pi\sqrt{\det(Q_{jt})}} \cdot e^{-\frac{1}{2}\Delta z_{jt}^T Q_{jt}^{-1} \Delta z_{jt}}$$

"α_{jt}"

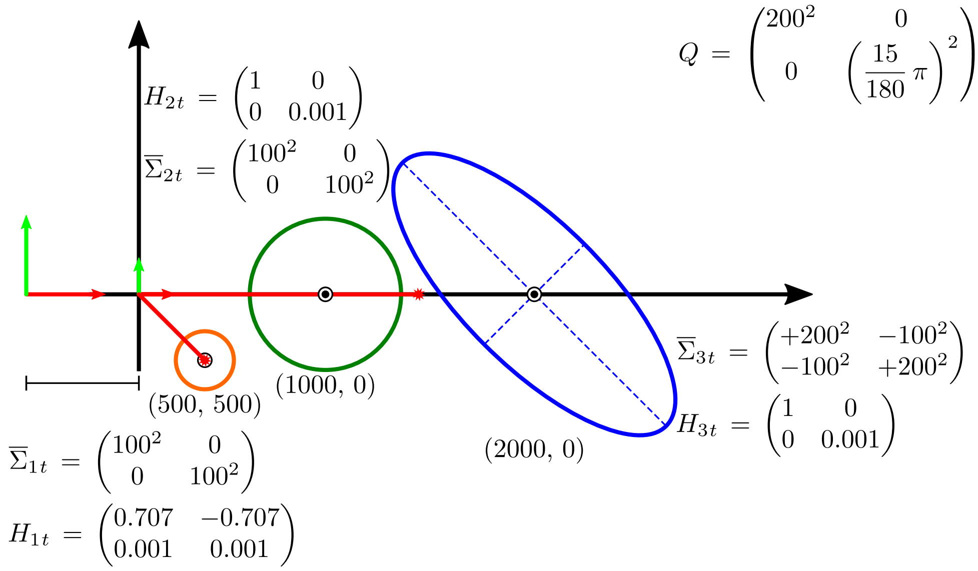


$$\hat{\vec{z}}_{jt} = \begin{bmatrix} \hat{D}_{jt} \\ \hat{\phi}_{jt} \end{bmatrix} = h(\vec{P}_{it}, j)$$

$$\vec{z}_t = \begin{bmatrix} D_t \\ \phi_t \end{bmatrix}$$

$$\underset{\text{"N}_{-}\text{t"}}{\Sigma_k} = \begin{bmatrix} \sigma^2_{x_{\ell k}} & \sigma_{x_{\ell k} y_{\ell k}} \\ \sigma_{x_{\ell k} y_{\ell k}} & \sigma^2_{y_{\ell k}} \end{bmatrix}$$

Q_{tk} (explanation) ⇒ The uncertainty in the value $h_{tk}(p_{it}, k)$, i.e, Q_{tk}, is due to the uncertainty of the landmark "k", i.e, Σ_k, which then translates into uncertainty in a distance and bearing angle, i.e, $H_{tk}\Sigma_k H_{tk}^T$, plus the uncertainty of the actual measurement of the sensor, i.e, Q_z



$$Q_{j_t} = H_{j_t} \cdot \overline{\Sigma}_{j_t} \cdot {H_{j_t}}^T + Q$$

$$Q_{1_t} = \begin{pmatrix} 5 \cdot 10^4 & 0 \\ 0 & 0.089 \end{pmatrix} \qquad Q_{2_t} = \begin{pmatrix} 5 \cdot 10^4 & 0 \\ 0 & 0.079 \end{pmatrix} \qquad Q_{3_t} = \begin{pmatrix} 8 \cdot 10^4 & -5 \\ -5 & 0.079 \end{pmatrix}$$

$$\vec{z}_{1_t} = \begin{pmatrix} 500\sqrt{2} \\ \arctan(1) \end{pmatrix} = \begin{pmatrix} 707.107 \\ 45^\circ \end{pmatrix} \qquad \vec{\hat{z}}_{1_t} = \begin{pmatrix} 500\sqrt{2} \\ \arctan(1) \end{pmatrix} = \begin{pmatrix} 707.107 \\ 45^\circ \end{pmatrix}$$

$$\vec{z}_{2_t} = \begin{pmatrix} 1500 \\ 0 \end{pmatrix} \qquad \vec{\hat{z}}_{2_t} = \begin{pmatrix} 1000 \\ 0 \end{pmatrix}$$

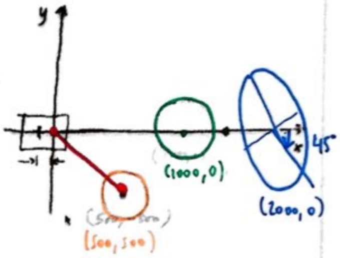
$$\vec{\hat{z}}_{3_t} = \begin{pmatrix} 2000 \\ 0 \end{pmatrix}$$

$$\mathscr{L}_{jit} = \frac{1}{2\pi\sqrt{|Q_{jt}|}}\, e^{-\frac{1}{2}\left(\vec{z}_{it} - \vec{\hat{z}}_{jt}\right)^T \cdot Q_{jt}^{-1} \cdot \left(\vec{z}_{it} - \vec{\hat{z}}_{jt}\right)}$$

$$\begin{aligned} \mathscr{L}_{11} &= 0.002 & \mathscr{L}_{21} &= 2.122 \cdot 10^{-5} & \mathscr{L}_{31} &= 1.366 \cdot 10^{-7} \\ \mathscr{L}_{12} &= 1.366 \cdot 10^{-7} & \mathscr{L}_{22} &= 0.000208 & \mathscr{L}_{32} &= 0.000419 \end{aligned}$$

```python
        [-100.0**2,  200.0**2 ]]))

if __name__ == '__main__':
    # Robot constants.
    scanner_displacement = 30.0

    # Filter constants.
    measurement_distance_stddev = 200.0  # Distance measurement error of cylinders.
    measurement_angle_stddev = 15.0 / 180.0 * pi  # Angle measurement error.
    Qt_measurement_covariance = \
        np.diag([measurement_distance_stddev**2,
                 measurement_angle_stddev**2])

    # Define a particle: position (x, y) and orientation.
    p = Particle(np.array([-scanner_displacement, 0.0, 0.0]))
    add_landmarks(p)
    N = p.number_of_landmarks()

    # Compute expected measurements.
    for i in xrange(N):
        print "Landmark", i, "----------"
        em = p.h_expected_measurement_for_landmark(i, scanner_displacement)
        print " Expected range:", em[0], "bearing [deg]", em[1]/pi*180
        H, Ql = p.H_Ql_jacobian_and_measurement_covariance_for_landmark(
            i, Qt_measurement_covariance, scanner_displacement)
        print " Covariance of measurement:\n", Ql

    # Compute correspondence likelihoods.
    # Define a set of measurements: (range, bearing).
    print "Measurement likelihoods ----------"
    measurements = [
        ("close to landmark 0", np.array([500*sqrt(2), -pi/4])),
        ("exactly between landmark 1 and 2", np.array([1500.0, 0.0]))
        ]
    for (text, m) in measurements:
        print "Likelihoods for measurement", text
        likelihoods = p.compute_correspondence_likelihoods(
            m, N, Qt_measurement_covariance, scanner_displacement)
        print likelihoods
```

Ln: 140 Col: 20



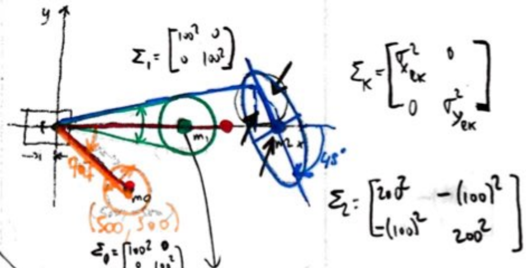

```
Python 2.7.3 |EPD_free 7.3-2 (32-bit)| (default, Apr 12 2012, 11:28:34)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Landmark 0 ----------
 Expected range: 707.106781187 bearing [deg] -45.0
 Covariance of measurement:
[[  5.00000000e+04   0.00000000e+00]
 [  0.00000000e+00   8.85389195e-02]]
Landmark 1 ----------
 Expected range: 1000.0 bearing [deg] 0.0
 Covariance of measurement:
[[  5.00000000e+04   0.00000000e+00]
 [  0.00000000e+00   7.85389195e-02]]
Landmark 2 ----------
 Expected range: 2000.0 bearing [deg] 0.0
 Covariance of measurement:
[[  4.00000000e+04  -5.00000000e+00]
 [ -5.00000000e+00   6.85389195e-02]]
Measurement likelihoods ----------
Likelihoods for measurement close to landmark 0
[0.0023920377034520276, 2.1215588824945e-05, 4.826261771486e-10]
Likelihoods for measurement exactly between landmark 1 and 2
[1.36641638243582276e-07, 0.00020847619108492608, 0.00042908533374704956]
>>>
```

Ln: 26 Col: 4

Q_{t0}

Q_{t1}

Q_{t2}



$\Sigma_1 = \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix}$

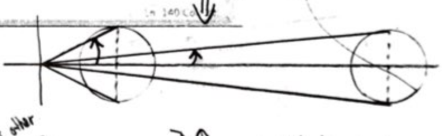
$\Sigma_K = \begin{bmatrix} \sigma_{x_{EK}}^2 & 0 \\ 0 & \sigma_{y_{EK}}^2 \end{bmatrix}$

$\Sigma_2 = \begin{bmatrix} 200^2 & -(100)^2 \\ -(100)^2 & 200^2 \end{bmatrix}$

$\Sigma_0 = \begin{bmatrix} 100^2 & 0 \\ 0 & 100^2 \end{bmatrix}$

Correlation between range (D_{t2}) and bearing angle (ϕ_{t2})

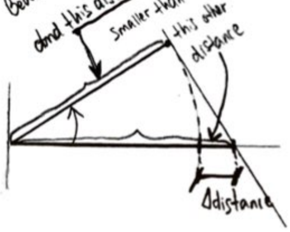
The further the point is the lesser the uncertainty in the bearing angle



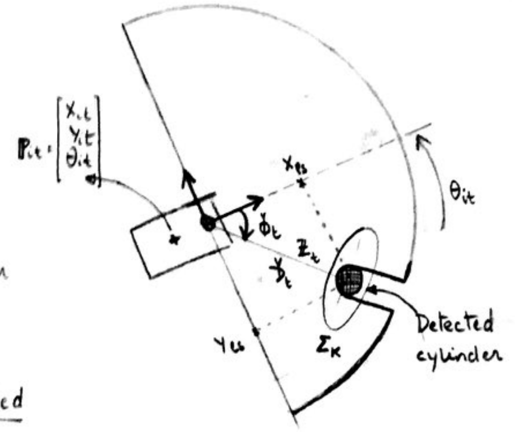
$D_{tK} \uparrow \rightarrow$ uncertainty $\phi_{tK} \downarrow$

If the bearing angle gets larger the distance gets smaller $\rightarrow$ thats why the $-$ in the term $-5'0000 \Rightarrow \phi_t \downarrow$

$\Rightarrow$



(3) Initialize a new landmark



After computing the likelihood for the landmark in the picture the algorithm determines that this landmark is a new landmark and it should be incorporated into the list of landmarks for the particle the algorithm is working with.

$$\underbrace{(x_{\ell k}, y_{\ell k})}_{m_K} = \underbrace{h^{-1}(\mathbb{P}_{it}, sd, Z_t)}$$

- With $\mathbb{P}_{it}$ and sd compute the scanner's pose
- With Z_t compute (x_{sk}, y_{sk}), i.e, the landmark's coordinates in the scanner's (sensor) reference frame.
- Invoke legoLogfile.scanner_to_world (scanner_pose, (x_{sk}, y_{sk}))

$$H_{tk} = \frac{\partial h}{\partial m_K}\Big|_{(x_{\ell k}, y_{\ell k})} = \begin{bmatrix} \frac{\partial D_{tk}}{\partial x_{\ell k}} & \frac{\partial D_{tk}}{\partial y_{\ell k}} \\ \frac{\partial \phi_{tk}}{\partial x_{\ell k}} & \frac{\partial \phi_{tk}}{\partial y_{\ell k}} \end{bmatrix}$$

translates (propagates) the uncertainty in the landmark's position, Σ_K, into an uncertainty in the expected measurement, Q_{tK}.

Now, the situation is different. The robot takes a scan and the algorithm makes an observation in the scan (detects a cylinder), i.e, $Z_t = \begin{pmatrix} \check{D}_t \\ \check{\phi}_t \end{pmatrix}$. The uncertainty in this observation, Q_Z, translates into an uncertainty (error) in the landmark's position, Σ_K.

So we need the jacobian of $h_t^{-1}(.)$, i.e, the inverse of $H_{tK} \Rightarrow H_{tK}^{-1}$

$$\Sigma_K = H_{tk}^{-1} \cdot \underbrace{Q_Z}_{\begin{bmatrix} \sigma^2_{\check{D}t} & 0 \\ 0 & \sigma^2_{\check{\phi}t} \end{bmatrix}} \cdot \left(H_{tk}^{-1}\right)^T \left[= \begin{pmatrix} V_{\parallel} & V_{\perp} \end{pmatrix} \cdot \begin{bmatrix} \lambda_{\parallel}^2 & 0 \\ 0 & \lambda_{\perp}^2 \end{bmatrix} \cdot \begin{pmatrix} V_{\parallel} & V_{\perp} \end{pmatrix}^T \right]$$

$\alpha_{\parallel} = \mathrm{atan}\left(\frac{V_{\parallel y}}{V_{\parallel x}}\right)$

$\alpha_{\perp} = \alpha_{\parallel} \pm 90°$ or $= \mathrm{atan}\left(\frac{V_{\perp y}}{V_{\perp x}}\right)$

$V_{\parallel}$: Eigenvector showing the direction of the ellipse axis parallel to the scan ray

$V_{\perp}$: EigenVector showing the direction of the ellipse axis perpendicular to the scan ray

$\lambda_{\parallel}$: Semilength of the ellipse axis parallel to the scan ray.

$\lambda_{\perp}$: Semilength of the ellipse axis perpendicular to the scan ray.

Previously we had the covariance of the observation Q_Z and the covariance of the landmark, Σ_K. We computed H_{tK} and with those terms we computed the covariance of the expected measurement, Q_{tK}

$$Q_{tK} = H_{tK} \Sigma_K H_{tK}^T + Q_Z$$

Now, we have only the covariance of the observation Q_Z. We compute H_{tK}^{-1} and then:

$$\Sigma_K = H_{tK}^{-1} Q_Z (H_{tK}^{-1})^T$$

That's the only term we have in this situation.

Exercises of adding new landmarks

$$Z_t^{(1)} = \begin{bmatrix} \check{D}_t \\ \check{\Phi}_t \end{bmatrix} = \begin{bmatrix} 1000 \\ 0^\circ \end{bmatrix} \xRightarrow{h^{-1}(\cdot)} \text{Add a new landmark at } (1000, 0)$$

$$\Sigma_1 = \begin{bmatrix} 200^2 & 0 \\ 0 & 261.8^2 \end{bmatrix} = \underset{V_\parallel \quad V_\perp}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} \underset{\lambda_\parallel \quad \lambda_\perp}{\begin{bmatrix} 200^2 & 0 \\ 0 & 261.8^2 \end{bmatrix}} (V_\parallel V_\perp)^T$$

$\alpha_\parallel = \text{atan}\left(\frac{V_{\parallel y}}{V_{\parallel x}}\right) = \text{atan}\left(\frac{0}{1}\right) = 0^\circ \rightarrow \lambda_\parallel = 200$

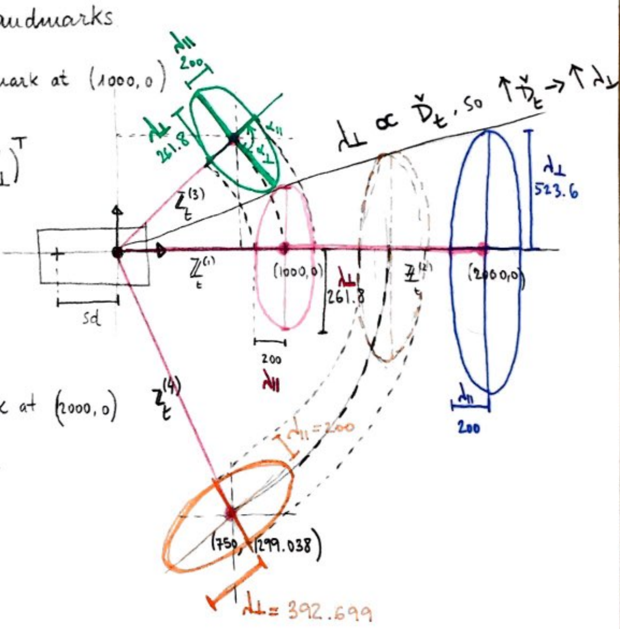
$\alpha_\perp = \text{atan}\left(\frac{V_{\perp y}}{V_{\perp x}}\right) = \text{atan}\left(\frac{1}{0}\right) = 90^\circ \rightarrow \lambda_\perp = 261.8$

$$Z_t^{(2)} = \begin{bmatrix} \check{D}_t \\ \check{\Phi}_t \end{bmatrix} = \begin{bmatrix} 2000 \\ 0 \end{bmatrix} \xRightarrow{h^{-1}(\cdot)} \text{Add a new landmark at } (2000, 0)$$

$$\Sigma_2 = \begin{bmatrix} 200^2 & 0 \\ 0 & 523.6^2 \end{bmatrix} = \underset{V_\parallel \quad V_\perp}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} \underset{\lambda_\parallel \quad \lambda_\perp}{\begin{bmatrix} 200^2 & 0 \\ 0 & 523.6^2 \end{bmatrix}} (V_\parallel V_\perp)^T$$

$\alpha_\parallel = \text{atan}\left(\frac{V_{\parallel y}}{V_{\parallel x}}\right) = \text{atan}\left(\frac{0}{1}\right) = 0^\circ \rightarrow \lambda_\parallel = 200$

$\alpha_\perp = \text{atan}\left(\frac{V_{\perp y}}{V_{\perp x}}\right) = \text{atan}\left(\frac{1}{0}\right) = 90^\circ \rightarrow \lambda_\perp = 523.6$



$$Z_t = \begin{bmatrix} \check{D}_t \\ \check{\Phi}_t \end{bmatrix} = \begin{bmatrix} 2000 \\ 45^\circ \end{bmatrix} \xRightarrow{h^{-1}(\cdot)} \text{Add a new landmark at } (\overset{707'107}{1000\sqrt{2}}, \overset{707'107}{1000\sqrt{2}})$$

$$\Sigma_3 = \begin{bmatrix} 54269.46 & -14269.46 \\ -14269.46 & 54269.46 \end{bmatrix} = \begin{array}{c} V_\parallel \quad V_\perp \\ \begin{bmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{bmatrix} \end{array} \begin{bmatrix} (200)^2 & 0 \\ 0 & (261.8)^2 \end{bmatrix} (V_\parallel V_\perp)^T$$

$\alpha_\parallel = \text{atan}\left(\frac{0.707}{0.707}\right) = 45^\circ \longrightarrow \lambda_\parallel = 200$

$\alpha_\perp = \text{atan}\left(\frac{V_{\perp y}}{V_{\perp x}}\right) = \text{atan}\left(\frac{-0.707}{0.707}\right) = -45^\circ \longrightarrow \lambda_\perp = 261.8$

$$\mathbb{Z}_t^{(4)} = \begin{bmatrix} \check{D}_t \\ \check{\Phi}_t \end{bmatrix} = \begin{bmatrix} 1500 \\ 60 \end{bmatrix} \overset{h^{-1}(.)}{\Longrightarrow} \text{Add a new landmark at } (750, -1299.038)$$

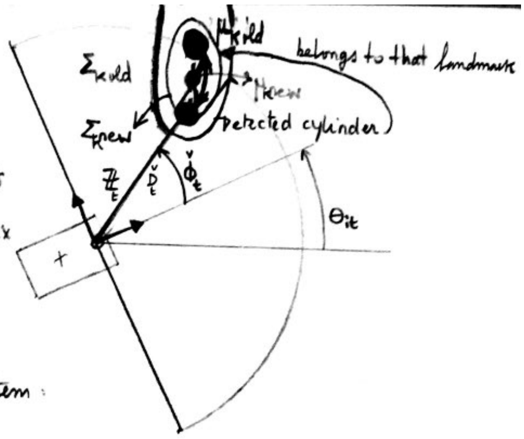
$$\Sigma_4 = \begin{bmatrix} 125659.427 & +49455.493 \\ +49455.493 & 68553.142 \end{bmatrix} = \begin{array}{c} \begin{bmatrix} -0.5 & 0.866 \\ 0.866 & 0.5 \end{bmatrix} \\ \begin{array}{cc} V_{\parallel} & V_{\perp} \end{array} \end{array} \cdot \begin{bmatrix} (200)^2 & 0 \\ 0 & (392.699)^2 \end{bmatrix} \left(V_{\parallel}\ V_{\perp}\right)^T$$

$$\alpha_{\parallel} = \operatorname{atan}\left(\frac{V_{\parallel y}}{V_{\parallel x}}\right) = \operatorname{atan}\left(\frac{0.866}{-0'5}\right) = -60^{\circ} \longrightarrow \lambda_{\parallel} = 200$$

$$\alpha_{\perp} = \operatorname{atan}\left(\frac{V_{\perp y}}{V_{\perp x}}\right) = \operatorname{atan}\left(\frac{0'5}{0.866}\right) = 30^{\circ} \longrightarrow \lambda_{\perp} = 392{,}699$$

$$\boxed{\begin{array}{c} \lambda_{\perp} \propto \check{D}_t \\ \text{Therefore,} \\ \check{D}_t \uparrow \Longrightarrow \lambda_{\perp} \uparrow \end{array}}$$

(c) Landmark Update



As a result of this update process, the landmark's position moves a bit and the covariance matrix get smaller.

Each landmark has an EKF to keep track ~~them~~.

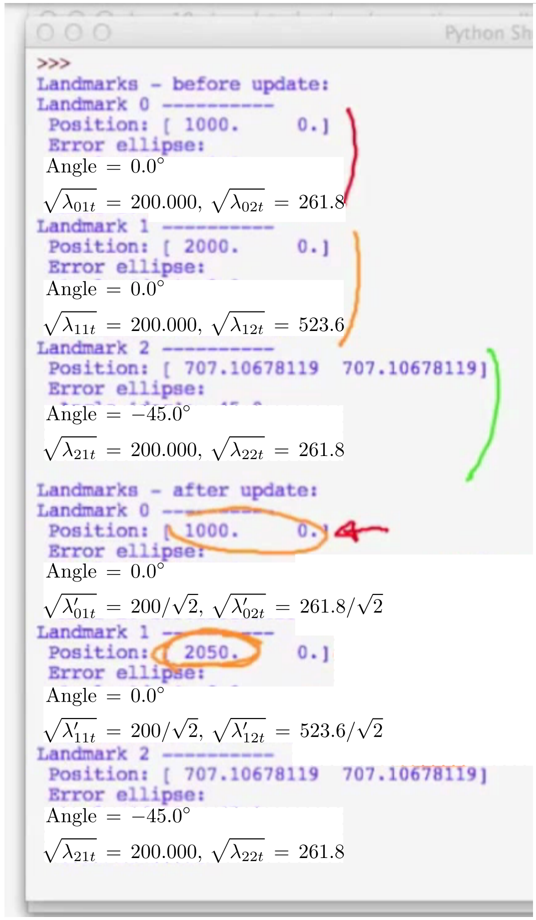
$$K_t = \Sigma_{k\,old} \cdot H^T_{tk} \cdot \underbrace{\Big(\underbrace{H_{tk}\, \Sigma_{k\,old}\, H^T_{tk}}_{\text{Uncertainty of our landmark propagated through our observation}} + \underbrace{Q_Z}_{\text{Observation Uncertainty}} \Big)}_{Q_{tk}}^{-1}$$

$$K_t = \Sigma_{k\,old} \cdot H^T_{tk} \cdot Q^{-1}_{tk}$$

$$\mu_{k\,new} = \mu_{k\,old} + K_t \underbrace{\left(Z_t - h(\mathbb{P}_{it}, k) \right)}_{\text{Innovation}}$$

We are going to use the coordinates of the old landmark stored at index k in the list of landmarks for particle i.

$$\Sigma_{k\,new} = (I - K_t \cdot H_{tk})\, \Sigma_{k\,old}$$



Python Sh

```
>>>
Landmarks - before update:
Landmark 0 ----------
 Position: [ 1000.      0.]
 Error ellipse:
```
Angle $= 0.0°$

$\sqrt{\lambda_{01t}} = 200.000,\ \sqrt{\lambda_{02t}} = 261.8$

```
Landmark 1 ----------
 Position: [ 2000.      0.]
 Error ellipse:
```
Angle $= 0.0°$

$\sqrt{\lambda_{11t}} = 200.000,\ \sqrt{\lambda_{12t}} = 523.6$

```
Landmark 2 ----------
 Position: [ 707.10678119   707.10678119]
 Error ellipse:
```
Angle $= -45.0°$

$\sqrt{\lambda_{21t}} = 200.000,\ \sqrt{\lambda_{22t}} = 261.8$

```
Landmarks - after update:
Landmark 0 ----------
 Position: [ 1000.      0.]
 Error ellipse:
```
Angle $= 0.0°$

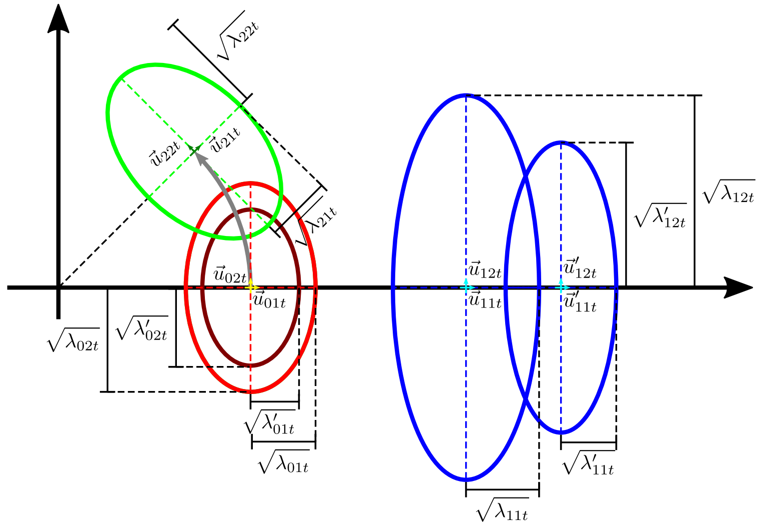
$\sqrt{\lambda'_{01t}} = 200/\sqrt{2},\ \sqrt{\lambda'_{02t}} = 261.8/\sqrt{2}$

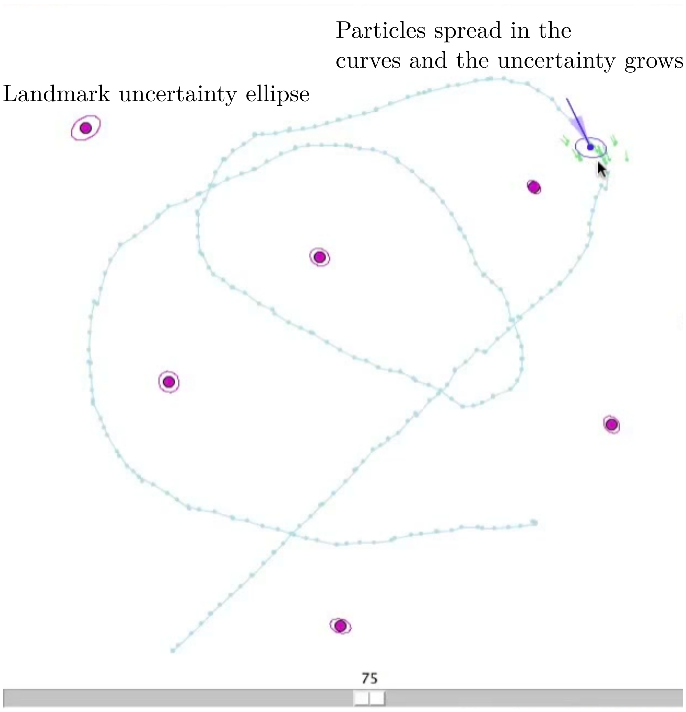
```
Landmark 1 ----------
 Position: [ 2050.      0.]
 Error ellipse:
```
Angle $= 0.0°$

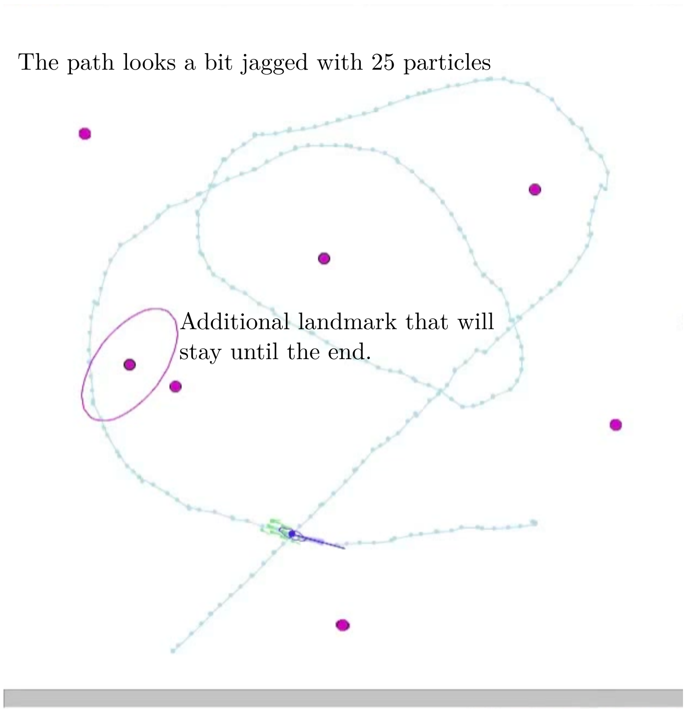
$\sqrt{\lambda'_{11t}} = 200/\sqrt{2},\ \sqrt{\lambda'_{12t}} = 523.6/\sqrt{2}$

```
Landmark 2 ----------
 Position: [ 707.10678119   707.10678119]
 Error ellipse:
```
Angle $= -45.0°$

$\sqrt{\lambda_{21t}} = 200.000,\ \sqrt{\lambda_{22t}} = 261.8$







That additional landmark appears in a particle at some point of the execution of the algorithm and stays till the end of its execution. This happens because the particle that contains the estimated position and covariance matrix of that ellipse is duplicated during the resampling process, probably because it's very close to the mean state. Posterior resampling processes duplicate that landmark over and over again until that landmark is presented in every particle. None of the measurements taken by the laser scanner is assigned to that landmark anymore. Therefore, it's necessary to find a way to get rid of those ghosts landmarks that appear at some point in time.