PNT Fusion Algorithms Part 2: Particle Filters, Factor Graphs & SLAM

Dr. Maarten Uijt de Haag and Mats Martens, M.Sc.
Institute of Aeronautics and Astronautics
Flight Guidance and Air Transport
Marchstrasse 12, 10587 Berlin
GERMANY

maarten.uijtdehaag@tu-berlin.de

ABSTRACT

In this paper, we will build on the discussion on the integration algorithms in Part 1 and address estimators that are well-suited for systems that exhibit a high degree of non-linearity in the measurement and/or system (dynamics) equations. Specifically, we will discuss the particle filter and factor graphs. In addition to these two estimation approaches, we will discuss their application in integrated navigation systems as well as simultaneous localization and mapping (SLAM) methods.

1.0 MOTIVATION AND BACKGROUND

Navigation performance is often expressed in terms of its performance parameters: accuracy, integrity, availability, and continuity. Accuracy is defined as the degree of conformance between the estimated or measured position and/or the velocity of a platform at a given time and its true position or velocity. Integrity, on the other hand, is the ability of a navigation system to provide timely warnings to users when the system should not be used for navigation. And, whereas continuity is ability of a system to perform its intended function without interruption during the intended operation if it did so at the beginning of the operation, availability is defined as the ability of the system to provide usable service within the specified coverage area [1].

Depending on the application, the required performance of the navigation capability (i.e., the required navigation performance) may differ. For example, the required navigation performance of a manned or unmanned aircraft during the landing phase will be more stringent than during its cruise phase. Also, within the performance parameters there may be some variation. For example, when using navigation to help following a route while driving a car may require decent accuracy but will have no strong requirements with respect to integrity as the driver of the vehicle takes over that role. However, in case of the autonomous flight of a UAV, the integrity of the solution may have an increased role when used to stay clear of obstacles for example.

Often one source of navigation may not be sufficient to meet all navigation performance parameters associated with the mission or application's operational scenario. A good example is the use of global navigation satellite systems, GNSS (e.g., GPS, Galileo, GLONASS, Beidou), which has widely been considered the enabler of many high precision application (e.g., agriculture, unmanned aerial vehicles, car-navigation, mobile phones, etc.). However, there are limitations to GNSS that have become evident over time as we have increasingly come to rely on it for navigation. In challenging environments such as indoor, subterranean, urban areas, under foliage, etc., for example, measurements from GNSS are often unavailable or only sparsely available. Furthermore, if available, the measurement performance may be deteriorated due to multipath or the lack of direct line-of-sight (i.e., only measurements based on reflected signals are available: non-line-of-sight reception).

If one source of navigation is not sufficient to meet the mission's required navigation performance, either alternative navigation sources must be found such as laser scanners, cameras, beacon-based systems (e.g., pseudolites, UWB), and signals of opportunities, or data from multiple sensors (or navigation aids) must be combined (i.e., integrated or fused) in such a way that the Required Navigation Performance of the intended operation can be met. This is referred to as assured navigation. An example of an integration strategy is illustrated in Figure 1-1. In this figure, a core sensor is identified that is available all the time, but not perfect, e.g., an inertial navigation system (INS) with a large drift error. Its output is integrated with the outputs of other sensors for estimation and correction of the drift and other errors. At the same time, the INS will be there when the other sensors are not available and, therefore, bridge any outages.

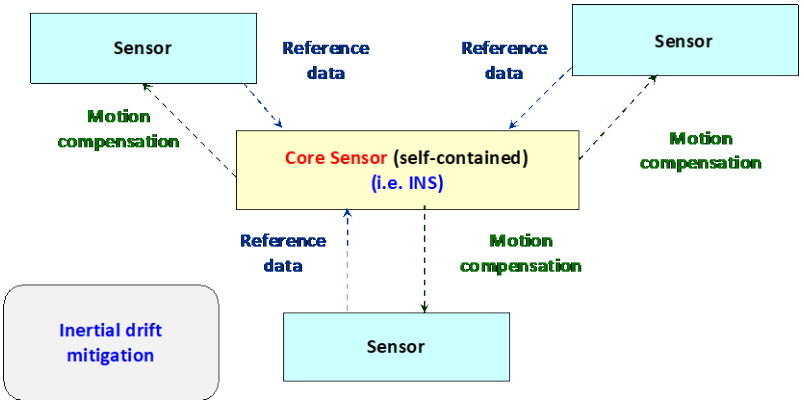

Figure 1-1: Sensor fusion or integration built around a core-sensor that is always available but requires periodic updating or calibration.

Note that all these integration methods can be applied to cooperative systems that exchange either measurement or state estimates and associated covariances either centralized or distributed among the various users. In the latter case, one must take care of incorporating the various participant contributions' covariances and appropriately using these to weigh the individual results following the discussion in part 1.

2.0 DEFINITION OF THE INTEGRATED NAVIGATION PROBLEM

An important first step, besides the definition of the navigation performance requirements, is the identification of the state vector, $\mathbf{x}_k$, at time epoch 'k', i.e., what parameters must be estimated at each time epoch. This state vector may include position, velocity, attitude, errors, etc. When identified, the sensor measurements are related to the state vector through the so-called measurement equation. An example, of a typical measurement equation is given by:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k \tag{1}$$

where $\mathbf{h}(\cdot)$ is a (non-)linear function of the state vector and $\mathbf{v}_k$ is an additive measurement noise vector that is often assumed to be a normally distributed vector with covariance matrix $\mathbf{R}$, or:

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{2}$$

Equation (1) is often expressed probabilistically by the so-called likelihood function; a probability density function (PDF) that expresses what the measurement (vector) looks like given a state vector $\mathbf{x}_k$:

$$p(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{h}(\mathbf{x}_k), \mathbf{R}) = \eta \cdot exp\left\{-\frac{1}{2}[\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k)]^T \mathbf{R}^{-1}[\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k)]\right\} \quad (3)$$

Again Equation (3) assumes a normally distributed random variable. In addition to the measurement equation, we also must define the system or state propagation equation which captures the dynamics of the state vector. A typical example of this equation is:

$$\mathbf{x}_k = \mathbf{g}(\mathbf{x}_{k-1}) + \mathbf{w}_k \quad (4)$$

where $\mathbf{g}(\cdot)$ is a (non-)linear function of the state vector and $\mathbf{w}_k$ is an additive noise vector that represents the uncertainty in the system model and is often assumed to be a normally distributed vector with covariance matrix $\mathbf{Q}$, or:

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \quad (5)$$

Like Equation (3), the state propagation can be expressed probabilistically.

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{g}(\mathbf{x}_{k-1}), \mathbf{Q}) = \eta \cdot exp\left\{-\frac{1}{2}[\mathbf{x}_k - \mathbf{g}(\mathbf{x}_{k-1})]^T \mathbf{Q}^{-1}[\mathbf{x}_k - \mathbf{g}(\mathbf{x}_{k-1})]\right\} \quad (6)$$

In a typical estimation problem, the goal is to determine an estimate of the $\mathbf{x}_k$, $\hat{\mathbf{x}}_k$. In part 1 this was achieved using Kalman filters, or, Extended or Unscented Kalman Filters (EKF, and UKF) in case of non-linear measurement and /or state propagation equations.

Probabilistically, the best estimate can be obtained from the so-called *posterior* distribution: the probability density of $\mathbf{x}_k$ given by a measurement vector $\mathbf{z}_k$:

$$p(\mathbf{x}_k|\mathbf{z}_k) \quad (7)$$

In basic filtering, a cost function is used to obtain the best estimate from the posterior density function. Example cost functions are the squared error, absolute error, and uniform error, corresponding to the mean, median and mode of the posterior, respectively.

The posterior can be related to the likelihood function using Bayes rule, or:

$$p(\mathbf{x}_k|\mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\mathbf{z}_k)} = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{\int_{-\infty}^{\infty} p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k)d\boldsymbol{x}} = \eta \cdot p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k) \quad (8)$$

where $p(\mathbf{x}_k)$ is the marginal PDF of $\mathbf{x}_k$(a.k.a. the *prior* as it represents prior knowledge of the state vector) and $p(\mathbf{z}_k)$ is the marginal PDF of $\mathbf{z}_k$ (a.k.a. the evidence).

In the previous equations or models, we assumed the so-called Markov property, that states that the current measurement vector is only dependent on the current state, and that the current state is only dependent on the previous state, or:

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}) = p(\mathbf{z}_k|\mathbf{x}_k) \quad (9)$$

$$p(\mathbf{x}_k|\mathbf{x}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \quad (10)$$

where $\mathbf{x}_{1:k} = \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k$ represents the previous states.

Rather than filtering, one could also choose to collect all measurements over a time interval and use them to estimate the state estimates for all or some epochs within the time interval. For example, fixed-interval smoothers estimate the state for all epochs on a time interval of size N from $k-N$ to k. Alternatively, fixed-point, and fixed-lag smoothers estimate the state at a fixed point or a fixed delay in the past [1]. Probabilistically, these three smoothers are given by:

Fixed-interval smoother	$p(\mathbf{x}_{k-N:k}\|\mathbf{z}_{k-N:k})$	(11)
Fixed-point smoother	$p(\mathbf{x}_{l}\|\mathbf{z}_{k-N:k})$	(12)
Fixed-lag smoother	$p(\mathbf{x}_{k-M}\|\mathbf{z}_{k-N:k})$	(13)

where $k-N<k-M<k$.

Bayesian filters, such as the Kalman filters and the particle filter discussed in the next section are based on obtaining an expression for the posterior that consists of a prediction and measurement update part, or:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k}) &= \eta \cdot p(\mathbf{z}_k|\mathbf{x}_k,\mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \\ &= \eta \cdot p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \\ &= \eta \cdot p(\boldsymbol{z}_k|\boldsymbol{x}_k)\int p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1},\boldsymbol{x}_{k-1})\,p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})d\boldsymbol{x}_{k-1} \\ &= \eta \cdot p(\boldsymbol{z}_k|\boldsymbol{x}_k)\int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})\,p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1})d\boldsymbol{x}_{k-1} \end{aligned} \tag{14}$$

In the steps show in Equation (14), we make use of the Bayes theorem, the total probability and Markov properties from probability theory. For a typical Bayesian filter, the prediction and measurement update parts follow from Equation (14) as follows:

Prediction	$\overline{bel}(\mathbf{x}_k) = \int p(\mathbf{x}_k\|\mathbf{x}_{k-1})\,p(\mathbf{x}_{k-1}\|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1}$	(15)
Update	$bel(\mathbf{x}_k) = \eta \cdot p(\mathbf{z}_k\|\mathbf{x}_k)\overline{bel}(\mathbf{x}_k)$	(16)

The Kalman filter equation can be derived from Equations (15) and (16), by substituting the PDFs by Normal distributions with the appropriate means and covariances. The posterior density is often extended by including the robot or vehicle's action, $\mathbf{u}_k$, $p(\mathbf{x}_k|\mathbf{z}_k,\mathbf{u}_k)$.

Note that the covariance intersection (CI) method discussed in Part 1and illustrated using a simulation, can also be more generally described using PDFs [4][5]:

$$p(\mathbf{x}_k|\mathbf{z}_A,\mathbf{z}_B) = \frac{p^{\omega}(\mathbf{x}_k|\mathbf{z}_A)p^{1-\omega}(\mathbf{x}_k|\mathbf{z}_B)}{\int p^{\omega}(\mathbf{x}_k|\mathbf{z}_A)p^{1-\omega}(\mathbf{x}_k|\mathbf{z}_B)d\mathbf{x}_k} \tag{17}$$

where $\mathbf{z}_A$ and $\mathbf{z}_B$ are the measurements from user A and B, respectively, and $\omega \in [0,1]$ is the weight factor discussed in Part 1.

3.0 PARTICLE FILTERS

Particle filters (PF) are typically meant for cases where the measurement and/or the system equation are non-linear, resulting in sub-optimal or erroneous results when using Kalman filters.

The PF is based on the principle that, if one would know the posterior, one could draw samples from that PDF, plot a histogram of the resulting samples and calculate the sample mean as an estimate for the actual mean, providing an approximation of the minimum mean square estimate, $\hat{\mathbf{x}}_k = E\{p(\mathbf{x}_k|\mathbf{z}_{1:k})\} = E\{\mathbf{x}_k|\mathbf{z}_{1:k}\}$. This approach is referred to as sequential importance sampling (SIS) [3], or:

$$\hat{\mathbf{x}}_k = E\{\mathbf{x}_k|\mathbf{z}_{1:k}\} = \sum_{i=1}^{M} w_k(\mathbf{x}_k^i)\mathbf{x}_k^i \tag{18}$$

where $w_k(\mathbf{x}_k^i)$ are the relative histogram values, or weights for value $\mathbf{x}_k^i$. The latter are referred to as particles.

Figure 3-1 shows an example of the histogram for 5000 samples drawn from a known posterior distribution.

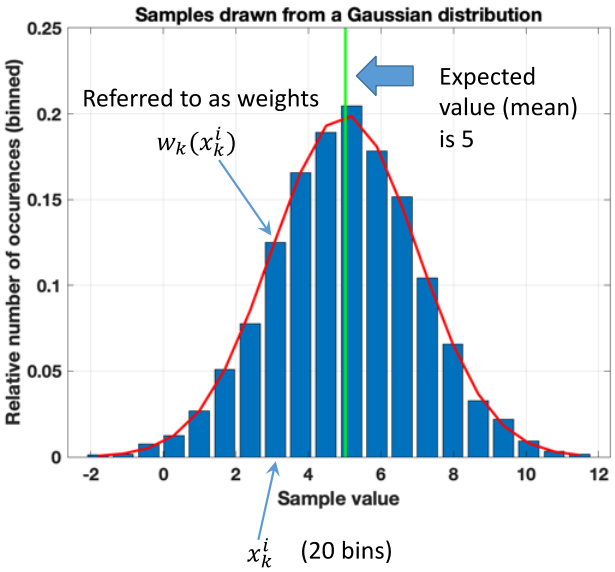

Figure 3-1: Histogram example for drawing samples from an example (Normally distributed) posterior distribution.

Since the posterior density is often not explicitly available it is impossible to sample directly from that distribution. Instead, we can sample from a known, easy-to-sample, proposal or importance distribution 'q', and determine the 'histogram' weights as follows:

$$w_k = w_{k-1}\frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i,\mathbf{z}_{1:k},)} \tag{19}$$

where q' is referred to as the proposal distribution and its choice is a critical design issue for a successful particle filter. The derivation of Equation (19) can be found in [3]. Various requirements exist for a good proposal distribution including enough support for the true posterior distribution (heavy-tailed distributions are preferred) or the inclusion of recent observations.

Table 3-1: Sequence Importance Sampling (SIS) - Algorithm

1	**Algorithm** $\{\mathbf{x}_k^i, w_k^i\}_{i=1:N} = \mathbf{SIS}\left(\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1:N}, \mathbf{z}_k\right)$	

2	$for\ i = 1, \ldots, N$	
3	$sample\ \mathbf{x}_k^i \sim q(\mathbf{x}_k^i \vert \mathbf{x}_{k-1}^i, \mathbf{z}_{1:k},)$	**Obtain the particles**
4	$\tilde{w}_k^i = w_{k-1}^i \frac{p(\mathbf{z}_k \vert \mathbf{x}_k^i) p(\mathbf{x}_k^i \vert \mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i \vert \mathbf{x}_{k-1}^i, \mathbf{z}_{1:k},)}$	**Compute Importance weights**
5	$endfor$	
6	$\eta = \sum_{i=1}^{N} \tilde{w}_k^i$	**Compute normalization factor**
6	$for\ i = 1, \ldots, N$	
8	$w_k^i = \tilde{w}_k^i / \eta$	**Normalize the weights**
9	$endfor$	

One example for the proposal distribution is state propagation density, or $q(\mathbf{x}_k^i \vert \mathbf{x}_{k-1}^i, \mathbf{z}_{1:k},) = p(\mathbf{x}_k^i \vert \mathbf{x}_{k-1}^i)$. Substituting this proposal distribution into Equation (19) yields:

$$w_k = w_{k-1} p(\mathbf{z}_k \vert \mathbf{x}_k^i) \tag{20}$$

Another popular PF filter is the so-called bootstrap filter or Sampling Importance Resampling (SIR) that simplifies the weight selection to:

$$w_k = p(\mathbf{z}_k \vert \mathbf{x}_k^i) \tag{21}$$

Once executing the SIS and SIR, it is important to realize that the variance of the importance weights increases stochastically over time; Or, in other words, after a certain number of recursive steps most particles have a negligible weight. This effect is illustrated when plotting the cumulative sum of the weights over time (see Figure 3-2).

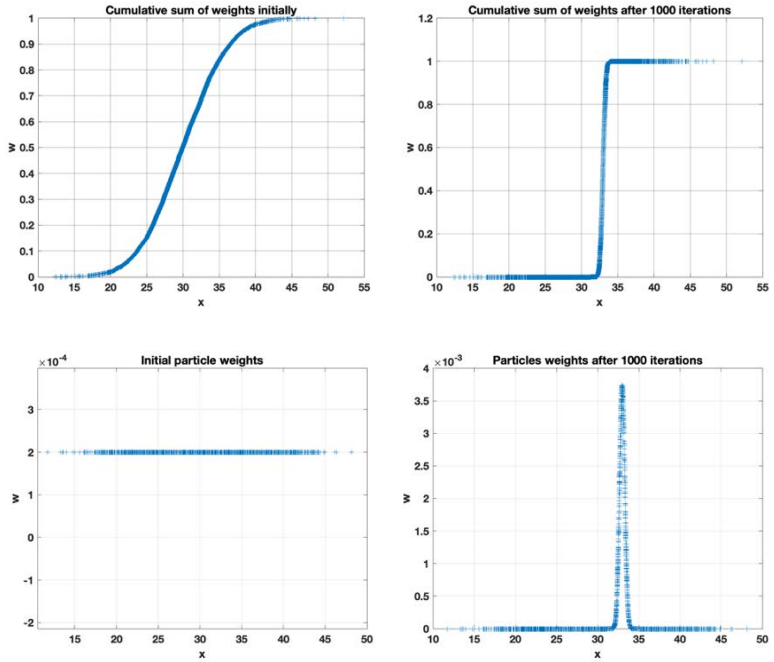

Figure 3-2: Cumulative sum of weights initially (top-left) and after 1000 iterations (top-right) and the actual weight values initially (bottom-left) and after 100 iterations (bottom-right).

The solution to this problem is to re-sample the particles. In the re-sampling process, we keep and multiply the particles with high importance weights and discard the particles with low importance weights. More details on this process can be found in [3]. The resulting cumulative sum and weights after 100 iterations are shown in Figure 3-3. A typical measure to indicate how many particles are effectively participating in "main" part of the PDF is:

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}\left(w_t^i\right)^2} \tag{22}$$

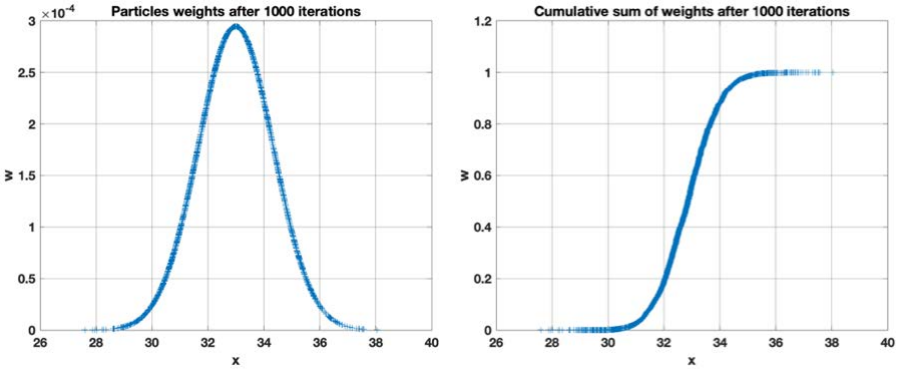

Figure 3-3: Cumulative sum of weights and weights when using re-sampling after 1000 iterations.

Consolidating the SIR and the resampling steps, leads to the PF algorithm in Table 3-2.

Table 3-2: Particle Filter - Algorithm

1	**Algorithm** $\{\mathbf{x}_k^i, w_k^i\}_{i=1:N} = PF\left(\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1:N}, \mathbf{z}_k\right)$	
2	$\{\mathbf{x}_k^i, w_k^i\}_{i=1:N} = \mathbf{SIR}\left(\{\mathbf{x}_{k-1}^i, w_{k-1}^i\}_{i=1:N}, \mathbf{z}_k\right)$	Perform SIR filtering
3	$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}\left(w_t^i\right)^2}$	Compute Effective Number of Particles
4	$if\ \widehat{N}_{eff} < N_{thr}$	
5	. $\{\mathbf{x}_k^{j*}, w_k^j\}_{j=1:N} = \mathbf{Resample}\left(\{\mathbf{x}_k^i, w_k^i\}_{i=1:N}\right)$	Resample
6	$endif$	
6	$\hat{\mathbf{x}}_k = \sum_{j=1}^{N} w_k^j \mathbf{x}_k^j$	State Estimate
8	$\mathbf{P}_k = \sum_{j=1}^{N} w_k^j \left(\mathbf{x}_k^j - \hat{\mathbf{x}}_k\right)\left(\mathbf{x}_k^j - \hat{\mathbf{x}}_k\right)^T$	Covariance Estimate

The PF requires many more computations than a typical EKF or even an UKF. Therefore, it is important to assess beforehand if the Gaussian assumption is not valid or if the non-linearities in the measurement and system equations lead to a posterior that is likely not Gaussian, justifying the selection of a PF to solve the problem at hand.

An example of a problem that is non-linear, but not enough so to warrant a PF, is the local ranging problem based on two two-way ranging (TWR) sources at locations (-50km, 50km) and (50km, 50km) with a range

noise of about 50m 1-σ and a constant velocity (CV) system model with a process noise equal to 10m/s. For the particle filter 5000 particles were chosen. Figure 3-4 and Figure 3-5 show the results for the EKF and PF, respectively, and one can clearly observe that the performance is very similar and that, thus, an EKF implementation should be sufficient.

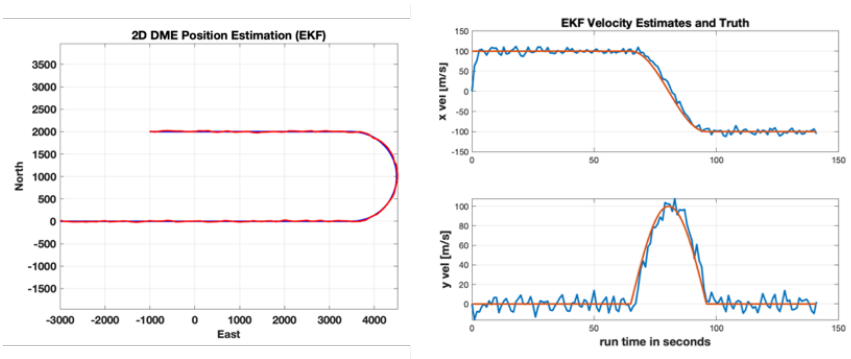

Figure 3-4: Terrestrial ranging: range-based position (left) and velocity (right) estimates using EKF.

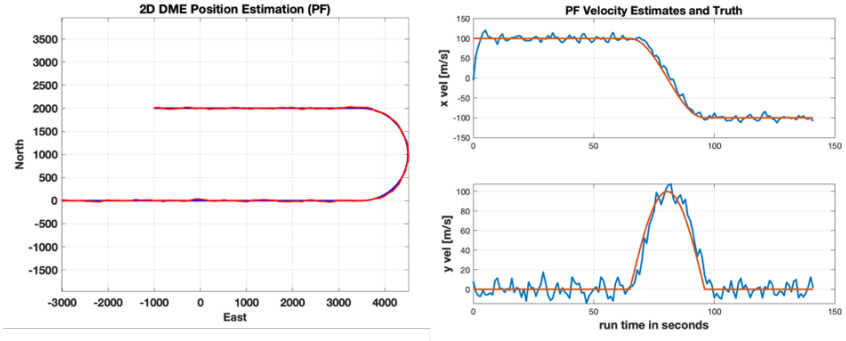

Figure 3-5: Terrestrial ranging: range-based position (left) and velocity (right) estimates using PF.

Extended Kalman Filter

```
#   ------------------------------------
#   "Predict"
#   ------------------------------------

#   Project the state ahead
x_pre = PHI @ x_est[:, [kk]]

#   Project the error covariance ahead
P_pre = PHI @ P @ PHI.T + Q
```

Mean: $\hat{\mathbf{x}}_k^- = \mathbf{g}(\hat{\mathbf{x}}_{k-1})$ Covariance: $\mathbf{P}_k^- = \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{Q}$

```
#   ------------------------------------
#   "Update"
#   ------------------------------------

for jj in range(N):

    # Compute the nominal range
    z_est[jj, 0] = np.linalg.norm(x_pre[[0, 2]] - x_dme[:, [jj]])

    # Compute the H-matrix (measurement matrix)
    H[jj, 0] = (x_pre[0] - x_beacon[0, jj]).T / z_est[jj]
    H[jj, 2] = (x_pre[2] - x_beacon[1, jj]).T / z_est[jj]

    #   Compute the Kalman gain
    K = P_pre @ H.T @ np.linalg.inv(H @ P_pre @ H.T + R)

    #   Update estimate with measurement
    x_est[:, [kk]] = x_pre + K @ (z - z_est)

    #   Save the innovations/residuals
    resi[:, [kk]] = (z - z_est)

    #   Update the error covariance
    P = (I - K @ H) @ P_pre
```

Particle Filter

```
#   ------------------------------------
#   "Predict"
#   ------------------------------------

for ii in range(N):

    #   Draw sample from the transition density
    q = PHI @ xp[:,[ii]]
    xp[:,[ii]] = np.random.multivariate_normal(…
```

$\mathbf{x}_k^i \sim p(\mathbf{x}_k^i | \mathbf{x}_{k-1}^i) = N(\mathbf{g}(\mathbf{x}_{k-1}^i), \mathbf{Q})$

```
#   ------------------------------------
#   "Update"
#   ------------------------------------

for jj in range(N):

  …
  …

  #   Compute the measurement with the particle (h(x)'
  hx = np.zeros((2,1))
  hx[0] = np.sqrt( (xp[0,ii]-x_beacon[0,0])**2 + (xp[2,ii]-x_beacon[0,1])**2)
  hx[1] = np.sqrt( (xp[0,ii]-x_beacon[1,0])**2 + (xp[2,ii]-x_beacon[1,1])**2)

    #   Compute the Importance weight
    p = multivariate_normal.pdf(z.T, mean=hx.flatten(), cov=R)
    w[0,ii] = p
```

$\tilde{w}_k^i = p(\mathbf{z}_k | \mathbf{x}_k^i) = N(\mathbf{h}(\mathbf{x}_k^i), \mathbf{R})$

Figure 3-6: EKF versus PF code structure for the ranging beacon example.

An example of how the PF differs from the EKF is shown below for the above example using a fragment of Python code that was used for its implementation. In the EKF (left side of Figure 3-6), the probability density function of the prediction and updates are represented by their mean and covariance (matrix) of normal distributions $N(\hat{\mathbf{x}}_k^-, \mathbf{P}_k^-)$ and $N(\hat{\mathbf{x}}_k, \mathbf{P}_k)$, respectively. In case of the PF, however, a set of N possible estimates (particles) with associated weights, represent the whole PDF (no longer assumed to be Normal or Gaussian). In case of the bootstrap PF you assume a Normally-distributed input to obtain the particles of the prediction and the filter's update stages.

The PF is better suited for problems that are either strongly non-linear or do have underlying PDFs that are not Normal in nature. A good example is the terrain referenced navigation (TRN) problem shown in Figure 3-7.

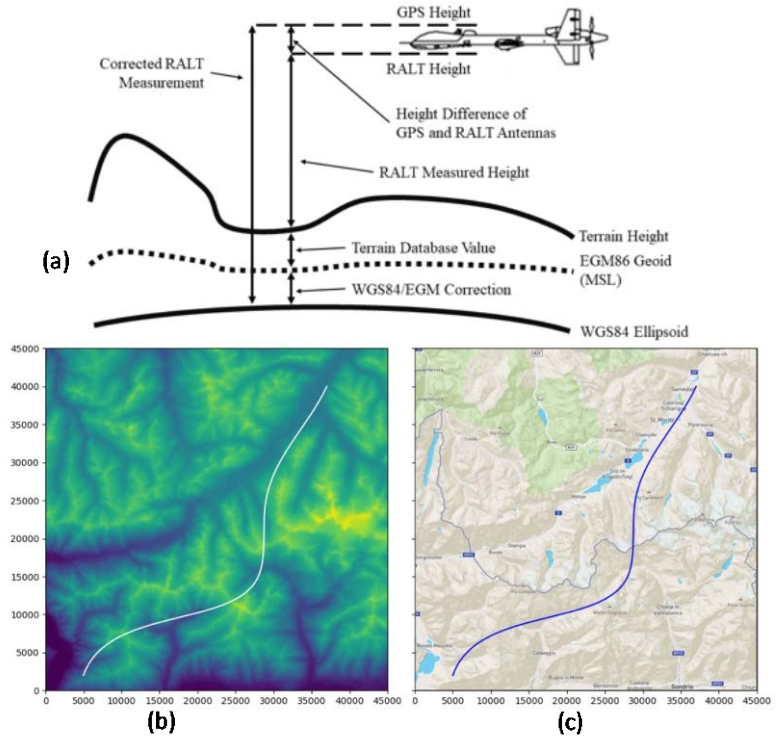

Figure 3-7: (a) Basic terrain referenced navigation concept using terrain elevation database and radar altimeter, (b) digital elevation model (DEM) for Samedan, Switzerland with overlaid reference trajectory, (c) regular map of Samedan with overlaid trajectory.

In the implemented TRN, the height above ground level (AGL) measurements from the aircraft's radar altimeter are used in combination with a Digital Elevation Model or DEM (i.e., a terrain database) to estimate the user position. The measurement equation is given by:

$$z_{RAD} = h_k - h_{DEM}(\mathbf{x}_k) + \mathbf{v}_k \tag{23}$$

where h_k is the z-component of $\mathbf{x}_k$, $h_{DEM}(\mathbf{x}_k)$ is the terrain height at location $\mathbf{x}_k$ obtained from a database lookup in the DEM. The error $\mathbf{v}_k$ consists of the radar altimeter measurement noise and interpolation errors.

In the example implementation, inertial navigation system information is incorporated into the filter by including it as a forcing function in the system equation (i.e., the proposal distribution) or $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{u}_k)$. The results of this PF are shown in Figure 3-8. Figure 3-8(a) shows the initial distribution of the particles over the search space and a resulting estimate that is way off. Over time, however, many particles further away from ownship disappear and the estimate slowly converges to the true location; see Figure 3-8(b) and (c). Then

the filter slowly converges with some far away particles still in existence; Figure 3-8(d) and (e). However, the filter finally converges with all particles concentrated around the true ownship location.

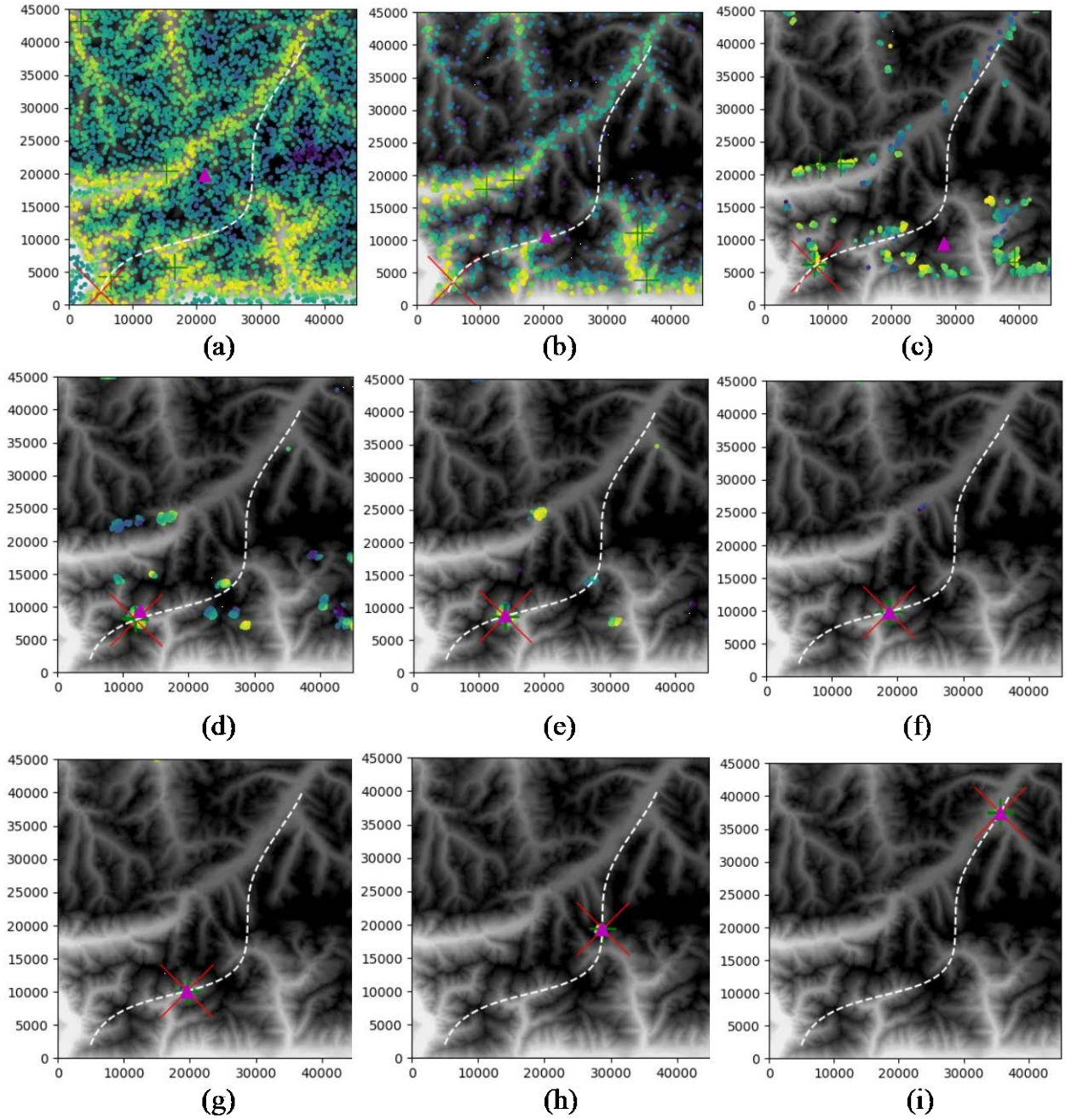

Figure 3-8: (a) Unknown initial position; particles are evenly distributed over the DEM search space, (b) - (c) large offsets in the position estimates (magenta triangles) due to the presence of many particles far away from ownship, (d) – (f) "far away" particles become sparse, and estimate converges to true location, (g) – (f) convergence maintained throughout the remainder of the flight.

The estimation error and its covariance are shown in Figure 3-9. The convergence of the particles and, thus, the estimate can clearly be observed in the results: after about 50 seconds the results are converged to an error of around 25m.

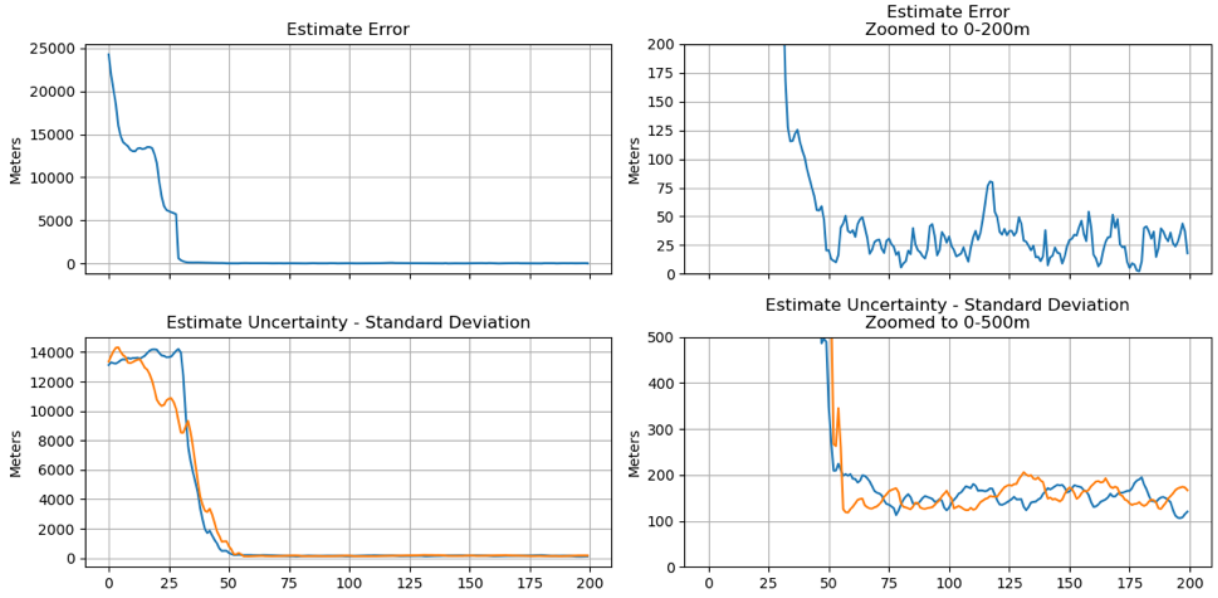

Figure 3-9: TRN PF estimation error and standard deviation.

The above example includes the PF performance for a single platform. In case multiple platforms observe the same state, multiple state estimates can, again, be fused using methods such as CI. An example of this method is discussed in [6].

4.0 FACTOR GRAPH-BASED ESTIMATION METHODS

An alternative to sequential estimation methods such as the EKF and the PF discussed in the previous section, is the use of a batch of data (across sensors and/or across time) to obtain the best estimate of the current state or sequence of states using the extended posteriors in Equations (11), (12), and (13). The popularity of these factor graph-based methods is, in part, based on the availability of a large variety of using non-linear least squares solver tools such as g2o [7], Ceres [8], GTSAM [9], SymForce [10] and the various tools available in Matlab.

Using Bayes theorem and the earlier mentioned Markov property, the posterior for a fixed-interval smoother of Equation (11) can be re-written as:

$$p(\mathbf{x}_{0:k}|\mathbf{z}_{0:k}) = \eta p(\mathbf{x}_0)\prod_k p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{z}_k|\mathbf{x}_k) \tag{24}$$

where $p(\mathbf{z}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ are given by Equations (3) and (6), respectively, assuming they are Normally distributed. $p(\mathbf{x}_0)$ is the prior of the state vector. Substituting Equations (3) and (6) into (24) and taking the negative log of the posterior yields:

$$C + \mathbf{x}_0\mathbf{P}_0^{-1}\mathbf{x}_0^T + \frac{1}{2}\sum_k [\mathbf{x}_k - \mathbf{g}(\mathbf{x}_{k-1})]^T\mathbf{Q}^{-1}[\mathbf{x}_k - \mathbf{g}(\mathbf{x}_{k-1})\,] + \frac{1}{2}\sum_k [\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k)]^T\mathbf{R}^{-1}[\mathbf{z}_k - \mathbf{h}(\mathbf{x}_k)] \tag{25}$$

$$\Rightarrow J(\mathbf{x}_{1:k}) = C + \mathbf{x}_0\mathbf{P}_0^{-1}\mathbf{x}_0^T + \frac{1}{2}\sum_k F_g(\mathbf{x}_k, \mathbf{x}_{k-1}) + \frac{1}{2}\sum_k F_h(\mathbf{x}_k, \mathbf{z}_k) \tag{26}$$

where $F_g(\mathbf{x}_k, \mathbf{x}_{k-1})$ and $F_h(\mathbf{x}_k, \mathbf{h}_k)$ are referred to as the factors. This expression is often visualized as a graph where the factors represent constraints imposed by the dynamics model and the measurements. An example

of such a factor graph is shown in Figure 4-1. These constraints tie user location at different times to one another and to the (in this example) beacon locations.

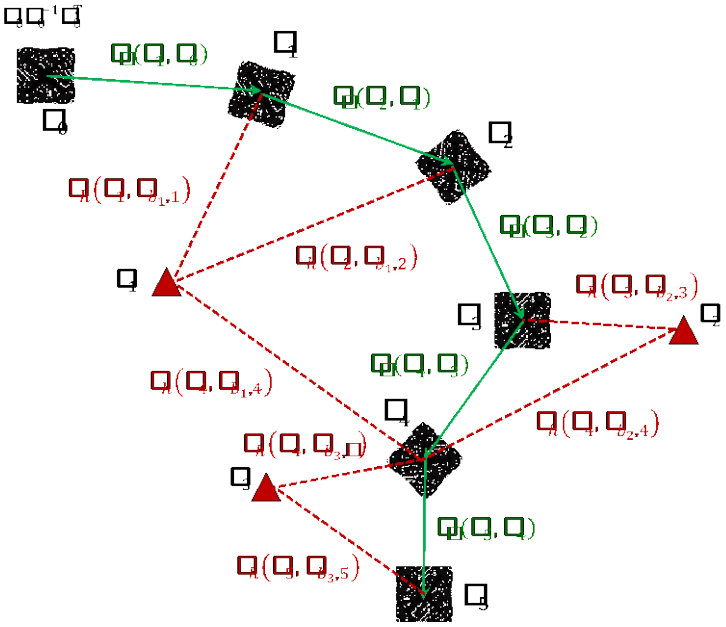

Figure 4-1: Factor graph: the various factors act as constraints imposed by measurements and knowledge of dynamics.

The factor-based estimation problem can now be formulized as an optimization problem, also known as factor graph optimization (FGO), that tries to find the sequence of user locations:

$$\hat{\mathbf{x}}_{1:k} = arg\min_{\mathbf{x}_{1:k}} J(\mathbf{x}_{1:k}) \tag{27}$$

In essence, the optimization tries to figure out what sequence of states, $\hat{\mathbf{x}}_{1:k}$, explains the obtained measurements the best given an underlying model of the vehicle's motion. The solution to equation (27) can be obtained using a variety of approaches from the non-linear least squares solver literature such as the Gauss-Newton and Levenberg-Marquardt (LM) methods. Specifically, for Simultaneous Localization and Mapping (SLAM) and Bundle Adjustment (BA), various better performing and efficiently implemented approaches to solving the above non-linear least squares problem have been proposed like, for example, g2o in [7] and iSAM in [9].

The results for a 3D simulation are shown in Figure 4-2 and Figure 4-3. This example simulates a 3D UAV trajectory consisting of 5 locations. During the flight, the UAV makes range measurements to 6 ranging beacons, resulting in 6 factors for each of the 5 locations. The ranging factor is given by:

$$F_{i,b_l} = \{\rho_{i,b_l} - \|\mathbf{r}_{b_l} - \mathbf{r}_{n,i}\|\}/\sigma_{i,b_l} \tag{28}$$

where ρ_{i,b_l} is the range measurement from UAV 'i' to beacon 'l', $\mathbf{r}_{b_l}$ is the location of beacon 'l', $\mathbf{r}_{n,i}$ is the location of UAV 'i', and σ_{i,b_l} is the nominal standard deviation of the ranging measurement. The standard deviation σ_{i,b_l} may be different for each UAV position as it can depend on the individual ranging radio performance or the distance from the user to the beacon. In this example, the ranging standard deviation ranges from 5cm to 20 cm based on the actual performance of Ultra-wideband (UWB) range radios used by the authors in their research UAV platforms.

The initial positions $\mathbf{x}_0$ through $\mathbf{x}_5$ are set to the origin of the local coordinate frame as shown in Figure 4-2 on the left side. Figure 4-2 (right) and Figure 4-3 (left and right) show the results after iteration 6, 9 and 13, respectively. After 13 iterations, the convergence criterion has been met and the trajectory estimate $\hat{\mathbf{x}}_{0:5}$ finalized.

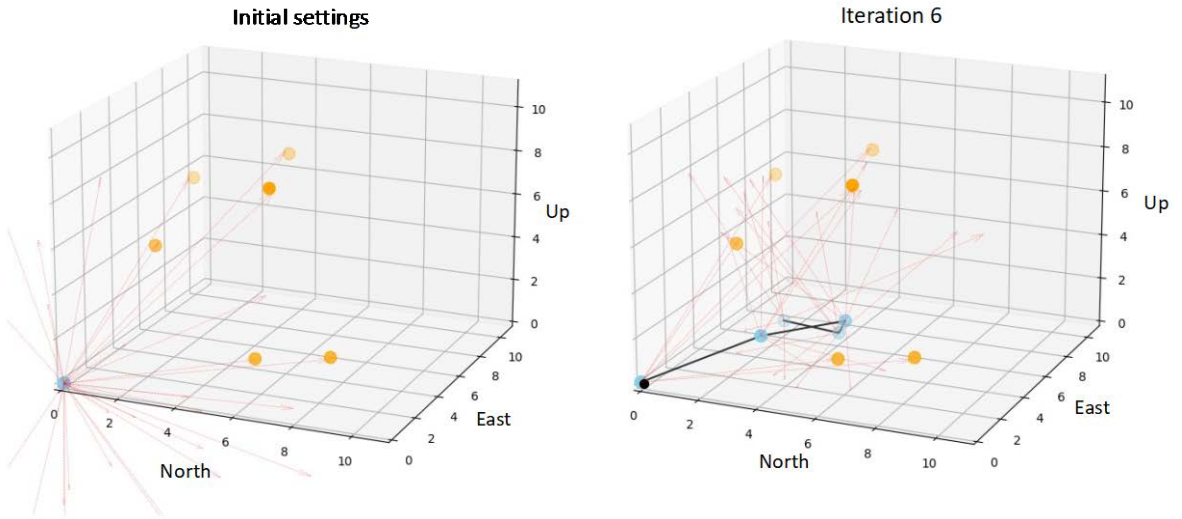

Figure 4-2: Beacon-based navigation results: (left) initial values for the trajectory; (right) trajectory estimate for iteration 6.

The ranging factor in Equation (32), could also be modified to reflect the Global Navigation Satellite System (GNSS) and used to compute GNSS position and trajectories.

$$F_{i,g_l} = \{pr_{i,g_l} - \|\mathbf{r}_{g_l} - \mathbf{r}_{n,i}\| - \delta t_{clk}\}/\sigma_{pr,g_l} \tag{29}$$

where pr_{i,g_l} is the pseudorange measurement from UAV 'i' to satellite 'l', $\mathbf{r}_{g_l}$ is the location of satellite 'l' at the relevant epoch, $\mathbf{r}_{n,i}$ is the UAV position, δt_{clk} ist the user clock offset, and σ_{pr,g_l} is the standard deviation of the pseudorange measurement. Note that for standalone GNSS position calculations, one should use the ordinary least squares (OLS), weighted least squares (WLS), or an Extended Kalman filter (EKF) as it is computationally more efficient. However, when integrating GNSS with other sensors for which factors can be setup, the factor-based descriptions may be more appropriate.

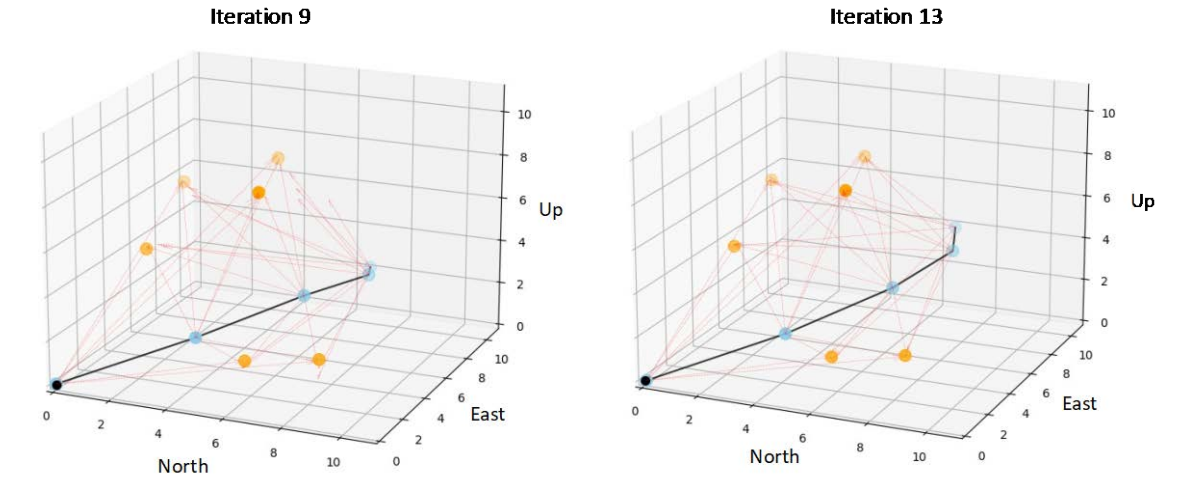

Figure 4-3: Beacon-based navigation results: (left) trajectory estimate for iteration 9; (right) trajectory estimate for final iteration 13.

Figure 4-4 (left) shows the results of the factor-graph-based method using GNSS pseudoranges. Basically, the results are identical as when we used a WLS solution.

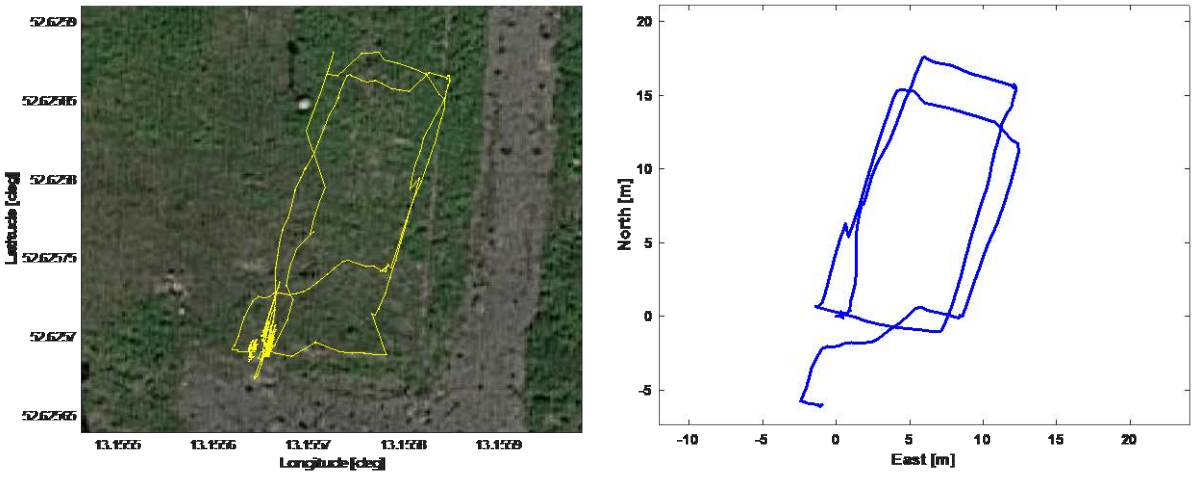

Figure 4-4: Example of computing a GNSS trajectory using factors rather than a WLS for a UAV flight in the TU Berlin UAV flight test area: (left) pseudorange-based positioning; (right) sequential carrier-phase based dead-reckoning.

In addition, we implemented a carrier-phase based position change estimator using the factor that relates changes in UAV position to sequential difference in carrier-phase measurements, or:

$$F_{i,sd_l} = \{sd_{i,g_l} - c_{dopp} - c_{geom} + \mathbf{e}_{g_l,prv} \cdot \Delta\mathbf{r}_{n,i} - \delta t_{clk}\}/\sigma_{sd,g_l} \tag{30}$$

where $\mathbf{e}_{g_l,prv}$ is the unit vector pointing to the GNSS satellite from the previous epoch, c_{dopp} is a Doppler compensation term, c_{geom} is a change in geometry compensation term, and $\Delta\mathbf{r}_{n,i} = \mathbf{r}_{n,i}(t_k) - \mathbf{r}_{n,i}(t_{k-1})$. More details on these compensation terms can be found in [12].

Rather than a state vector that contains only the user position, factor graph-based methods often include the user pose, i.e., both the user position and the orientation. The user's absolute pose is defined by the orientation of user 'i' with respect to the navigation frame, $\mathbf{R}_{n/b_i}$, part of the rotation group $\mathbf{SO}(3)$, and its position in the navigation frame, $\mathbf{r}_{n,i} \in \mathbb{R}^3$, given by the matrix:

$$\mathbf{T}_{n/b_i} = \begin{bmatrix} \mathbf{R}_{n/b_i} & \mathbf{r}_{n,i} \\ 0 & 1 \end{bmatrix} \tag{31}$$

The pose $\mathbf{T}_{n/b_i}$ is part of the so-called $\mathbf{SE}(3)$ group of rigid body motions and its manipulation follows Lie algebra [13] which must be considered when deriving the Jacobians used for the optimization process.

5.0 SIMULTANEOUS LOCALIZATION AND MAPPING

In addition to estimating the PNT-related state vector, it may often be necessary to simultaneously derive a map of the user or vehicle's surroundings. This map can then be used for collision avoidance, path planning, or it may just be one of the end-products of the navigation process. The approach of estimating pose and map at the same time is referred to as Simultaneous Localization and Mapping, or SLAM. Whereas in localization, the user is only interested in estimating state vector consisting of the position, velocity or pose (e.g., $\mathbf{x} =$

$[x, y, z, \varphi, \theta, \psi]^T$), SLAM methods estimate a state vector **y** that includes both the pose, **x,** and the map, **m**; or:

$$\mathbf{y}_k = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{m} \end{bmatrix} \tag{32}$$

In this equation, the subscript 'k' has been included to indicate the time epoch.

Both feature-based and featureless SLAM approaches exist. Whereas, in case of feature-based SLAM, the map consists of discrete features in the environment, the map in featureless SLAM is represented otherwise. Examples of maps in featureless SLAM are occupancy grids.

5.1 Feature-based SLAM

For example, some earlier SLAM methods such as the one described in [14] describes a method that takes 2D laser scans of the environment, extracts tree features from the laser scans using a predefined model of the trees, and then represents the trees by points in the map. The filter used to estimate the pose and the map simultaneously is an information filter [15][16] discusses several aspects of SLAM that must be considered, including observability, convergence, sensor and process models, consistency, information exploitation and efficiency.

Figure 5-1 illustrates the basic principle of feature-based SLAM using a 2D laser scanner. Figure 5-1 shows the initial 2D pose $\mathbf{x} = [x \quad y \quad \psi \ ]^T$ of the platform (i.e., robot) and the 6 point features, $\mathbf{m}_1 = [m_{x,1} \quad m_{y,1} \ ]^T$ through $\mathbf{m}_6 = [m_{x,6} \quad m_{y,6} \ ]^T$ in the environment. At time epoch t_1, features $\hat{\mathbf{m}}_1$ and $\hat{\mathbf{m}}_2$ are detected in the laser scan. As the feature extraction process introduces measurement errors, both map entries will have an associated uncertainty visualized by their covariance ellipses. Next, a dynamics model for the user is used to predict the platform position.

$$\bar{\mathbf{x}}_{k+1} = g(\mathbf{x}_k, \mathbf{u}_{k+1}) \tag{33}$$

where $\mathbf{u}_{k+1}$ is the control input. Note that in SLAM, the control input is often provided by odometer, inertial measurements and/or a model of the vehicle. This is different from typical implementations in the integrated navigation community, where the inertial measurements are considered in the measurement equations of the filter, rather than the state propagation equations.

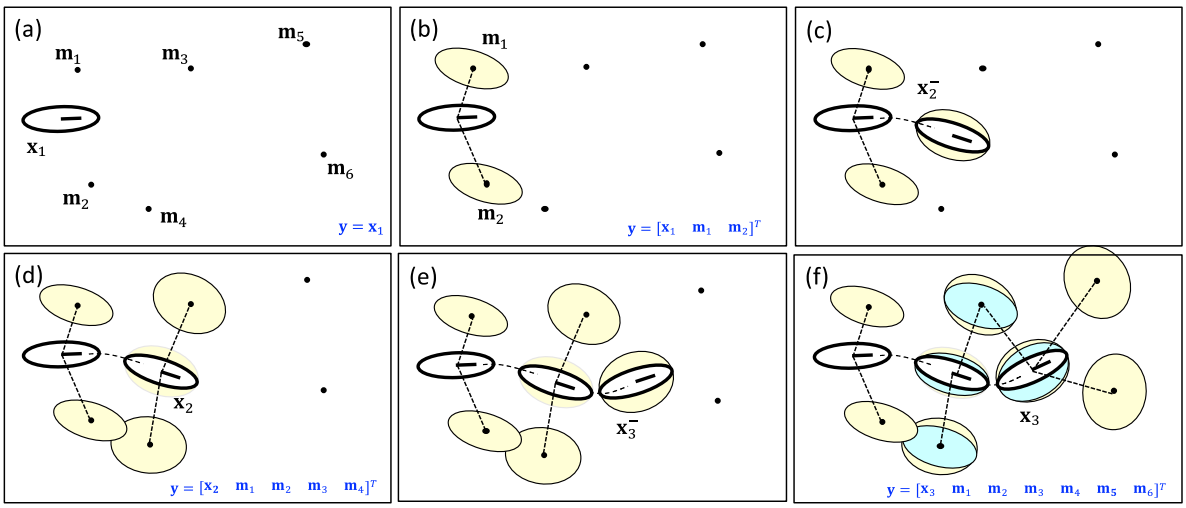

Figure 5-1: Basic principle of feature-based SLAM.

As illustrated in Figure 5-1, the prediction step introduces uncertainty, $\mathbf{P}_{xx}^{-}$ attributed to uncertainty in the control inputs or dynamics model. Next (Figure 5-1d), the prediction is combined with the measurements at time-epoch t_2 to estimate feature locations features $\hat{\mathbf{m}}_3$ and $\hat{\mathbf{m}}_4$, and state estimate $\hat{\mathbf{x}}_2$. Note that the uncertainty of these features is larger than $\hat{\mathbf{m}}_1$ and $\hat{\mathbf{m}}_2$, since it includes both the pose uncertainty and the laser-measurement uncertainty. Figure 5-1e shows the results of the next prediction step and the resulting uncertainty ellipse. In Figure 5-1e, the laser-scanner on the platform observes features $\hat{\mathbf{m}}_5$ and $\hat{\mathbf{m}}_6$ and re-observes feature $\hat{\mathbf{m}}_3$. Revisiting feature $\mathbf{m}_3$ causes a reduction in covariance of the related state variables ($\mathbf{x}_3$, $\hat{\mathbf{m}}_3$, $\hat{\mathbf{m}}_4$). At a larger scale this is referred to as *loop closure and* can lead to significant reduction of the estimate covariance when parts of the environment are re-visited after long periods of time. During the steps show in Figure 5-1, the dimension of the state vector $\boldsymbol{y}$ and corresponding covariance matrix, $\mathbf{P}$, increases with every new map feature that is added. Examples of an extended Kalman filter (EKF) and an information filter (IF) based SLAM implementation can be found in [14] and [15], respectively. With many more in the SLAM literature.

An important step in feature-based SLAM is the data association process. One data association approach calculates the Mahalanobis distance (i.e., the normalized squared innovation), M_{ij}, from each observed feature '*i*' to each map feature '*j*'. Typically, a statistical threshold (or gate) for M_{ij} is defined against which associations are tested: if the measurement is within this so-called tracking gate, the association is valid. In case the innovation is Gaussian distributed, M_{ij} is χ^2 distributed. A possible threshold could be $M_{ij} < \gamma = 6$. For this value $P_{\chi^2}(M_{ij} \leq 6) = 0.95$. In some cases, multiple measurements fall within a tracking gate, in which case, additional rules must be established to identify the correct association. One example would be the nearest-neighbor rule, which identifies the measurement with the smallest Mahalanobis distance as the correct association.

For large maps (i.e., a great number of features and, thus, a large state vector and covariance matrix), it will be computationally infeasible to implement an EKF or IF. Various methods have been investigated to deal with this increase in dimensionality. Examples include the use of sub-maps, the removal of landmarks that do not have to be further estimated, covariance intersection (CI) methods, and sparse extended information filters (SIEF).

A PF feature-based SLAM implementation has also been implemented as one of the earlier feature-based implementations [17]. These methods exploit the fact that knowledge of the platform's true path would render the position of *M* features or landmarks conditionally independent, or:

$$p(\mathbf{x}_k, \mathbf{m}|\mathbf{z}_k, \mathbf{u}_k) = p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{u}_k)p(\mathbf{m}|\mathbf{x}_k, \mathbf{z}_k) = \underbrace{p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{u}_k)}_{\text{Pose}} \underbrace{\prod_{i=1}^{M} p(\mathbf{m}_i|\mathbf{x}_k, \mathbf{z}_k)}_{\text{Individual features}} \tag{34}$$

This factorization is referred to as Rao-Blackwellization. FastSLAM is a SLAM method that exploits this factorization and combines it with a participle filter (PF). The result is a filter in which each particle consists of a 2D pose vector and *M* simple 2D Kalman filters for each of the landmarks.

Most recent approaches use FGO to solve the feature-based SLAM problem. In these SLAM methods, the measurements and control inputs are stored in a graph-like structure and optimized using a linear solver. Note that the factors are a function of both the state vector and the features stored in the map (e.g., bearing or range measurements to the features). Example graph-based SLAM methods are GraphSLAM [15], iSAM [18], and g2o [7]. Often, the graph is built up while data is being collected from the sensor in the so-called front-end processor (see Figure 5-3). This front-end could calculate an initial estimate of the platform trajectory and map, perform data association, and build the factor graph. The back-end of the processor then solves the factor graph for the optimal trajectory and map.

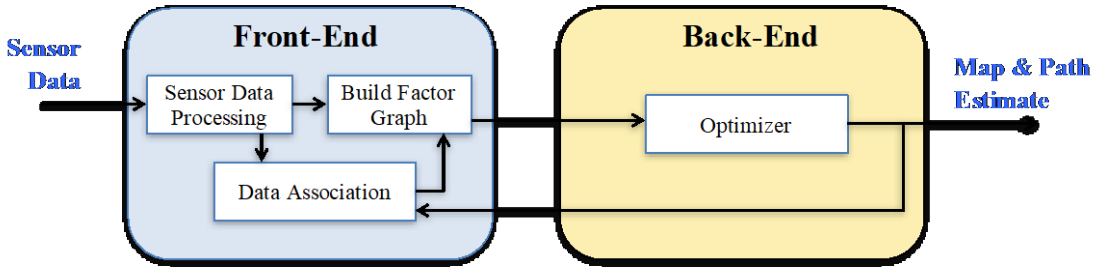

Figure 5-3: Front-end and back-end processing for graph-based SLAM methods.

5.2 Featureless SLAM

Instead of representing the observed environment by features, featureless methods represent the environment by alternative means such as parametric models, surfaces, meshes and occupancy grids. Grid based mapping was first introduced for sonar sensors [19] and followed by the concept of occupancy grid in [20]. The latter is also referred to as an evidence grid.

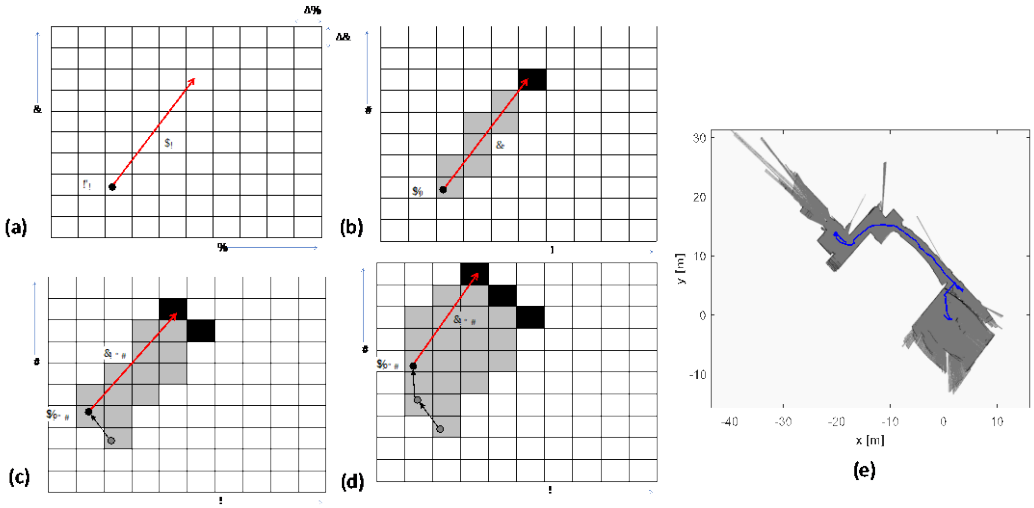

Figure 5-4. (a) through (d) Basic principle of forming an occupancy grid (grey: misses, black: hits): (e) Example of an occupancy grid with a superimposed aerial robot trajectory.

In occupancy grid-based methods using, for example, laser range scanners, the range measurements are used to determine what areas of the user's environments are occupied. The basic principle of forming the occupancy grid is illustrated in Figure 5-4a-d for multiple time epochs and a single laser beam. A 2D grid with spatial resolution $\Delta x, \Delta y$ is defined initially. Next, treat the measurements, $\mathbf{z}$, as a beam rather than a point, and find all cells of the grid that are traversed by this beam and identify those either as 'free' (gray) or as an 'end-point' (black). For known pose estimates, $\hat{\mathbf{x}}_k$, this process can be continued for each time epoch. Figure 5-4e shows an example of an occupancy grid obtained by using a 2D laser scanner on an indoor UAS. The white areas in the example are "undiscovered," whose status can be occupied or not, the grey areas are unoccupied (referred to as a "miss") and the black cells indicate where a laser beam "hit" an object indicating an edge between occupied and unoccupied areas. It is important to define a good model of the sensor (e.g., laser or sonar) when identifying each traversed cell. This 2D grid could be expanded to a 3D grid and represented using voxels or an octree, but computational complexity should be considered in that case.

Estimating the platform's pose based on an occupancy grid, typically follows the following steps:

1) **Grid estimation**: build a local map from a laser scan.

2) **Matching and outlier rejection**: match the local map or laser scan against a target map (or submap), $\mathbf{m}_k$. The target map could be a map based on previous sensor observation or an *a priori* map available to the platform.

3) **Optimization**: minimize an error metric function to obtain the motion or pose estimates.

The matching process is illustrated in Figure 5-5. Suppose we have a map based on all measurements so far, $\mathbf{m}$, given by the shown occupancy grid. Furthermore, we have predicted the platform pose based on, for example, odometer or inertial measurements, $\mathbf{u}_k$, or a dynamics model $\hat{\mathbf{x}}_k^- = g(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$. Now, suppose we receive three laser range measurements, $\mathbf{z}_k$, corresponding to the three "rays" in Figure 5-6. Figure 5-6 (a) through (d) show four examples of choices for the pose estimate $\hat{\mathbf{x}}_k$: $\hat{\mathbf{x}}_{a,k}, \hat{\mathbf{x}}_{b,k}, \hat{\mathbf{x}}_{c,k}$ and $\hat{\mathbf{x}}_{d,k}$.

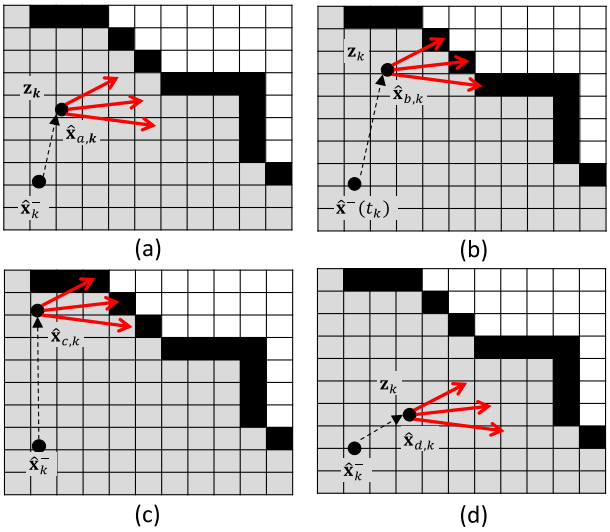

Figure 5-5: Pose estimation based on matching the laser scan against available map, $\mathbf{m_k}$.

For each of these choices you can calculate the map that would result from these measurements:

$$\hat{\mathrm{m}}_{\mathrm{z}} = \mathrm{h}^{-1}[\mathrm{x}(t_k), \mathrm{z}(t_k)] \tag{35}$$

Next, the distance between this map and the existing map can be established by finding the closest occupied grid element. For larger distances, the likelihood that there is a match between this measurement and the map becomes smaller. The likelihood function can, thus, be modelled as a function of d. A normal distribution could be used for this relationship, $\mathcal{N}(d, \sigma^2)$. Given all measurements, the likelihood function of all measurements can be approximated as.

$$\begin{aligned} p(\mathbf{z}(t_k)|\mathbf{x}(t_k), \boldsymbol{m}) &\approx \prod_i \mathcal{N}(d, \sigma^2) \\ d &= \min_{[xy]} \left\{ \left| \hat{\mathbf{m}}_{\mathbf{z}} - \mathbf{m}^{[xy]} \right| \text{ given } \mathbf{m}^{[xy]} \text{ is occupied} \right\} \end{aligned} \tag{36}$$

For our example, $P(\mathbf{z}(t_k)|\mathbf{x}_d(t_k), \mathbf{m})$ is the smallest. Thus, off all four options, (b) is the most likely match. Using the maximum likelihood criterion.

This laser scan matching process can also be formulated using a cost criterion that must be minimized:

$$\mathcal{C}(\mathbf{R}_k^M, \mathbf{t}_k^M) = \sum_{i=1}^{K} \left[1 - M_{smooth}\left(\mathbf{R}_k^M \mathbf{p}_i(t_k) + \mathbf{t}_k^M\right)\right]^2 \quad (37)$$

where $\mathbf{R}_k^M$ and $\mathbf{t}_k^M$ are the rotation and translation from the laser scan frame to the local map coordinate frame, and $M_{smooth}$is smooth version of the probability values in the local submap of the occupancy grid. Equation (37) can be solved using a numerical solver. For example, Google cartographer [21] uses the Ceres solver [8] to estimate the rotation and translation within a submap.

A good example of what happens when a loop closure function is not included is shown in Figure 5-6. This figure shows a 2D map generated using a small UAV (sUAS) equipped with a laser range scanner mounted and flown in an indoor office environment using a grid-based SLAM method (Ohio University Stocker engineering building).

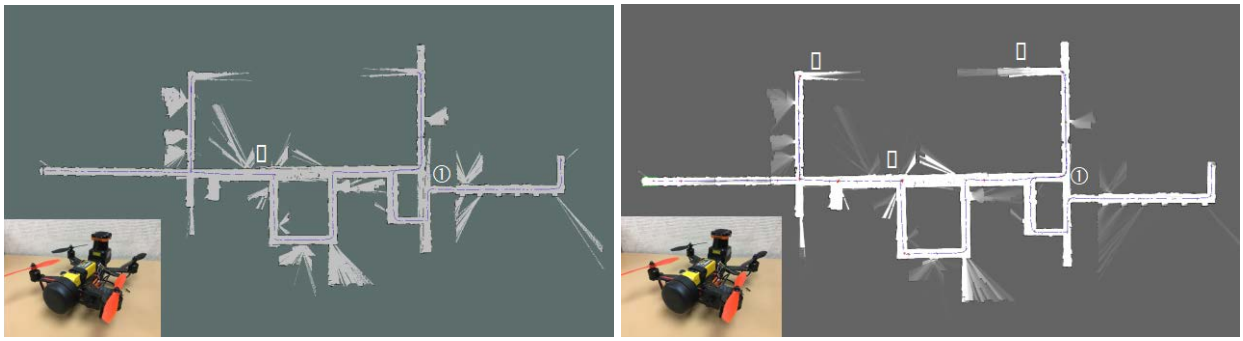

Figure 5-6 (left) Small UAV mapping results for Ohio University Stocker Center third floor using HectorSLAM [22]; (right) sUAS mapping results for Ohio University Stocker Center third floor using Google Cartographer [21]

The lack of loop closure can be easily observed at locations 1 and 2 where map distortion is apparent. This effect can be attributed to a bias in the pose estimate when the sUAS returns to location 1 (and location 2) due to the inherent dead-reckoning behavior of SLAM and possible lack of observability in the along-track direction due to similarity in the environment (i.e., lack of sufficient features).

Alternatively, Figure 5-6 (right) shows the results for an FGO-based method that includes loop-closure such as Google cartographer; they do not suffer from these artifacts. The larger map in Google cartographer consists of numerous smaller submaps related using relative poses that are re-estimated using non-linear solvers (FGO) such as Ceres when an area is revisited. During this mapping mission, the sUAS did not fly through the hallway between locations 3 and 4 resulting in an unresolved misalignment between the two visible hallway segments. It is expected that this misalignment would have been resolved by loop closure optimization if the sUAS would have completed its mission and flown the extent of the hallway.

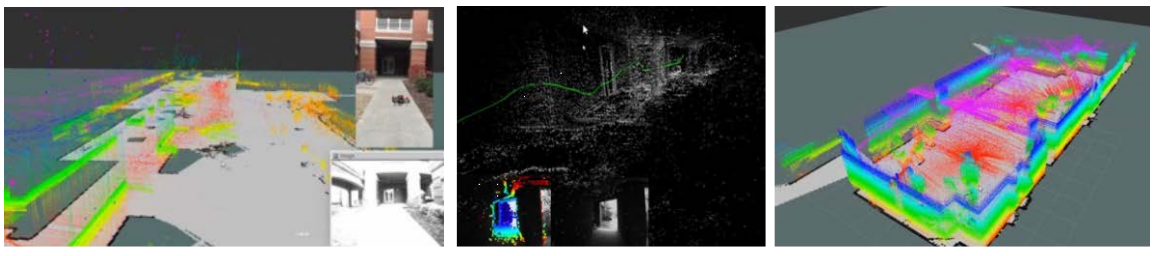

Figure 5-7 (a) sUAS equipped with two laser scanners and a camera flying in a challenging environment; (b) Vision-based SLAM results, (c) Laser-based feature-less SLAM results from the two laser scanners.

Figure 5-7 shows the results of applying both laser-based and vision-based SLAM techniques onboard an sUAS [23].

Many SLAM approaches have been developed that exploit the cooperation of multiple vehicles/agents (ground or air), a.k.a. Cooperative SLAM (C-SLAM) or distributed SLAM [24][25]. Most of these methods focus extensively on the merging of the maps formed by the individual members or agents using the exchange of measurements or knowledge of common regions and objects, or by exchanging the maps and aligning them using point feature matching, scan matching or other map matching techniques.

6.0 SUMMARY AND CONCLUSIONS

This paper gives an overview of particle filters, factor graph optimization methods, and simultaneous localization and mapping focusing on the vehicle navigation problem. The discussed methods can be used as the underlying estimator when performing PNT in environments where sensor integration is required to meet the required navigation performance for the application. Furthermore, all discussed methods lend themselves for cooperative estimation either by fusing the individual states of the cooperating vehicles using, for example, CI, or directly by collection the measurements from others and setting up a factor graph. It is important to always consider of the chosen method is appropriate in terms of computation requirement (e.g., PF versus EKF or UKF).

7.0 REFERENCES

[1] International Civil Aviation Organization. ICAO Doc 9613, Performance-based Navigation (PBN) Manual. Technical report, International Civil Aviation Organization, Montreal, 2008.

[2] P. S. Maybeck, *Stochastic Models, Estimation and Control – Vol. II*, Navtech Books and Software Store, 1994.

[3] B. Ristic, et al., *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House, 2004.

[4] R. P. S. Mahler "Optimal/robust distributed data fusion: a unified approach", Proc. SPIE 4052, Signal Processing, Sensor Fusion, and Target Recognition IX, (4 August 2000); https://doi.org/10.1117/12.395064.

[5] M. B. Hurley, "An Information Theoretic Justification for Covariance Intersection and its Generalization," in *Proceedings of the 5th Inter- national Conference on Information Fusion (Fusion 2002)*, Annapolis, Maryland, USA, Jul. 2002.

[6] O. Hlinka, O. Sluciak, F. Hlawatsch, M. Rupp, "Distributed data fusion using iterative covariance intersection," *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[7] R. Kuemmerle, et al., "g2o: A General Framework for Graph Optimization," Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2011.

[8] S. Agarwal, K. Mierle, et al., "Ceres Solver", http://ceres-solver.org.

[9] F. Dellaert and GTSAM Contributors, "borglab/gtsam," Georgia Tech Borg Lab, https://github.com/borglab/gtsam, May, 2022.

[10] H. Martiros, et al., "SymForce: Symbolic Computation and Code Generation for Robotics," Proceedings of Robotics: Science and Systems, 2022.

[11] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *Int. Journal of Robotics Research,* 25(5-6):403, 2006.

[12] F. van Graas, A. Soloviev, "Precise Velocity Estimation Using a Stand-Alone GPS Receiver," *NAVIGATION: Journal of The Institute of Navigation,* Vol. 51, No. 4, 2004.

[13] J.Sola, J. Deray, D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2018.

[14] Guivant, J., E, Nebot, S. Baiker, "Localization and Map Building Using Laser Range Sensors in Outdoor Applications," *Journal of Robotic Systems*, 2010, 17(10), pp. 565-583.

[15] Thrun, S., Burgard, W., & Fox, D., *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, Cambridge: MIT Press, 2005.

[16] Dissanayake, G., Williams, S., Durrant-Whyte, H., & Bailey, T. , "Map Management for Efficient Simultaneous Localization and Mapping (SLAM)," *Autonomous Robots*, 12 (3), 2002, pp. 267–286.

[17] D. Hähnel, D. Fox, W. Burgard, S. Thrun, "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2003.

[18] Kaess, M., A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, Vol. 24, No. 6, 2008, pp. 1365-1378.

[19] Moravec, H., and Elfes, A., "High Resolution Maps from Wide Angle Sonar," *IEEE International Conference on Robotics and Automation (ICRA)*, 1985, pp. 116-121.

[20] Elfes, A., "Using Occupancy Grids for Mobile Robot Perception and Navigation," *IEEE* , 22 (6), 1989, pp. 46-57.

[21] Hess, W., D. Kohler, H. Rapp, and D. Andor, "Real-Time Loop Closure in 2D LIDAR SLAM," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016. pp. 1271–1278.

[22] Kohlbrecher, S., O. von Stryk, J. Meyer, U. Klingauf, "A Flexible and Scalable SLAM System with Full 3D Motion Estimates," *Proceedings of the IEEE Conference on Safety, Security, and Rescue Robotics*, 2011.

[23] M. Uijt de Haag, J. Robinson, J. Huff, A. Schultz, "Assessing Indoor Environments with sUAS through Real-Time Virtual Reality and Assured Navigation," *Proceedings of the ION International Technical Meeting (ITM)*, Reston, VA, January 2018.

[24] Y. Rizk, M. Awad, E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Computing Surveys (CSUR)*, 52(2), 1-31, 2019.

[25] H. C. Lee, S. H. Lee, T. S. Lee, D. J. Kim, B. H. Lee, "A survey of map merging techniques for cooperative-SLAM," *Proceedings of the IEEE 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 285-287, 2012.