A Tripartite Filter Design for Seamless Pedestrian Navigation using Recursive 2-Means clustering and Tukey Update

Pekka Peltola, Jialin Xiao, Chris Hill and Terry Moore
Nottingham Geospatial Institute
Nottingham University
Nottingham, England
{pekka.peltola, jialin.xiao, chris.hill,
terry.moore}@nottingham.ac.uk

Fernando Seco and Antonio R. Jiménez
Centre for Automation and Robotics, CAR
CSIC-UPM
Madrid, Spain
{pekka.peltola, fernando.seco, antonio.jimenez}@csic.es

***Abstract*—Mobile devices are desired to guide users seamlessly to diverse destinations indoors and outdoors. The positioning fixing subsystems often provide poor quality measurements with gaps in an urban environment. No single position fixing technology works continuously. Many sensor fusion variations have been previously trialed to overcome this challenge, including the particle filter that is robust and the Kalman filter which is fast. However, a lack exists, of context aware, seamless systems that are able to use the most fit sensors and methods in the correct context. A novel adaptive and modular tripartite navigation filter design is presented to enable seamless navigation. It consists of a sensor subsystem, a context inference and a navigation filter blocks. A foot-mounted inertial measurement unit (IMU), a Global Navigation Satellite System (GNSS) receiver, Bluetooth Low Energy (BLE) and Ultra-wideband (UWB) positioning systems were used in the evaluation implementation of this design. A novel recursive 2-means clustering method was developed to track multiple hypotheses when there are gaps in position fixes. The closest hypothesis to a new position fix is selected when the gap ends. Moreover, when the position fix quality measure is not reliable, a fusion approach using a Tukey-style particle filter measurement update is introduced. Results show the successful operation of the design implementation. The Tukey update improves accuracy by 5% and together with the clustering method the system robustness is enhanced.**

Keywords—tripartite; modular; pedestrian navigation; Kalman; particle; filter; clustering; Tukey; adaptive

I. INTRODUCTION

Outdoors, the Global Navigation Satellite Systems (GNSS) receiver provides navigation services to the user. Indoors and in urban environment, multipath and signal attenuation notably hamper the receiver operation, and complementary technologies need to be used. Environmental feature information, like building lights in [1], can be observed and used together with beacon based positioning fixing systems and with dead-reckoning. As no single technology solution is yet available, the multi-sensor fusion approach seems to be the answer for the indoor-outdoor navigation scenario.

Multi-sensor fusion involves handling of varying types of information sources. Groves et al. [2], examine the complexity and ambiguity that are inherent in multi-sensor systems. The best sensors and methods differ according to the changing environment and context. The challenge is then, how to use the most fit methods in the correct context and how to resolve situations when sensor subsystems provide discontinuous and bad quality positioning fixing measurements.

This paper examines these two challenges in navigation application development. Seamlessness demands the navigation application to be dynamic and adaptive. Dynamic, in the sense of the system being able to respond quickly to a change in context. And adaptive, as being able to select the correct configuration and methodology for the situation. Using a Kalman filter saves computational power, when the solution is clearly unimodal. A particle filter is the more robust option otherwise. Although a particle filter can survive multimodal situations, it needs additional aiding in the inference of the correct solution.

To enhance seamlessness, a novel modular tripartite pedestrian navigation filter design was developed in this paper. Fig. 1 shows the top-down block diagram. The tripartite filter consists of three blocks and enables dynamic plug-and-play style approach for each block of the navigation system design. The design can adapt solution computing resources and

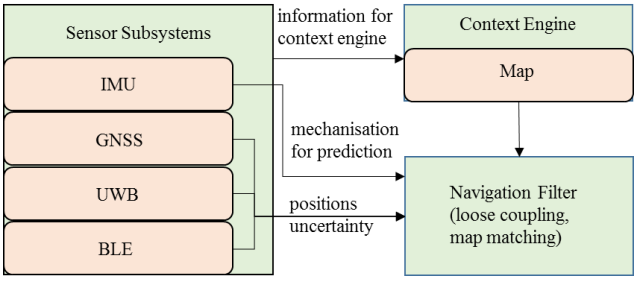

Fig. 1. The top-down structure of the tripartite filter. IMU, GNSS, UWB and BLE sensor subsystems are loosely coupled in the navigation filter according to the control of the context engine.

methods according to the inferred context. Four previously preselected sensor technologies [3, 4] were used in an example implementation. Position estimates with uncertainties from three sensor subsystems are loosely coupled with the inertial solution. The inertial dead-reckoning is used to track the foot and it provides the propagation model and a short-term solution for both indoor and outdoor environments. A GNSS receiver provides position information outdoors, which is fused with the IMU data. Indoors, the Bluetooth low energy (BLE) and Ultra-wideband (UWB) sensor subsystem position estimates replace the GNSS in the fusion process. A Gaussian measurement model is used, adjusted with a Tukey approach.

A novel recursive 2-means clustering method was developed for resolving multimodal situations that result, due to gaps in positioning fixing measurements. The resampling method was readjusted and a Tukey-style particle weight update was developed to enhance the particle filter survivability when positioning fixes are of a bad quality.

Walk tests were made on the Jubilee Campus of the University of Nottingham. The campus represents a challenging urban environment with areas susceptible to multipath. The filter implementation is targeted for pedestrian applications at first. The design is modular and easily extendable to vehicle based navigation.

The section II examines the background. The section III introduces the tripartite filter design, the Tukey update and the recursive 2-means clustering. The section IV evaluates the operation of the modular and adaptive filter design. Section V is the Discussion and section VI concludes the research outcome.

II. Background

The challenges examined in this paper were encountered in the previous work [3]. The Fig. 2 is an example of a situation where the user moves from indoors to outdoors. The Indoor UWB position subsystem measurements stop at the entrance. The UWB position measurements are not shown in the figure. During the cold start of a GNSS receiver, the fused solution relies only on the inertial mechanisation. The fusion solution is shown in magenta. The yellow arrow indicates a point, when

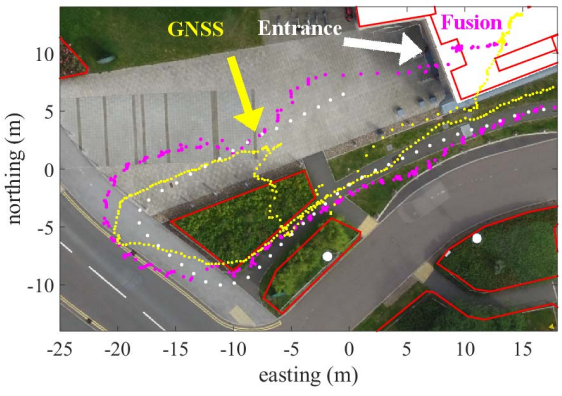

Fig. 2. Indoor outdoor context switch. GNSS measurements are yellow. The fusion solution is magenta. White is the reference. At the yellow arrow, the GNSS quality improves rapidly.

the GNSS measurement quality improves. The inertial track heads slightly too much north before the quality of the GNSS measurements increases.

Considering the challenges in more general terms, Groves et al. [2] identified four issues in multi-sensor navigation - complexity, context, ambiguity and environmental data handling. Complexity comes with the task of combining information from many different types of sensors. Rein and Biermann [5] agree that fusing the different information from different sources having varying worldviews is a complex task. Worldview here refers to, for example, sensors not necessarily being developed for navigation purposes, but still being used as such. Rein and Biermann state that the worldview can be described as the state of the system and as the relationships between the system blocks. A centrally processed cooperative positioning system with a joint state was used for tracking four users in [6].

Sensors provide information on different levels of abstraction. Snidaro et al. [7] represent the abstraction of data in horizontal and vertical levels. The vertical abstraction level of sensor data describes what information abstraction level the sensor outputs, ranging from raw binary sensor data to highly abstract natural language-based data representations. For sensor data, the horizontal abstraction can be described as different ways (physical measures) of representing the same vertical level sensor data. Zichar [8] examines the amount of necessary information presented for the user. It is essential to weigh the relationship between computing resources and processing only the necessary information for the current task. A navigation system should be able to handle situations both with excess and lack of information. This paper examines information fusion using loosely coupled context sensitive approach.

Different sensor fusion methods exist. Zaugg et al. [9] used both particle and Kalman filter for target tracking. A Kalman filter is fast while particle filter is more robust in resolving ambiguous positioning scenarios. To be able to use the complementary characteristics of both methods in the same application, a context derivation module is needed.

Detection of the context is thus essential. Groves [2] divides context determination into two. The behavioural context determination measures the user behaviour, while environmental context determination measures the surroundings. Rivero-Rodriguez et al. [10] compared 5 different methods for deriving the user context. The behaviour of how the mobile phone was used in different locations was examined. Bagged Tree method resulted to be the most efficient in deriving the correct location context. Location contexts were in the abstraction level of home, work etc.

Parviainen [11] further divides the context determination into participatory and opportunistic approaches. The source for the context information can come directly from the user. According to Rein and Biermann [5], this requires that the system is able to correctly translate or format, the sometimes highly abstract user input. Another way, is to passively measure available information without user intervention. In our work we applied the latter, although it is straightforward to add user input based context inference methods into our design.

Different contexts can be described as states. A finite-state machine (FSM) includes a list of states and possible transitions with different conditions. Perttula [12] and Kronenwett [13] used a state diagram to represent step detection algorithms. A FSM can switch between states according to the external inputs and defines the system behaviour. We used a two state FSM that includes a particle filter state and a Kalman filter with a pentagon buffer state. The transition conditions will be introduced.

Perttula [12] used a line segment intersection method for map matching the particles that cross wall lines. We used this with a method that checks if a particle is inside a polygon that is defined as unpassable.

Adapting to the changing availability of different positioning system measurements and the varying quality of the positioning fixing measurements should be taken into account. A GNSS receiver multipath condition is shown in Fig. 3. Result differs from the reference. The challenge is in how to fuse bad quality position fixes. Radavi tested different windows for location estimation in [14]. The Tukey-style update for the particle filter was developed for these situations.

When positioning fixing gaps appear or measurement quality is low for longer periods the estimation becomes more uncertain. K-means clustering is one of the well-known algorithms for cluster analysis. The algorithm partitions a set of observations into k clusters and aims to minimize each within-cluster sum-of-square. In [15], Razavi et al. introduced a k-means algorithm for fingerprint clustering in floor estimation. This method reduced the fingerprinting data size by storing and transmitting the cluster heads and related floor labels. Further, Altintas and Serif [16] presented a k-means clustering based method to improve the K Nearest Neighbour (KNN) algorithm in RSS based indoor positioning. According to the distance between the mobile user and the neighbours, the k-means algorithm can group the nearest neighbours to raise the selection of neighbouring points for location calculation.

In this study, a recursive 2-means clustering method is designed for scenarios with gaps or bad quality in positioning fixing system measurements. The method enables following multiple hypotheses and selecting the closest matching track, when positioning fixes are again available.

The questions raised in this section will be examined in the following sections. First, the adaptive filter structure for the differing contexts is presented. Afterwards, the Tukey update and the recursive 2-means clustering are presented to compensate for the trouble encountered with gaps and low-quality position fixes.

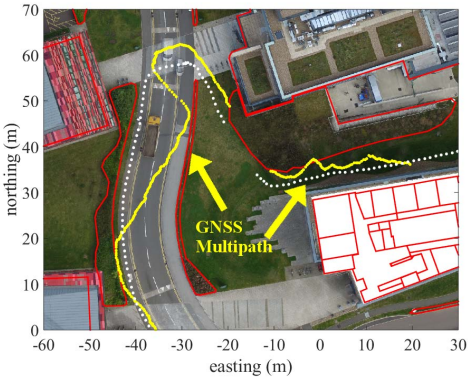

Fig. 3. GNSS (yellow) receiver multipath. White is the reference path. Red lines are walls.

III. Tripartite Filter Structure and Operation

To increase the adaptability of a navigation system a tripartite filter was designed. The tripartite filter consists of three blocks as seen in Fig. 1. These three blocks divide the necessary responsibilities for an adaptable navigation system. The sensor subsystems block handles the raw sensor data by formatting and transmitting it to the two other blocks. The context engine block contains the state information and infers changes to the system state using the latest sensor data and the previous state information. The navigation filter derives the new positioning solution.

Fig. 4 shows the MATLAB main loop activity diagram and the tripartite filter object oriented design diagram. A control setup file defines what objects each block will contain when the application is executed. Sensor subsystems object consists of a list of sensor subsystems and of a virtual method that handles the selection of the next measurement and its format. Similarly, the context engine object has a list of context derivation methods and a virtual method that handles the execution order of the method list. In addition, the context engine holds the map and the state information. Further, the navigation filter contains a list of solution derivation methods and a virtual method that handles the execution order of this list. The main loop runs until a stop condition is true. In our implementation this was when no more measurements were available. Now we examine each block in greater detail.

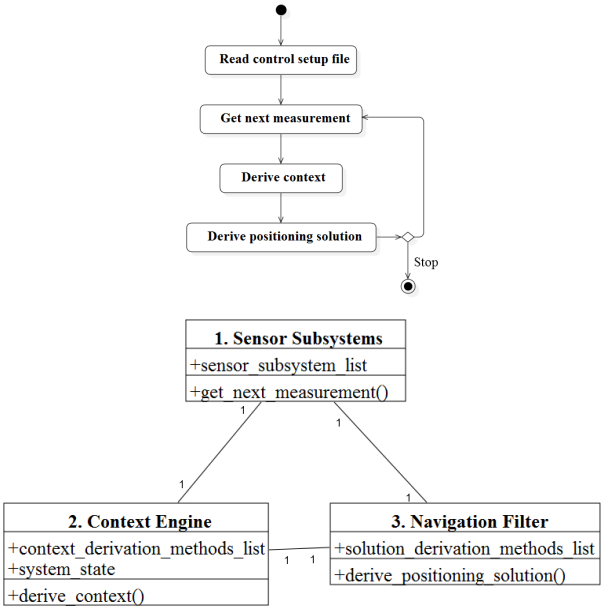

Fig. 4. Above the Matlab main loop executes the virtual methods in the blocks numbered from 1 to 3.

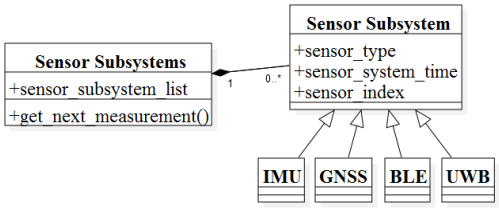

Fig. 5. Sensor subsystems object provides the latest measurement formatted for the context engine and the navigation filter.

A. *Sensor Subsystems Object*

The sensor subsystems object consists of a list of sensor subsystem objects. Our implemented example has four sensor subsystems. Fig. 5 shows the object diagram. The inertial measurement unit object implementation (IMU) is for the Microstrain 3DM-GX4-45 sensor in 100Hz sampling rate mode. The GNSS sensor is the U-Blox 6 GNSS receiver inside the Microstrain sensor recording at 4Hz rate. It provides position and position error estimates. The ultra-wideband (UWB) ranging kit is the Decawave's TREK1000, outputting ranges at 1Hz rate. The UWB sensor subsystems provides weighted iterative least squares position and position error estimates when all three anchors provide ranges. The Bluetooth low energy (BLE) subsystem consists of six BLE anchors; of two Raspberry Pis with CSR BLE adapters and of four Kontak.io beacons. The Raspberry Pis advertise at the rate of 50Hz and the Kontakt beacons at 10Hz. A path-loss grid map (10cm grid side length) probability estimate is derived for each anchor when a 1 second window of measurements arrives. These six difference maps are combined. The most likely grid position and a standard deviation estimate are provided.

C-code was written for Debian, running on a laptop which recorded measurements for each sensor with associated system time stamps and sensor time stamps. The tripartite filter implementation uses the recorded output files in MATLAB. The tripartite filter is designed for real-time applications. If the sensor subsystems are implemented with real-time input data pipes the system would run in real-time.

The virtual method, *get_next_measurement()*, gets the latest measurement, formatted by the corresponding sensor subsystems implementation, and passes it onward. This method is the first line in the main program loop (see Fig. 4). The program loop processes one measurement a time.

TABLE I. STATE FLAGS

STAIRS	INDOOR	ENTRANCE	ZUPT	STEP	POSITION	POSITION FIX GAP

B. *Context Engine Object*

The context engine holds the system state attributes. The state information is required to be able to adjust the navigation filter parameters like the propagation [17] or the update model. The state consists of flags, shown in Table I, and of the position solution that is the northing and the easting on a local tangent plane. The flags can be either true or false. The context

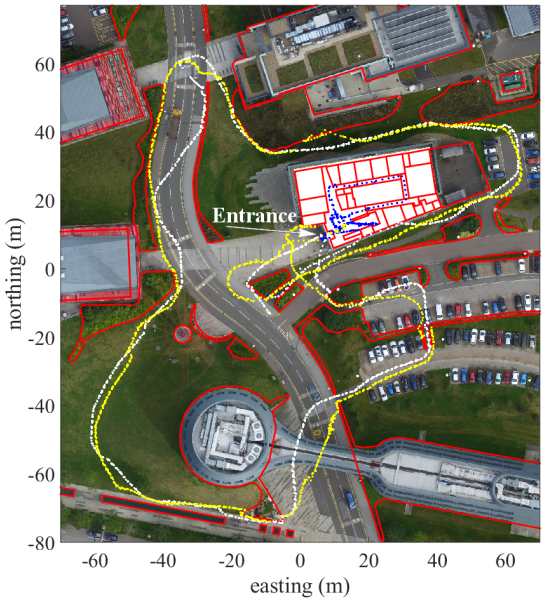

Fig. 6. The white track shows user walking south from the entrance, walking clockwise around the area outdoors and then entering indoors. The blue track shows the indoor walk. The user leaves the building. The yellow track is the reverse walk trace outdoors.

engine has a list of context inference methods that adjust these flags. In addition, the context engine holds the map object. The map consists of a wall information structure acquired from kml files and of a high definition georeferenced map image that was recorded with an unmanned aerial vehicle. Fig. 6 shows the georeferenced map placed into a local tangent plane figure in MATLAB. Walls are in red. The resulting track of a one seamless test walk is shown in three parts. The white track starts from the entrance and travels south. After a 500m clockwise walk in the area the user enters indoors. The blue track shows the indoor walk. The user then leaves the building and walks the outdoor track in the reverse direction. This is shown in yellow.

Fig. 7 shows the structure of the context engine object holding the implemented methods and the map. The context derivation methods are a continued development of the work

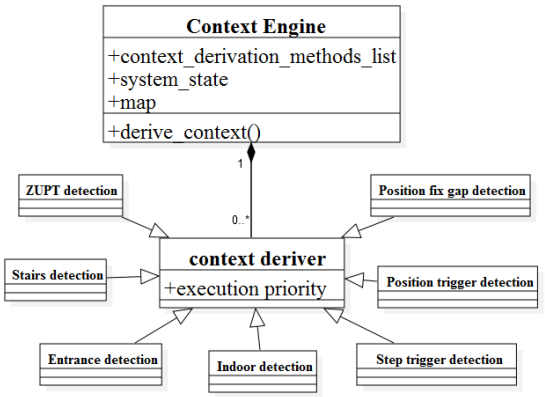

Fig. 7. The context engine derives the flags using the list of implemented methods and according to the previous state and the latest measurement.

done previously and described in [3]. The zero update (ZUPT) detection method checks the near real-time low pass filtered outputs of the total accelerometer magnitude using two thresholds (1,08g and 0.92g) and of the total gyroscope magnitude using one threshold (0.3rad/s). A combined AND function determines the ZUPT flag for the current epoch. Current epoch is delayed by 70ms. This is the time constant for the ZUPT filter. It smoothens the spikes encountered, especially when the foot hits the ground. The step flag is risen, every time the ZUPT value switches from true to false.

Position flag is risen every time a position measurement arrives from the three positioning fixing sensor subsystems. If no position measurement has arrived within a threshold of three seconds, the positioning fix gap detection flag is raised.

The stairs detection method checks the position change in z-direction and compares the position with the stairs location on the map every time step flag is risen. Stairs flag is raised or lowered if both apply.

The entrance flag is risen if the current position is near (below threshold) an entrance. Entrance flag enables the code object for the detection of the indoor flag. The indoor flag is changed if the position is inside or outside and positioning fixes change from low quality GNSS to high quality UWB or BLE or vice versa.

The *derive_context()* virtual method is the second function call in the main loop (see Fig. 4). All inference methods are executed. The context inference methods have execution priority order defined in the control setup file. The state, the flags, adjust the navigation filter behaviour. The navigation filter adjusts the flag values, after acting upon them. Feedback for the sensor subsystems can be used if necessary.

The state, or "worldview" as Rein and Biermann [5] describe, is implemented in the context engine. It is not internal to the context engine and can be shared. This enables designing inter-relations between the three functional blocks and possible other blocks including the execution order of the virtual methods. The design is developed for navigation applications and includes common general methods found in many navigation applications.

C. Navigation Filter Object

The navigation filter derives the position solution according to the current state, which is the flags and the previous position from the context engine. The *derive_positioning_solution()* virtual function is executed in the main loop (see Fig. 4). This processes the list of filters according to the priorities that are defined in the control setup file.

Fig. 8 shows the implemented filters. HF is a hybrid filter that consists of a particle and Kalman filters. The predict, update and resample methods are implemented in this filter. MM is the map matcher which uses the map together with the derived position solution from the HF. SC is the recursive 2-means clustering filter, which is initialised when needed. TFM is a track form matching filter that compares the inertial only Kalman and the GNSS tracks. This paper concentrates on examining the HF and SC filters. The idea and purpose of the MM and TFM filters are shortly examined here.

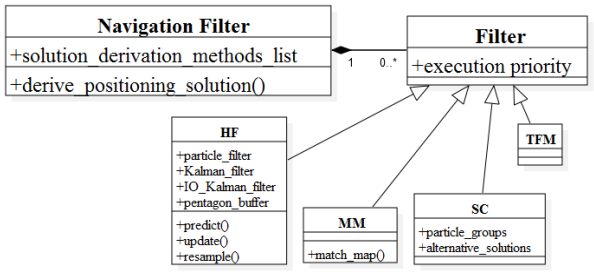

Fig. 8. The virtual derive_positioning_solution() method executes the implemented filter methods according to the execution priority.

The map matcher (MM), implements the interaction logic or the collision logic (CL), between the walls, particles, the main solution and the pentagon buffer. These were presented previously in [3]. Line segment intersection method is used together with a method that checks whether a point is inside a polygon.

The track form matching filter (TFM) was developed to compare the inertial only track and the GNSS track. The inertial track can usually be trusted to measure short walking distances well (see Fig. 14). GNSS trail on the other hand can change very quickly in short distances as seen in Fig. 3. Thus, comparing track direction changes between the inertial and GNSS tracks, in short track sections (up to 70m), we can infer whether the GNSS track might suffer from a condition, like multipath. Likewise, for longer track sections (over 100m), we can compare the inertial track with the averaged GNSS track to get an indication if the inertial track has a turn or scale bias. The study of these two filters (MM and TFM) are left for future.

1) Hybrid Filter, HF

The hybrid filter implements the predict, update and resample methods. It consists of three separate filters. The IO_Kalman_filter (Fig. 8) is the inertial only, 15 error state Kalman filter [18], which uses only the measurements from the inertial measurement unit. The states are attitude, velocity and position with gyro and accelerometer biases. The Particle filter uses this for its propagation. Also, track form matching filter uses the inertial only solution. The particle_filter has 200 particles. This was found adequate to our need for accuracy (meter-level) in [3]. A particle state consists of x and y positions, direction of movement and weight values. The Kalman_filter is a 15 error state Kalman filter using both the inertial and position fixing measurements. The

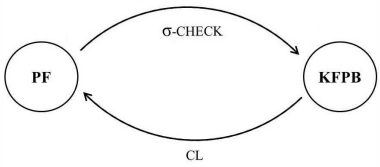

Fig. 9. The state machine for the hybrid filter HF. PF is the particle filter state. KFPB is the Kalman filter with pentagon buffer state. σ-check measures the standard deviation of the particles from the weighted mean of the particles' position and direction of movement. CL is the Collision logic implemented in the map matching (MM) filter.

pentagon_buffer is a five-point error estimation buffer that is around the main solution of the Kalman filter and was also introduced in [3].

Fig. 9 shows the internal state machine for the HF that was implemented previously in [3]. HF has two states noted as PF and KFPB. The filter starts in the PF state, which is the particle filter state. In the beginning, IO_Kalman_filter runs in parallel with particle_filter. The IO_Kalman_filter solution is relative and only follows the changes in the inertial track. Particle_filter solution is absolute and includes the position fixing measurements. When the first positioning fixing measurement arrives, particles are sampled according to the position and the error estimate of the measurement. Afterwards, the σ-check method measures the standard deviation of the particle positions from the weighted solution and the standard deviation of the direction of movement of the particles from the average weighted solution. This check is conducted in the prediction method before predicting. Thresholds define when the state changes. The Kalman filter with the Pentagon buffer is then initialised. The Collision Logic (CL) was implemented in the Map Matcher (MM), in [3], and initiates particle filter back when Kalman filter with the pentagon buffer no longer are able to track and describe the probability of the position.

The particle filter is updated every time either a step or a position trigger is initiated by the context engine (see Fig. 7). The particles are propagated according to the IO_Kalman_filter position change since the last update. System noise is added in the prediction/ propagation stage of the particles. The propagation characteristics were derived in [3]. A modified measurement model was developed for the update stage. This is the Tukey update and is examined next. In addition, the resampling was adjusted to meet the needs of the novel recursive 2-means clustering filter.

a) Tukey update

Usually particle weights are updated using a Gaussian sampling of the positioning fixing measurement using the position and variance measures provided by the sensor. The horizontal accuracy provided by the GNSS receiver was used to define the update distribution standard deviation. Often the uncertainty information provided is not large enough or trustworthy, especially in urban environments with multipath.

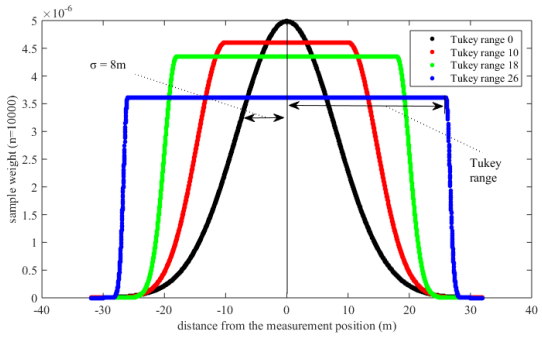

Fig. 10. The Tukey style distribution with different Tukey ranges and a fixed original Gaussian standard deviation value 8m. The Blue figure shows the situation where the Tukey range is close to the 3-sigma value of the standard deviation.

For these situations a Tukey measurement update was developed. Fig. 10 shows multiple Tukey distributions and the usual Gaussian distribution. The Tukey range defines an area around the main measured position where the weights of the particles are weighed equally. The larger the Tukey range the more the main solution follows the inertial measurements since the weight is equal inside the Tukey range. We used Tukey ranges around 10m in our tests. A GNSS receiver horizontal accuracy measure was also adjusted by multiplying it by a factor 4 to cover the GNSS track error.

b) Resampling approach

Usually a measure of the particle weights is followed (effective sample size) and a resampling is made if a threshold value is exceeded. This is to prevent impoverishment.

Instead, we resampled a percentage of particles at each particle filter update. A percentage of particles is resampled according to how much time has passed since the last update. We used 2%/s value in our tests, which was found to behave adequately both indoors and outdoors. If the user stays still the time counting does not proceed.

This approach lets us dynamically change the setting of particle resampling. If the context would require it would be possible to initiate, more or less of the particles to be resampled or to resample percentage of particles initiated by a developed context inference method in the context engine. For example, only particles that collide into walls during positioning fixing gap could be resampled. Moreover, the particle resampling takes into account, into which sub-groups the particle belongs to. The particles are resampled according to the subgroup they belong to. The recursive 2-means clustering filter necessitates this approach. The recursive 2-means clustering filter is introduced next.

2) Recursive 2-means clustering, SC

The recursive 2-means clustering filter runs with the particle filter of the HF. It consists of an extra state for each particle. This field is the groups numbers. SC groups particles

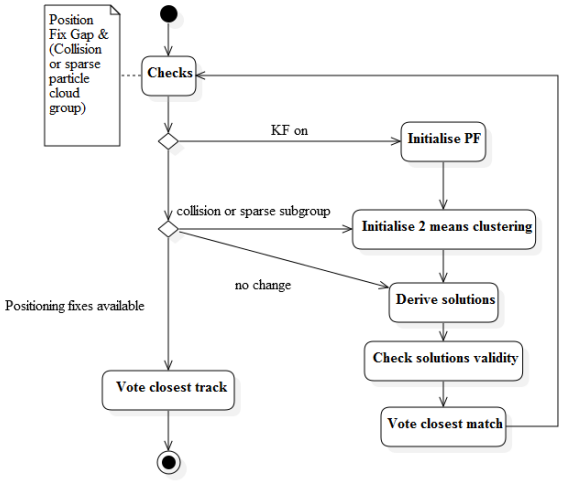

Fig. 11. The activity diagram of the recursive 2-means clustering filter over its life cycle.

into groups according to the sparseness and collisions of the particle clouds.

In our implementation, SC is turned on when the position fix gap trigger is on (see Fig. 7) and when the particle cloud has become too sparse (threshold for particle deviation). Also, if the position fix gap trigger is on and the main solution collides, the SC is turned on. Fig. 11 shows the activity diagram for the life cycle of the recursive 2-means clustering filter. In short, it is created when a positioning fix gap appears, and particle cloud is too sparse or when an unresolved collision happens. Multiple solution hypotheses are then followed. The SC is terminated when positioning fixes reappear. The closest track of the multiple hypotheses of the SC filter is voted as the correct track solution. The operation is followed in detail in the evaluation section with an example.

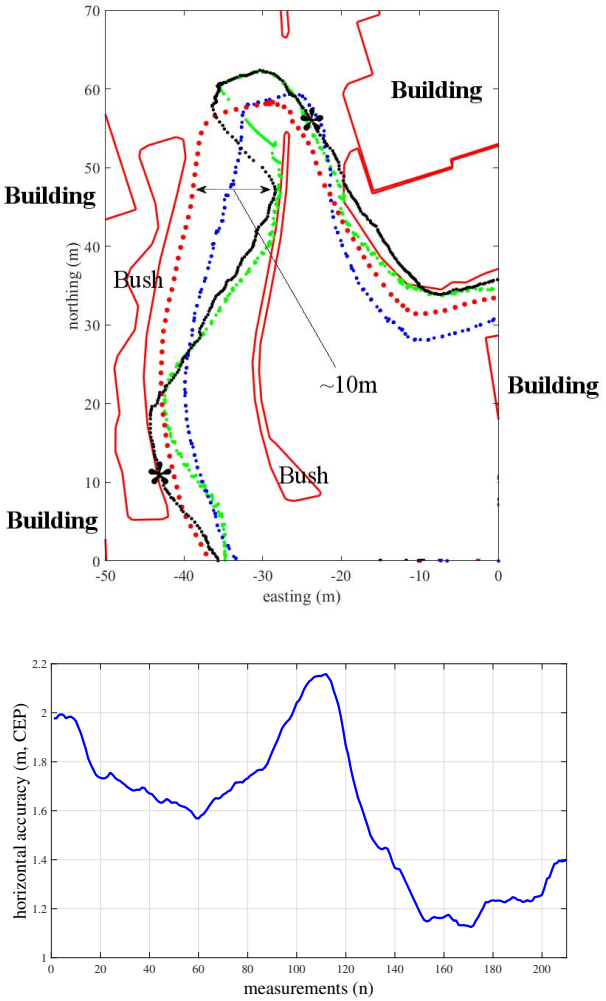

Fig. 12. Above GNSS is black, red is reference. The green track is fusion before. The blue track is fusion after. Below is the horizontal accuracy, used for the measurement update noise estimation, recorded between the two black asterisks in the track figure above. Tukey update gives the inertial solution more influence, seen in the blue track.

IV. Evaluation

The modular design enables dynamic use of sensors and methods in a navigation application. Additional sensors and methods can be added or removed on-the-fly. The system achieves seamless behaviour in our test cases conducted at the campus. The accuracy of the implementation using the system design improves by 5% in our test case.

A. Tukey particle filter measurement update

Tukey update behavior is examined. An example is shown in Fig. 12 of an unreliable position and error estimate of a GNSS positioning fixing system (in black). Fig. 13 shows an image of the two buildings on the east side. A normal Gaussian measurement update was used for the particles weight update resulting in the green trail. The measurements are recorded in an urban canyon with buildings that easily create multipath conditions for the GNSS receivers. The receiver error estimate below can be seen increasing until the user arrives at the north side in the image. The true error can be seen growing from the reference track in red. At the north side the receiver quickly corrects the position and the error estimate. Before this, the error estimate is slightly too small compared to the real error.

The Tukey update preserves particle weights, relative to the GNSS measurements. The Tukey range, together with the standard deviation value, define the update distribution (see Fig. 10) and so, the new weight for the particles. The blue trail shows a change in the main solution track when the multipath disturbance occurs and when inertial measurement is given more influence via the Tukey update. Tukey range was set to 10 metres and measurement deviation to be four times the horizontal accuracy value provided by the receiver. The challenge in the determination of the true quality for both the positioning fixes and the inertial track still remains.

The Tukey update method improves the final solution by 5%. In the Fig. 12 the solution before and after the disturbance follows the GNSS more loosely and could be weighted more at these places. If the inertial solution is of a bad quality, Tukey approach does not help. The key in using the looser update method would be to enhance the quality of the error estimation method of both the GNSS and the inertial tracks. For a GNSS receiver inside a mobile phone and when in urban canyon, the resulting track often looks like the track in Fig. 12. In these cases, Tukey range approach offers additional option for the

Fig. 13. The area is shown from south to north.

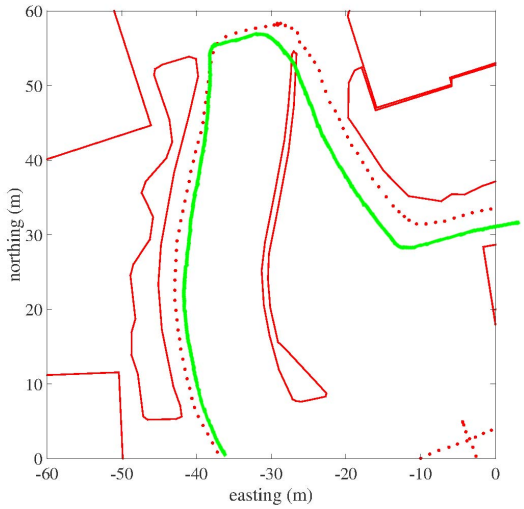

Fig. 14. The foot-mounted inertial dead-reckoning trail (green) deviates approximately 3% from the reference track (red) during the walk.

fused solution to achieve better accuracy when used with foot-mounted inertial unit.

Fig. 14 shows the inertial only track for the case depicted in the Fig. 12. We can say that the foot mounted inertial dead-reckoning is of a good quality. The deviation error percentage for the inertial track in this case is approximately 3%. This proves that for the test case the fusion solution should prefer the inertial track over the GNSS track.

B. Recursive 2-means clustering

Two simple simulated scenarios were created to test the recursive 2-means clustering filter. The results for these scenarios can be seen in the Fig. 15 and Fig. 16. Fig. 15 depicts the particle cloud sparseness while Fig. 16 shows the collision scenario.

As Fig. 11 defines, when there is a gap in positioning fixes and the particle cloud is too sparse, the recursive 2-means clustering filter is initialised. Fig. 15 shows the pentagon buffer around the Kalman filter solution. If the HF is in KFPB state, instead of sparseness of particles the distances to the buffer points are followed. The buffer points are the 3 sigma points of the main solution as described in [3]. The buffer points propagate away from the main solution if no positioning fixes are present. The recursive 2-means clustering changes the state of the HF from KFPB into PF when the buffer sparseness becomes too big. This means a particle filter is initialised. Recursive 2-means clustering has a threshold value for sparseness, the pentagon buffer side and or the particle cloud standard deviation length. If the current error of the filter has become too sparse the recursive 2-means clustering is initialized.

When particles have only one group, the main solution of the particle filter and the previous main solution make up a divisive line. Particles are grouped into two groups according to this line. After this, if the individual groups' solutions become too sparse, the division process is made again. This time the division is based on a line between the individual group solution and the individual groups' previous solution. The solution for the main group is followed as well as the solutions for the two new divided groups. Fig. 15 shows the splitting line for the particles due to sparseness.

The particles are resampled inside their group as stated previously. The resampling happens either when colliding with a wall or when particle weight is within the percentage threshold that is to be resampled. If all of the particles of one group are killed that is they bump into walls, they are resampled into the remaining groups according to the particle weights. Fig. 15 depicts two cases, the particle cloud sparseness and the buffer point distance to the main solution being over the threshold. The drawn pentagon buffer and the particles are from different simulation runs using different settings.

Fig. 16 depicts the collision scenario. This time, a collision initiates the recursive 2-means clustering when there is a gap in the positioning fixing measurements. The main particle group and each subgroup has a solution. This is the weighted average position of each group's particles. In addition, if the group collided, two additional solutions exist. These are solutions

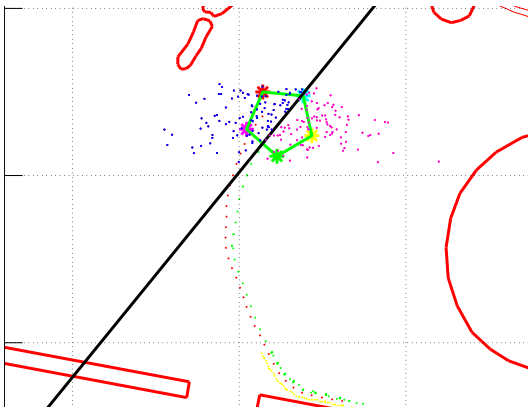

Fig. 15. The recursive 2-means clustering after sparseness threshold has been crossed (blue and magenta particles). The red track is the reference track. The main solution trail is in green. The yellow track is the GNSS track with a gap start.

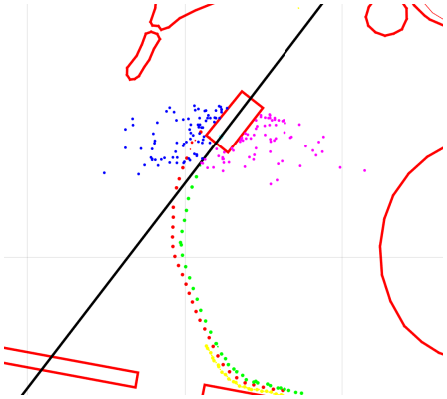

Fig. 16. The recursive 2-means clustering after collision. When the main solution collides during positioning fix gap, recursive 2-means clustering is initialised. The cloud is divided into two according to the main solution direction of movement.

between the colliding main solution of the parent group and the divided two new groups. Two offset values for the two new divided groups are initialised at the collided parent group solution. These two offset solutions are then moved out of the walls towards the new divided groups' solutions. This is shown on the top in Fig. 17. Five solutions now exist.

The validity of the main particle filter solution is invalid as it collided. Thus, the four remaining hypotheses will be followed. The closest solution to the previous position will be voted each cycle. In the end, when the gap ends, the closest track is selected to the new available positioning fixing measurement. This is shown at the bottom of Fig. 17.

V. Discussion

For the first design, the aim was to retain the modularity and replaceability of the system blocks. This was achieved. The version in this paper is designed for loose coupled scenarios. Deeper integration version could be the next step. The next version of the software could also allow methods to be executed without fixed order.

The current design covers the single user case. An extension study could be made for multiple simultaneous co-operating targets. Multiple target tracking requires sufficient hardware capabilities. Computation of each target will be faster when parts of the load is distributed. If capable central processing was available, the target nodes could deliver lower level information (from individual positioning solution to raw sensor data) to the central node releasing their own computation power. Use cases could cover for example raw sensor data or individual positions from a squad of soldiers or firemen. The state in the context engine, presented in this work, could include all targets, enabling flexible cooperative positioning.

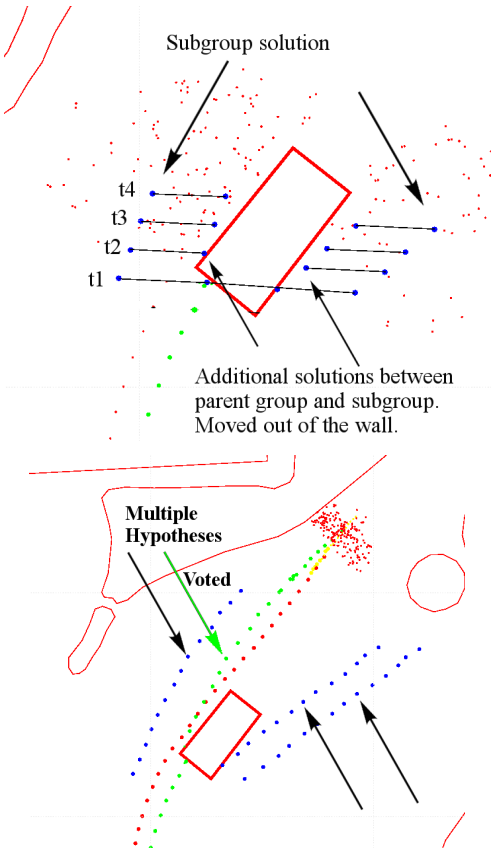

Fig. 17. Above four time instants of the 4 solutions the collision initiated 2-means clustering filter has derived. The main particle filter solution turns invalid after the collision. The solutions of the two particle subgroups are shown at the far ends. In between, we see two more solutions. These are between the main particle filter solution and the sub groups' solutions, removed out of the wall obstacle. Below the closest track is selected when the positioning fixing system again provides position measurements. This is shown in green.

VI. Conclusions

A general design framework for navigation applications was presented, enhancing the adaptive performance and robustness of navigation systems. The tripartite structure enables dynamic solution derivation using the most fit methods for the current context. The accuracy of the system implementation was enhanced by 5% in our test case.

The design enables dynamic addition and removal of filters, context inference methods and sensor subsystems. This increases the seamlessness of the design. The design implementation consisted of four sensor subsystems, context derivation methods and of a bank of filters. The design allows adding single particle or Kalman filters. The hybrid filter could be replaced with these separate filter versions.

The state was implemented inside the context engine. Separating this into a separate object could enable more flexible information exchange and computing design implementation options.

The hybrid navigation filter was presented that has two internal states. The computation is saved in the Kalman filter mode, while particle filter offers the robust alternative for more complex situations. Additionally, a recursive 2-means clustering filter was presented. Recursive 2-means clustering was developed for situations with positioning fixing gaps. Moreover, the Tukey style update was presented as a new way to adjust the particle filter fusion of positioning fixes together with resampling. Simple tests show the design in operation. Enhancing the inference processes in the collision logic and the position fixing and dead reckoning subsystems quality inspection, would further improve the final accuracy and robustness of the design.

The recursive 2-means clustering enables real-time multiple track hypothesis evaluation. This approach could be modified to help in cases where positioning fixes are available, but not essentially of good quality, as is the case with BLE.

Acknowledgement

This work was financially supported by EU FP7 Marie Curie Initial Training Network MULTI-POS (Multi-technology Positioning Professionals) under grant nr. 316528 and the Project TARSIUS of the Spanish Ministry of Economy, Industry and Competitiveness (ref. TIN2015-71564-C4-2-R).

REFERENCES

[1] A. R. Jiménez, F. Zampella and F. Seco, "Light-Matching: a new Signal of Opportunity for Pedestrian Indoor Navigation", 4th Indoor Positioning and Indoor Navigation Conference (IPIN 2013), Montbeliard, France, October 28–31st (2013).

[2] P. D. Groves, L. Wang, D. Walter, H. Martin, K. Voutsis and Z. Jiang, "The four key challenges of advanced multisensor navigation and positioning", 2014 IEEE/ION Position, Location and Navigation Symposium - PLANS 2014, Monterey, CA, 2014, pp. 773-792.

[3] P. Peltola, C. Hill and T. Moore, "Adaptive Real-Time Dual-Mode Filter Design for Seamless Pedestrian Navigation". International Conference on Localization and GNSS (ICL-GNSS), 2015.

[4] A. Basiri, P. Peltola, P. Figueiredo e Silva, E. S. Lohan, T. Moore and C. Hill, "Indoor positioning technology assessment using analytic hierarchy process for pedestrian navigation services," *2015 International Conference on Location and GNSS (ICL-GNSS)*, Gothenburg, 2015, pp. 1-6.

[5] K. Rein and J. Biermann, "Your high-level information is my low-level data - A new look at terminology for multi-level fusion," Proceedings of the 16th International Conference on Information Fusion, Istanbul, 2013, pp. 412-417.

[6] F. Seco, A. R. Jiménez and X. Zheng, "RFID-based centralized cooperative localization in indoor environments", 2016 Indoor Positioning and Indoor Navigation Conference (IPIN), Alcalá de Henares, Spain, October 4-7 (2016).

[7] L. Snidaro, J. Garcia, J. Llinas and E. Blasch, "Context-Enhanced Information Fusion: Boosting Real-World Performance with Domain Knowledge". Springer International Publishing. (2016).

[8] M. Zichar, "Geovisualization-related issues with cognitive aspects", 2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom), Budapest, 2013, pp. 503-508.

[9] D. A. Zaugg, A. A. Samuel, D. E. Waagen and H. A. Schmitt, "A combined particle/Kalman filter for improved tracking of beam aspect targets," IEEE Workshop on Statistical Signal Processing, 2003, 2003, pp. 535-538.

[10] A. Rivero-Rodriguez, H. Leppäkoski and R. Piché, "Semantic labeling of places based on phone usage features using supervised learning," 2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), Corpus Christ, TX, 2014, pp. 97-102.

[11] J. Parviainen, "Studies on Sensor Aided Positioning and Context Awareness", Thesis, Tampere University of Technology, 2016. 61 p. (Tampere University of Technology. Publication; Vol. 1408).

[12] A. Perttula, H. Leppäkoski, M. Kirkko-Jaakkola, P. Davidson, J. Collin and J. Takala, "Distributed Indoor Positioning System With Inertial Measurements and Map Matching", in IEEE Transactions on Instrumentation and Measurement, vol. 63, no. 11, pp. 2682-2695, Nov. 2014.

[13] N. Kronenwett, J. Ruppelt and G.F. Trommer, "Motion monitoring based on a finite state machine for precise indoor localization". Gyroscopy and Navigation, (2017), 8, (3), 190-199.

[14] D. Arora, D. Felix and M. McGuire, "Reducing the error in mobile location estimation using robust window functions," 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, BC, 2009, pp. 199-204.

[15] A. Razavi, M. Valkama and E.S. Lohan, "K-means fingerprint clustering for low-complexity floor estimation in indoor mobile localization", In Globecom Workshops (GC Wkshps), 2015 IEEE (pp. 1-7). IEEE.

[16] B. Altintas and T. Serif, "Improving RSS-based indoor positioning algorithm via k-means clustering", in Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless), 11th European (pp. 1-5). VDE.

[17] F. Zampella, A.R. Jiménez, F. Seco, Improving indoor positioning using an efficient Map Matching and an extended motion model, IEEE Transactions on Vehicular Technology, vol. 64, no. 4, pp. 1304-1317 (2015).

[18] Groves, P. D. (2013). Principles of GNSS, inertial, and multisensor integrated navigation systems. Artech house.