Abstract – An new object oriented development suite for data fusion is presented. It is shown how the various issues in the data fusion development like design, implementation, simulation and testing are realised and automated by this development suite. This allows the realisation of high sophisticated data fusion systems as applied in numerous civil and defence areas, like air traffic or satellite orbit control, coastal surveillance, air defence systems or more generally combat or battle management systems. The approach supports the design of multi layer data fusion systems with centralised, decentralised or hybrid architecture integrating numerous different information sources. Besides of the generation of an object oriented data fusion kernel the developer is also supported in rapid prototyping just at the beginning of the developing process. Further the final software and system testing and verification within a simulated and real environment is taken into account. Functionally, the data fusion development described in this paper includes the tracking, classification and identification topic. This enables a complete data fusion development in accordance with different industry or governmental standards.

Keywords: Data fusion, software development, integration & test, object oriented design.

1 Introduction

There are many contribution [3, 8, 9, 10, 12, 13] to the development of data fusion systems, which lead to a variety of concepts based on the experiences with multiple projects [4, 5, 6, 10, 11]. This paper presents a realisation of these concepts through an data fusion development suite, called Fusion++ [1]. The development suite provides a kernel which satisfies the requirements for modern data fusion systems. Further it supports the development process itself, i.e. requirements analysis, design, test and integration.

Other sources of information have also to be integrated into the data fusion process, e.g. tactical data links, whole
sensor networks, e.g. those of air traffic management (ATC), or C² systems. The target spectrum contain space, air, surface or subsurface objects. Furthermore, according to the individual application the communication infrastructures differs significantly. There may be an infrastructure based only on fibre optic cable (LWL), e.g. for maritime or ATC applications or those based on a complex mobile radio network like distributed ground based air defence systems. The aim of the multisensor data fusion described in this paper deals with the

- tracking (i.e. alignment, data association, estimation),
- classification and
- identification topic.

Hence, the data fusion development suite must be able to generate a real-time data fusion system, which consists of all the different algorithms suitable for the concrete application. It has to cover all the diverse information sources, sensor hardware realisations, communication aspects and different target spectrums as mentioned above. Also the individual operational, latency, accuracy, resolution, interoperability, or resources management requirements for data fusion systems has to be taken into account. To due so all the data fusion architectures, i.e. central, hybrid or decentred, must be mapped into the data fusion development suite. This aspects related to the fusion kernel are described in the second chapter.

1.2 Requirements for the Development Process

These data fusion systems are commonly developed in accordance with different industrial and governmental standards, which must be applied to the development process. This process is commonly partitioned into the following steps (e.g. according to ISO/IEC 12207) [4, 14]:

- System requirements analysis & design
- Software requirements analysis
- Software architectural design
- Software detailed design
- Software coding and testing
- Software integration & qualification testing
- System integration & qualification testing

Therefore, a development suite has also to address all these topics simultaneously. Hence system simulation, testing and its evaluation has to be considered. Besides the testing required by the formal development processes also rapid prototyping needs to be supported by simulation and testing environments. This is of substantial importance for the system and software requirements analysis and design aspects.

The presented development suite addresses these topics via a plug-in concept (Figure 2). Hence DLL-files for various testing, i.e. simulation, real data recording and injection and evaluation may be added into the development suite. The topics of the development process, i.e. the usage Fusion++ are discussed in the third chapter. This includes the design, testing and simulation aspects.

2 Object Oriented Data Fusion Kernel

First the data fusion kernel and its environment consisting of middleware and interfaces will be described bellow.

2.1 Middleware

A data fusion system is always integrated into a middleware specific to the application. This middleware provides a real-time database, which offers the input data to the data fusion and receives the data fusion results afterwards. Of course there exists numerous realisations of such a middleware provided by several companies using different mechanisms. Some middlewares works in a sequential manner, i.e. there are fixed loops for reading and writing data through an data fusion system. Other types of middleware are triggered by messages and events, i.e. the data fusion system is informed via a call back mechanisms whenever new relevant data is arrived. The kernels developed by the Fusion++ are modular in the sense, that they are encapsulated with respect to the middleware. So it is easy to implement the fusion kernel.
for various types of middleware of different manufacturers.

2.2 Interfaces

The interfaces are separated from the pure data fusion kernel through the declaration of an abstract interface. Through the fusion kernel generation the necessary input and output interfaces are created through inheritance from this abstract interface. These input and output interface are responsible for the transformation of external data formats into the fusion kernel internal one. For testing supposes it is also possible to implement this interfaces for various simulators or for recordings of real sensor data. The construct of virtual classes applies also for mechanism like recording, error handling etc. Furthermore these interface classes are also usable together with sockets. This allows the simulation of a sensor network on distributed platforms.

2.3 Data Flow

Already the processing of the source data shows the benefit of an object oriented approach for a data fusion system. One simply generates for every sensor of the sensor suite a instance of Sensor, which receives and processes the sensor data according to the individual specifications. Only what is need is a multiplexing function, which distributes the data. The sensor tracks are realised through instances of type SensorTrack while the sensor track containers corresponds to instances of SensorTrackArray. Assuming that a sensor delivers only raw plots the integration of a sensor tracker is also possible. Then the instances of SensorTrack are produced by a sensor tracker according to the sensor specific attributes. The object model is therefore near to the real world environment.

To implement the multisensor tracking each new instance of SensorTrack calls a new instance named FusionTrack or updates an existing one. This instance is responsible for the processing of those sensor tracks, which it has subscribed before. Therefore it is possible to use multiple multisensor trackers in the system which works in parallel. Using this multisensor trackers it is easy to design a complex fusion process through a fusion tree, whose nodes are the different sensor and multisensor trackers.

All sensor and multisensor trackers are configured by an XML file. This XML configuration determines which alignment, data association and estimation methods are used. Therefore it is possible to implement an R³ approach as well as a complicated UKF based AIMM combined with a multidimensional data association method within the fusion tree.

A multisensor tracker contains a container for the fused tracks, called FusionTrackArray, which refers to all tracks which are candidates for track update. The estimation algorithms are always called from the association process while calculating the likelihood matrices (tensors) for the association process whenever new sensor data arrives. Booth a scan oriented or a track while scan approach is supported. The updated fusion tracks calls the kernel and the kernel calls the output interface.

![Figure 4: Dataflow and classes.](image)

2.4 Functionality and Architecture

A approach to design complex data fusion systems is to split the fusion process into multiple fusion nodes [12], [3], [4].

![Figure 5: Fusion node.](image)

Therefore one constructs a directed fusion tree (figure 6), which condenses information more and more until the information suitable for an operator is delivered. The fusion is performed in the different nodes of this tree. Each node contains the functionalities data alignment, data association and state estimation. The structure of this fusion tree depends on different interdependent operational and technical constraints. The input and output data of such a fusion node describes the target states, i.e. position and dynamics and their accuracies for a determined timestamp. This may be done by delivering measurements or prefiltered information (tracks). Several sensors deliver also additional data related to the target attributes, interesting for classification, identification purposes. The required data fusion system performance
has to be mapped into performance requirements for each fusion node.

Figure 6: Fusion tree – An example.

2.4.1 Data alignment

Data alignment contains the topics of data storing, bias corrections and coordinate transformations. Therefore, all coordinate transformations are offered with respect to the Cartesian and polar CCS, NHCS and WGS84 systems. The data storing is realised by template classes for array and containers.

2.4.2 Data association

Data association examinates the relations between sensor or link data (sensor plots, sensor tracks, link tracks) and the objects specified by the application (air vehicle, ships, ...). A sensor or link data may be caused by one or multiple objects, or by the environment (false alarm). On the other side, a object may also be not detected by the sensor for several reasons. The data association generates, evaluates and selects hypothesis about the associations between sensor data and their origins. The choice of a data association algorithms depends on the specific applications. Examples of such algorithms are:

- nearest neighbour,
- multidimensional data association
  - Auction, JVC
  - Lagrange Relaxation
  - Linear Programming or
- JPDA.

Details may be found in [2], [3], [8].

2.4.2.1 Class model for data association

The data association inherits from the type Matrix. It contains a matrix for the correlation weights as a member variable. It is generated outside of the kernel s.t. it is possible to determine the association algorithm by configuration again. The concrete data association implementation happens through inheritance. The association is encapsulated through the class Assignment. This fills the Association with data and interprets the results. The Association needs no knowledge about tracks and kinematics.

For a sensor update (track while scan) or scan (scan oriented) the weight matrix is determined. The method Calculate delivers the association result. The assignment overtakes the interpretation of the result, i.e. the update of a FusionTrack respectively SensorTrack or the generation of a new FusionTrack respectively SensorTrack.

2.4.3 Estimation

This part includes algorithms addressing the kinematics, classification and identification estimation. For the estimation of the target kinematics the

- Alpha-beta filter
- Kalman filter
- Unscented Kalman filter
- IMM, AIMM, Dual IMM [7]

are available. These can be combined with different models for the prediction, projections onto the measurement space and noise models. The parameterisation of the filters is also supported by an GUI.

Besides the kinematics the classification and identification estimation is included. This addresses algorithms working within the

- Bayes theory,
- Dempster Shafer
- Fuzzy set theory or
- Expert Systems.

These concepts may also be combined into more complex algorithms addressing classification and identification topics.
2.4.3.1 Class model for estimation

The kinematics estimations (i.e. filtering) of a track is calculated by an instance of type Filter which is owned by each instance of FusionTrack. The proper implementation of the individual filter algorithms is encapsulated through a virtual class Filter. It is generated on runtime by the FilterFactory, whereas the choice of filter algorithms is based on a XML configuration file.

3 System Design, Simulation and Testing

3.1 System Design with Fusion++

The design of complex data fusion systems is supported by an graphical GUI interface.

For illustration purposes figure 10 shows the design of a data fusion systems consisting of two separated platforms. Each platform carries its own sensor and onboard the platform the sensor data are processed with a sensor tracker. The data of both platforms are fused with a multisensor tracker, which operates on the associated sensor plots founded by the two sensor trackers on the platforms, which perform the surveillance task.

All three trackers, i.e. booth sensor trackers and the multisensor tracker uses an IMM and nearest neighbour data association. Also the parameterisation of the estimation algorithms are supported by special GUIs (figure 11). Therefore, it is possible to exchange the algorithmic specification of the whole system very fast. The resulting configuration is coded within an XML file, as shown in figure 12.
The XML file is passed whenever the fusion kernel is started up. However it is translated into binary information, so that the configured data fusion system is a real-time application. The data fusion kernel itself is in ANSI C++ and therefore available for different operating systems.

3.2 Simulation and testing

For the testing of data fusion systems on has to distinguish between system and software test and integration. Software test is performed using a simulated environment for sensor data generation. Target trajectories are produced by scenario generators or alternatively GPS recordings are used. Sensor simulators produces sensor data out of this trajectories. The aim of the software test is to provide the succeeding system integration & test phase. It especially addresses the following test items:

- Data association functionality
- Estimation (kinematics, classification, identity) accuracy
- Stress tests
- Fusion of contradicting sensor inputs
- Expensive targets (e.g. TBM and missiles).

Mostly one performs Monte Carlo runs to establish the performance and constraints of the alignment, association and estimation. Monte Carlo runs offer a statistical analysis concerning the estimation and association quality, e.g. stability, loss, and switch of tracks. Further issues deal with track management (premature deletion, late initialisation/deletion), and tactical picture issues (segmented, redundant, false and missed tracks).

In the system test phase one uses

- Real sensors and real targets or
- Recordings and data injector tools

instead of sensor simulators and artificially generated trajectories. Recorded loggings are used not only for evaluation but also for tuning of algorithms and technical parameters and regression testing activities. Furthermore not all sensor combinations must be performed with expensive real targets, which helps to reduce the effort. To fulfil this task, it is necessary to have a data injector tool capable to re-inject the recorded sensor data in the system with the same time relations as processed in the live run.

The test objectives are dominated by

- Sensor capabilities
- Natural environment
- Target attributes and dynamics

The real sensor resolution, accuracy, scanning behaviour and internal signal processing respectively tracking may differ from the idealistic assumptions of the previous development phases. Platform performing the surveillance task may be affected by pitch and roll, which must be compensated. The noise in a simulation is generally modelled by Gaussian distributions, which are idealisations of the noise behaviour within the natural environment and real sensors. Measurements may be misaligned by effects which can not completely be considered in the pure simulation approach, like the physical signal propagation. There is also a gap between real and simulated target dynamics and manoeuvrability resulting in difficult correlations of its kinematical components. Information which is closely related to the real targets themselves is commonly used for classification and identification. Real IFF data may be corrupted through superposition, garbling, fruit or targets may be queried by side lobes hampering the precise azimuth measurement [4], [12], [15].

3.2.1 Software test with Fusion++

With the development suite presented in this paper it is possible to generate multitarget scenarios. To due so the target trajectories are created in a first step. Complex target trajectories may be composed of various elementary manoeuvre components, e.g. figure 13 shows such an weaving scenario.

In a second step the system automatically generates the plot respectively track files for this trajectories using the sensor suite defined in the system design step. Figure 10 shows a weaving scenario observed by the two sensor platforms as well as a central multisensor tracker as describe before. Figure 14 is the result of one of the sensor trackers on one platform.

The development suite contains different tools to support the evaluation. This covers topics like tactical display analysis (TDA), track control, sensor data validation, data analysis (graphical and quantitative), configurable track parameter comparison, and data logging filter (single track selection).
3.2.2 TDA

The TDA allows the selection of
- Single sensor tracks and plots
- Raw sensor plots
- Single fusion tracks
- GPS tracks
- CCS and WGS84 view
- Point and bearing track view
- View of heading vector
- Configurable history of trace
- Track position of cursor

Especially the easy selection of tracks through such an GUI is very important for real multi target scenarios where several thousands of targets are involved.

3.2.3 Track Control

A window for track control shows selected tracks and additional information like
- Shows Associations between Tracks
- Selected Tracks are shown on TDA
- Freeze Tracklist during fast update

- Show Details Windows for
  - Characteristics
  - Kinematics (as Graph and Data)
  - Associations

3.2.4 Data View of a Single Track

Besides the graphical visualisation it is also supported to show the data itself in table formats. These may be easily imported into other programs for further analysis:
- Plots in all coordinate systems
- Track positions in all coordinate systems
- Track velocities in all coordinate systems
- Pitch, roll, heading of platforms
- Range rate
- Extended data (IFF, EW,…)

3.2.5 Graph View of a Single Track

For each individual track it is possible to create plots with respect to
- Position (CCS & WGS84)
- Height vs. range
- Range, bearing, elevation, course, speed, altitude, altitude rate, range rate vs. time
- Calculation of mean value & RMS of each parameter
- Shows all associated sensor tracks vs. time

This provides the capability of simultaneous visualization of fusion output against the input data and ground truth. An example of this functionality is given in figure 15.

3.2.6 Data Validation

A data validation tool is able to scan recorded sensor data for inconsistency to sensor specification, missing or redundant data. Results are listed and can be saved as report file.
3.3 System Test with Fusion++

For system test and integration all the functionalities may be used with recorded real data instead of simulated data. This allows the tuning of the system to the real sensor suite and the real environment.

4 Conclusion

It is shown how the data fusion development concepts formulated by different authors can be supported by a development suite such that the relevant industry or governmental standards are satisfied. All the important steps in the development of data fusion systems are covered: prototyping for requirement analysis, architectural and detailed design, implementation respectively coding, test and integration on the software and system level. Hence one is able to produce high sophisticated, validated, object oriented, real-time data fusion systems.

References


