Sergiy Nazarko

EVALUATION OF DATA FUSION METHODS USING KALMAN FILTERING AND TRANSFERABLE BELIEF MODEL

Master's thesis

Mobile computing

28.11.2002

University of Jyväskylä

Department of Mathematical Information Technology

Author: Sergiy Nazarko

Contact information: Roninmäentie 6M 14B, Jyväskylä 40500, FINLAND, nazarko@cc.jyu.fi

Title: Evaluation of data fusion methods using Kalman filtering and transferable belief model

Work: Master's thesis

Pages: 70

Study Program: Mobile computing

Keywords: Kalman filtering, transferable belief model, Dempster-Shafer reasoning, target tracking, target identification

Abstract:

In the research work data fusion algorithms, which can be used for target identification and target tracking were evaluated. The aim of this work was to provide a clear picture about data fusion, particularly in case of target tracking. For doing this tracking tool application was developed as research environment, containing Kalman filter algorithm and transferable belief model implementations. The research process was concluded by validating the implementations using simulation settings found from research literature. Analysis showed the same performance as other research results abd the evaluation was found successful.

ACKNOWLEDGEMENTS

I have no doubt that this is the most read part of my Thesis, which I write with particular inspiration. There are a lot of people, whom I'd like to thank, and list of names might be quite long. So, I will not mention all of them, but anyway some names are listed here.

I'd like to show my gratitude and appreciation to my supervisors: Jarkko Vuori and Mikko Vapa, they are the best supervisors.

I'd like to thank Kharkov National University of Radioelectronics and University of Jyväskylä for the possibility to be a participant of Master's program.

I want to say "thank you" to my mother and father, I felt their support even they were far from me in distance but not in heart.

Finally I want to show my appreciation to Helen Kaykova and Vagan Terziyan, for all their moral support and very useful advices, which have very important meaning for me, and my living in Finland.

University of Jyväskylä, 21.11.2002

TERMS AND ABBREVIATIONS

BPA	Basic Probability Assignment.
Chedar	Originally named as a CHEap Distributed Architecture - P2P platform developed by Cheese Factory project
CLT	Central Limit Theorem
CPT	Conditional probability table
CPD	Conditional probability distribution
DSR	Dempster-Shafer Reasoning
DRC	Disjunctive rule of combination
FMT	Fast Möbius Transformation
GBT	Generalized Bayesian Theorem
JXTA	JXTA technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner
MTST	Maneuvering target statistical tracking, US NAVY tracker which uses Kalman Filter
MSE	Mean Square Error
P2P	Peer-to-Peer – computing paradigm where each node in the network can be at the same time client and server.
TBM	Transferable Belief Model

LIST OF SYMBOLS

$P(h/D)$	Conditional probability (event h conditioned on D)
$m(A)$	Mass function, represents the belief exactly committed to A
$m(\varnothing)$	Mass function of empty set
$Bel(A)$	Belief function, corresponds to an event that set A covers the true event
$m_1 \otimes m_2$	Combining two mass function into one, using Dempster's rule of combination
$\overline{A_y}$	Complementary set of a set A
$\|A\|$	Quantity of elements in set A
H^T	Transposed matrix H
H^{-1}	Inversed matrix H
$E[]$	Statistical expectation

CONTENTS

1 INTRODUCTION

1.1 Preface

Nowadays data fusion is actually a collection of computational methods in the areas of reasoning, estimation, and learning application. The goal application of this work was data fusion algorithms for target tracking. This work was made within Cheese Factory a Peer-to-Peer (P2P) research project located in University of Jyväskylä at Agora Center [CheeseFactory].

Cheese Factory studies peer-to-peer communication and behaviour of peer-to-peer networks concentrating on distributed search of resources and their efficient use. One idea is that results of this work might be further enhanced with P2P network to develop totally decentralized target system [Smirnova 2002]. In the project Chedar - distributed computing and storage system is being developed. Chedar is built upon Sun's JXTA and Java technologies and it enhances JXTA by introducing intelligent searching and connection management features. Java being portable and compatible with existing code produced by Cheese Factory was also selected for building the tracking tool “Eye of Ra”.

1.2 Research problem statement and related work

The problem from which this research was started is an applying data fusion techniques within P2P environment. To get an idea how data fusion techniques work and get an experience evaluation of data fusion techniques should be done.

In this document introduction to basic aspects of data fusion techniques is given, important concepts involved in data fusion are considered. The main consideration is given to data fusion techniques, which are used for target tracking and identification (Kalman Filtering). Research environment was also produced. The aim of this research is evaluation of data fusion techniques. For evaluation purposes tracking tool application called “Eye of Ra” has been designed and implemented. The application contains:

- User interface.
- Kalman Filter.
- Transferable belief model.

After implementation research process was started. Simulation set was selected from research publication to get a clear evaluation picture. Analysis of received results verified the correctness of developed application and the evaluation seemed to be successful.

Of course it’s not the first work, considering data fusion techniques. Ph.D. thesis work of Petri Korpisaari [Korpisaari 2001] was used as a basis. More information on target identification can be found in [SmetsPage]. A good introduction to probability theory and Kalman filtering is also in [Bar-Shalom 1993].

1.3 Structure of the Thesis

This work concentrates on data fusion techniques used for target tracking and identification and is divided into four chapters. Chapter 2 considers basic concepts of data fusion such as uncertainty and imprecision. Problems related to data fusion are considered and explained with introduction to Bayesian theory, transferable belief model, and Kalman filter. Chapter 3 introduces the tracking tool "Eye of Ra". With the features including user interface, Kalman filter, and transferable belief model algorithms. The code of the algorithms and extended outputs of Kalman filter are given in appendices at the end of the thesis. The thesis is concluded in the last chapter (Chapter 4). The chapter also describes the steps used during the research process and reviews the obtained results concluding the evaluation.

2 DATA FUSION

Data fusion is one of the branches of applied mathematics. Data fusion technology combines different pieces of information into some new compatible information (for example, sundial – combining direction of sun and compass to get information about time) or more accurate data (in case of several radars, more radars provide more information). Application of data fusion methods varies a lot from military applications such as target tracking, target recognition and identification, and situation assessment, to non-military data fusion applications for example machine vision application, robotics medical applications and so on. The data fusion itself includes many other disciplines – estimation theory, statistics and probability, decision-making theory, and many, depending of the area of applying. Data fusion is a formal framework in which the means and tools for the alliance of data originating from different sources are expressed.

Multisensor data fusion includes integration of data from multiple sensors, producing different data with different level of precision, the data being sometimes even conflicting. Furthermore data might be corrupted, so it can be said that problems of data fusion are nontrivial.

Data fusion can be viewed as a set of problems, which it tries to solve:

1) Fusion of temporal information – time always causes problem to engineers, because it deals with dynamics and arises two added problems: effective description of past, and fusion of past data with current situation.

2) Fusion of information from different sources – for example if we have one clock it's easy to describe our knowledge about time, but if other clock appears on the scene we're in trouble. How to combine data from both clocks and what's their accuracy becomes a problem.

3) Fusion of dissimilar information – information, which should be fused could be represented in several ways. For example, how to fuse information, which is represented via dissimilar alphabets?

2.1 Data fusion concepts

Before discussing data fusion techniques I will introduce some basic concepts, which are important from the point of view of data fusion.

There are two opposite concepts used in data fusion: knowledge and ignorance. Representation of what we know is actually representation of what we don't know. So lack of knowledge is actually ignorance, that consists of two forms:

a) Uncertainty – lack of knowledge that originates from the domain of interest, and which cannot be predicted and may be caused e.g. from inadequate model etc.

   Let's look through forms of uncertainty:

   - Incompleteness – sensors are possible to leave something out.
   - Imprecision – sensor can give only approximate information.
   - Inconsistency – sensors data may not always agree with each other.
   - Ambiguity – data from various sensors may be indistinguishable from one another.

b) Imprecision – lack of knowledge, which originates from the weak vocabulary. In case of radars it means that radars can have different level of accuracy.

Of course difference between uncertainty and impression seems to be not very precise, and often these two concepts are “fused” together.

2.2 Data fusion alphabets

In nature information doesn’t consist in any physical form [Korpisaari 2001]. To describe information we use alphabets, which helps us to represent information we have. Several concepts related to alphabets are: singletons, sets of random variables and fuzzy sets.

Singleton is atom of information we can use. Singletons do not contain any lack of knowledge, but it is a basis for describing other representation forms (such as sets), and the unknowing truth can be viewed as unknown singleton.

Singletons form sets, which can represent continuous-valued variables (in these cases sets have forms of intervals and consist of infinite quantity of singletons). Sets can be used to represent ignorance. Sets and singletons can be used when we cannot assign precise singletons to our knowledge. This concept is very important in case of target identification, in case when the radars, which give us information about target are not ideal (in real life there is no ideal radar).

For describing continuous-valued random variables probability density functions are used (the case when singletons are represented as interval). To represent discrete valued random variables probability mass function is used (in case of sets). Finding of appropriate probability density function or probability mass function is not trivial task. Nowadays there exists a lot of parameterised probability mass functions and the most known of them is Gaussian function, which has very nice mathematical properties (product of two Gaussian functions is still a Gaussian function), also well known Central Limit Theorem (CLT) - the mean (or sum) of a number of independent, identically distributed random variables will tend to be normally distributed, regardless of their distribution, if the number of different random variables is large enough [CLT]. Actually many of processes in real-life can be described with this function.

The next concept involved in data fusion is fuzzy logic [Yan 1995]. Fuzzy logic is a powerful problem-solving methodology with lots of applications dealing with control and information processing. Fuzzy logic provides a remarkably simple way to draw definite conclusions from vague, ambiguous or imprecise information. It resembles human decision making with its ability to work from approximate data and to find precise solutions. Unlike classical logic, which requires a deep understanding of a system, exact equations, and precise numeric values, fuzzy logic incorporates an alternative way of thinking. It allows modelling of complex systems using a higher level of abstraction originating from our knowledge and experience. Fuzzy logic for example allows expressing knowledge with subjective concepts such as very hot, bright red, and a long time, which are mapped into exact numeric ranges. Fuzzy logic deals with fuzzy sets, which are defined through the membership function that may have value from 0 to 1.

2.3 Bayesian reasoning and Bayesian networks theorem

Thomas Bayes was an English mathematician and theologican, who developed basic methods of probabilistic reasoning. His theorem is the central theorem of the whole probabilistic theory [Swinburne 2002].

2.3.1 Bayes' theorem

Bayes' theorem provides a way to apply quantitative reasoning to what we normally think of as "the scientific method". When several alternative hypotheses are competing for our belief, we test them by deducing consequences of each one, then conducting experimental tests to observe whether or not those consequences actually occur. If hypothesis predicts that something should occur, and that thing does occur, it strengthens our belief in the truthfulness of the hypothesis. Conversely, an observation that contradicts the prediction would weaker (or destroy) our confidence in the hypothesis.

In many situations, the predictions involve probabilities - one hypothesis might predict that a certain outcome has a 30% chance of occurring, while a competing hypothesis might predict a 50% chance of the same outcome. In these situations, the occurrence or non-occurrence of the outcome would shift our relative degree of believe from one hypothesis toward another. Bayes theorem provides a way to calculate these "degree of belief" adjustments and allows to reverse the conditioning event

Given training data *D*, posteriori probability of a hypothesis *h*, *P(h/D)* follows the Bayes theorem:

$$P(h/D) = \frac{P(D/h)P(h)}{P(D)} \tag{1}$$

To give an idea how Bayes' theorem works, let's look through simple example from [Bayes]:

Example: On the South Side the population is 30% white, 60% black, and 10% hispanic. If the crime rate is 0.015 for whites, 0.04 for blacks, and 0.02 for hispanics and a crime is committed on the South Side, what is the probability that it was committed by hispanic?

The units for the crime rate were not given but it does not matter since we are interested only in relative values. The sum for the three relevant groups is 0.075, so that we may take 15/75, 40/75, and 20/75 for the relative probability that a member of each particular group will commit a crime. The relative probability that a member of a particular group committed a particular crime is this times the probability that a member of that group was on South side. For whites the figure is 0.30 x 15/75 = 0.06. For blacks it is 0.60 x 40/75 = 0.32 and for Hispanics it is 0.10 x 20/75 = 0.0266. Dividing 0.0266 by the sum of these (0.4066), we conclude that the probability that the crime was committed by a Hispanic is 0.065. Of course this analysis ignores any forensic evidence that might be available. Let's look on applying Bayes' theorem for crime committed by Hispanic.

$$P(C/H) = \frac{P(H/C)P(H)}{P(C)} = \frac{(20/75)*0.1}{0.4066} = 0.065$$

Even though Bayesian theorem seems to be suitable for prediction there are some issues that limit its use in data fusion. The weak point of this theorem is that it requires initial knowledge of many probabilities that causes computational costs. Bayesian theorem implementation in data fusion is limited by the inability to depict the level of uncertainty in a particular sensor state [Liu 1992].

2.3.2 Bayesian networks

Graphical models are a marriage between probability theory and graph theory. Probabilistic graphical models are graphs in which nodes represent random variables, and the (lack of) arcs represent conditional independence assumptions. Hence they provide a compact representation of joint probability distributions. There exists two kinds of models: directed graphs (Bayesian networks) and undirected graphs (Markov random fields). Directed models are more popular within artificial intelligence and statistics communities, whereby undirected models are more popular in physics.

Although directed models have a more complicated notion of independence than undirected models, they do have several advantages. The most important is that one can regard an arc from A to B as indicating that A “causes” B. This can be used as a guide to construct the graph structure. In addition, directed models can encode deterministic relationships, and are easier to teach (fit to data).

In addition to the graph structure, the declaration of model parameters is needed. For a directed model, we must specify the Conditional Probability Distribution (CPD) at each node. If the variables are discrete, this can be represented as a Conditional Probability Table (CPT), which lists the probability that a child node takes on each of its different values for each combination of values of its parents. Consider the following famous example [Murphy], in which all nodes are binary, i.e., have two possible values, which are denoted by T (true) and F (false).

P(C=F)	P(C=T)
0.5	0.5

Table 2-1 – Probability for “weather to be cloudy”.

Cloudy	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

Table 2-2 – CPT of “sprinkler to be turned on” condition on “weather to be cloudy”.

Cloudy	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

Table 2-3 – CPT of “to be rainy” condition on weather “to be cloudy”.

Sprinkler	Rain	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Table 2-4 - CPT for “grass is wet” condition on event “Sprinkler” and “Rain”.

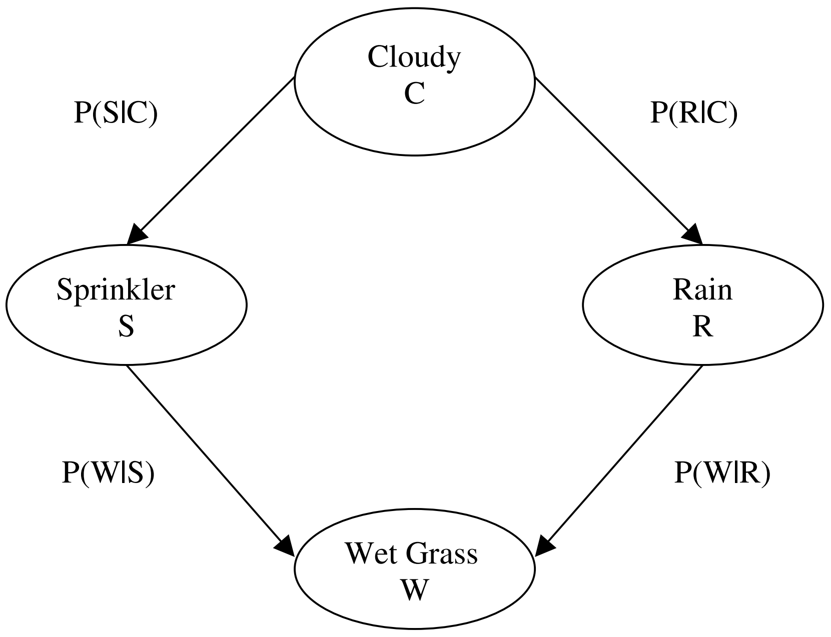


Figure 2-1 - Bayesian network for Murphy's example.

We see that the event "grass is wet" (W=true) has two possible causes: either the water sprinker is on (S=true) or it is raining (R=true). The strength of this relationship is shown in the table. For example, we see that P(W=true | S=true, R=false) = 0.9 (second row), and hence, P(W=false | S=true, R=false) = 1 – 0.9 = 0.1, since each row must sum to one. Since the C node has no parents, its CPT specifies the prior probability that it is cloudy (in this case, 0.5).

By the chain rule of probability, the joint probability of all the nodes in the graph above is

$$P(C,S,R,W) = P(C) * P(S \mid C) * P(R \mid C,S) * P(W \mid C,S,R)$$

By using conditional independency relationships (nodes, which are not connected by a directed arc are conditional independent), we can rewrite the formula this way:

$$P(C,S,R,W) = P(C) * P(S \mid C) * P(R \mid C) * P(W \mid S,R)$$

Now we can see that conditional independence relationships allow us to represent the joint probability more compactly. In this example savings are minimal, but when considering more complicated systems computational advantages are significant.

2.3.3 Inference in Bayesian networks

An usual task to solve using Bayesian networks is probabilistic inference. For example, consider the water sprinkler network, and suppose we observe the fact that the grass is wet. There are two possible causes for this: it is raining, or the sprinkler is on. Which is more likely? We can use Bayes' rule to compute the posterior probability of each explanation (where 0=false and 1=true).

$$P(S=1\mid W=1)=\frac{P(S=1,W=1)}{P(W=1)}=\frac{\Sigma_{c,r}P(C=c,S=1,R=r,W=1)}{P(W=1)}=\frac{0.2781}{0.6471}=0.43$$

$$P(R=1\mid W=1)=\frac{P(R=1,W=1)}{P(W=1)}=\frac{\Sigma_{c,r}P(C=c,S=s,R=1,W=1)}{P(W=1)}=\frac{0.4581}{0.6471}=0.708$$

So we see that it is more likely that the grass is wet because it's raining: the probability ratio is $0.708/0.43=1.65$

In the water sprinkler example, we had evidence of an effect (wet grass), and inferred the most likely cause. This is called diagnostic, or "bottom up", reasoning, since it goes from effects to causes, and also a common task in expert systems. Bayes nets can also be used for causal, or "top down", reasoning. For example, we can compute the probability that the grass will be wet given that it is cloudy. Hence Bayes nets are often called "generative" models, because they specify how causes generate effects.

2.4 Dempster-Shafer theory and transferable belief model

As was mentioned earlier, Bayesian decision theory is limited in its ability to handle uncertainty in sensor data. And this is a big problem in target identification, because sensor data by itself is very uncertain by its nature.

Dempster-Shafer model aims at quantifying degrees of belief [Smets 1993]. Dempster introduced his model in 1967, which was further developed by Shafer in 1976. Dempster-Shafer method allows combining two different mass function into one mass function. Before discussing more closely about Dempster-Shafer theory of evidence, we should look through some concepts in more detail.

2.4.1 Frame of discernment

Dempster-Shafer framework is based on the view where propositions are represented as subsets of a given set. Suppose we are concerned with value of some variable *u* and the set of its possible values is *U*. This set *U* is called as frame of discernment [Ruthven, 1999]. An example of a proposition is “the value of *u* is in A” for some $A \subseteq U$. So as we can see propositions of interest are in one-to-one correspondence with the subsets *U*. In our case frame of discernment is taken to be a set of known target types.

2.4.2 Basic probability assignment

Beliefs can be given to propositions to express their uncertainty. The beliefs are usually computed based on density function $m \to [0,1]$ called mass function or basic probability assignment (in some literature also referred as basic belief assignment):

$$m(\emptyset) = 0 \text{ and } \sum\nolimits_{A \subseteq U} m(A) = 1 \qquad (2)$$

m(A) represent the belief exactly committed to *A*, that is precise evidence that value of *u* is in *A*.

2.4.3 Belief functions

Target identification is the task of solving {attribute, target type} relations [Korpisaari 2001]. The main question is which target types are possible for each attribute? Very often one attribute value belongs to several target types. So, one attribute value should be mapped to a set of target types. For this purpose Dempster used a term *multi-valued mapping*.

It can happen that some of the subsets don't yield any probability at all through the mappings (their basic probability assignment is equal to zero). But it is logical to assign some belief to such set, because subsets have some belief mass. For example, we may assign belief to a set say *Bombers* if several types belonging to this set have some probability mass function, although *m(Bombers)* itself is zero.

Now I will introduce *belief functions*. A belief of given set A is defined as sum of basic probability assignments over all its subsets.

$$Bel(A) = \sum_{B \subseteq A} m(B) \qquad (3)$$

Intuitive meaning of belief function is that it corresponds to an event that set *A* in our case covers the true event. In practice, with *belief function* the *plausibility function* is also used:

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B) \qquad (4)$$

Plausibility function is closely related to *belief function*. When we have one of these functions we can calculate another via for example Möbius transformation, which will be discussed later.

2.4.4 Dempster's rule of combination

As was mentioned earlier Dempster-Shafer techniques combine two different mass functions into one, this is carried by finding consensus between these two functions. Let m_1 and m_2 be the basic probability assignments assigned to two subsets in the frame of discernment *U*. The new set (body of evidence) is defined by basic probability assignment *m* on the same frame *U*:

$$m(A) = m_1 \otimes m_2 = \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{\sum_{B \cap C \neq \emptyset} m_1(B) m_2(C)} \qquad (5)$$

2.4.5 Transferable belief model

Dempster-Shafer model makes a *closed-world assumption*, so it assigns a belief of empty set to zero. Of course, zero belief of an empty set is justified. The reasoning model assumes completeness of the frame of discernment meaning that frame includes all possible hypothesis. But it can very well happen that some hypothesis, because of measurements, are excluded from frame of discernment or unknown. In this way meaning of empty set is changed corresponding not only for impossibilities but also for unknown possibilities. This kind of approach is called *open-world assumption*. The difference between *open-world assumption* and *closed-world assumption* is that belief of empty set isn't assigned to be zero.

Transferable belief model (TBM) deals with conditional mass function. Conditional mass functions correspond to conditional probability functions, which are used in Bayesian networks. The difference is that conditional mass functions describe the belief over sets whereas conditional probability function defines belief over all elements.

The TBM has been developed to provide a model for representation of quantified beliefs. It is based on belief functions, and it corresponds to an interpretation of the model developed by Shafer. These are the concepts used in TBM:

1. Open-world assumption acknowledging that the frame of discernment might not be exhaustive, and implies that positive mass can be assigned to empty set.

2. Generalized Bayesian theorem.

3. Separation of credal level where beliefs are held, and pignistic level where beliefs are used for making decision.

4. The least commitment principles, that justifies the selection of the belief function that doesn't create unjustified beliefs.

2.4.6 Disjunctive rule of combination

The disjunctive rule of combination allows forward propagation when some evidence can be assigned to subsets rather than to single element. There is an assumption that conditional mass functions from single element hypothesis of a variable *A* to subsets of attribute are known. It can be represented through conditional probability table expanded to table with basic probability assignments by using disjunctive rule of combination. The main equations for disjunctive rule of combination are:

$$Bel(B_y \mid A_x) = \prod_{a_i \in A_x} Bel(B_y \mid a_i) \tag{6}$$

$$Pl(B_y \mid A_x) = 1 - \prod_{a_i \in A_x} (1 - Pl(B_y \mid a_i)) \tag{7}$$

Table with belief and plausibility functions can be defined through Fast Möbius transformation. Let's look through a simple example. Assuming we have a conditional probability table as follows:

	a1 (Bomber)	a2 (Fighter)	a3 (Carrier)
b1	0.8	0	0
b2	0	0.9	0.8
b3	0.2	0.1	0.2

Table 2-5 – Conditional probability table (target types a1, a2, a3 condition on attributes b1, b2, b3).

	{a1}	{a2}	{a1,a2}	{a3}	{a1,a3}	{a2,a3}	{a1,a2,a3}
{b1}	0.8						
{b2}		0.9		0.8		0.72	
{b1,b2}			0.72		0.64		0.576
{b3}	0.2	0.1	0.02	0.2	0.04	0.02	0.004
{b1,b3}			0.08		0.16		0.016
{b2,b3}			0.18		0.16	0.26	0.198
{b1,b2,b3}							0.208

Table 2-6 - Basic probability assignments received from Table 2-5 with disjunctive rule of combination.

2.4.7 Fast Möbius transformation

Now it's time to discuss more closely about Fast Möbius transformation. Table of basic probability assignments, belief functions and plausibility functions are very closely related to each other. When we have one of them the others can be received by using for example Fast Möbius transformation [Smets 2001]. Figure 2-2 illustrates the process of Fast Möbius transformation in case frame of discernment contains 3 elements.

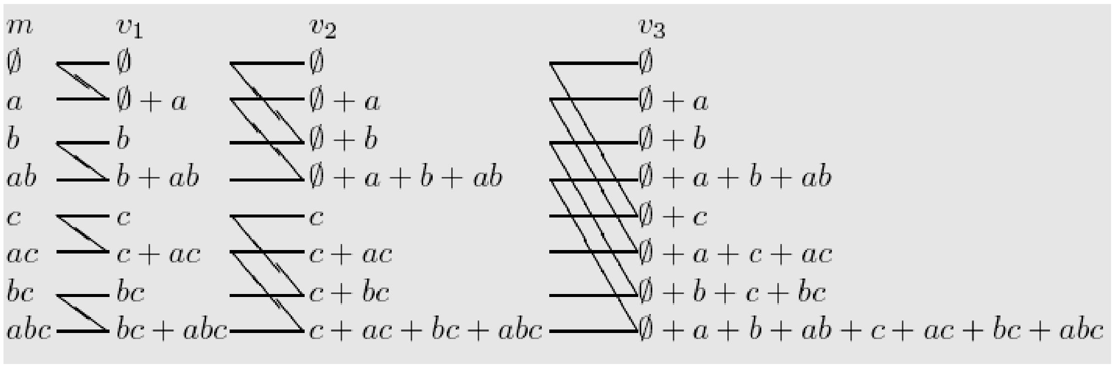


Figure 2-2 - Fast Möbius transformation when frame of discernment contains 3 elements (from basic probability assignments we get belief functions).

Suppose a basic probability assignment (bpa) vector *m* on a three element frame *U*. The column *m* lists the bpa vector. Then we compute a new vector, denoted *v*1. The values of the components of *v*1 are obtained by adding those values of the *m* vectors that are linked to *v*1 by a line. So *v*1(*0*) = *m*(*0*), *v*1(*a*) = *m*(*0*)+*m*(*a*), *v*1(*b*) = *m*(*b*), *v*1(*a, b*) = *m*(*a*)+*m*(*a, b*). The symbols listed in the *v*1 vector indicate the subsets of *m* which masses are included in the *v*1 value. Then we build the vector *v*2 by a similar method, adding the values of the *v*1 vectors that are linked by a line. So *v*2(*a, b*) is obtained by adding *v*1(*a*) and *v*1(*a, b*) hence to masses added up to now in *v*2(*a, b*) are *m*(0) + *m*(*a*) + *m*(*b*) + *m*(*a, b*) as indicated by the labels of *v*2. The *v*3 vector is built similarly. For instance, *v*3(*a, c*) = *v*2(*a*) + *v*2(*a, c*) = *m*(0) + *m*(*a*) + *m*(*c*) + *m*(*a, c*) which is *belief*(*a, c*).

Also plausibility functions can be received the same way. Details can be found in [Kennes 1991].

	{a1}	{a2}	{a1,a2}	{a3}	{a1,a3}	{a2,a3}	{a1,a2,a3}
{b1}	0.8						
{b2}		0.9		0.8		0.72	
{b1,b2}	0.8	0.9	0.72	0.8	0.64	0.72	0.576
{b3}	0.2	0.1	0.02	0.2	0.04	0.02	0.004
{b1,b3}	1	0.1	0.1	0.2	0.2	0.02	0.02
{b2,b3}	0.2	1	0.2	1	0.2	0.1	0.2
{b1,b2,b3}	1	1	1	1	1	1	1

Table 2-7 – Beliefs received from bpa's table applying Fast Möbius transformation.

	{a1}	{a2}	{a1,a2}	{a3}	{a1,a3}	{a2,a3}	{a1,a2,a3}
{b1}	0.8		0.8		0.8		0.8
{b2}		0.9	0.9	0.8	0.8	0.98	0.98
{b1,b2}	0.8	0.9	0.98	0.8	0.96	0.98	0.996
{b3}	0.2	0.1	0.28	0.2	0.36	0.28	0.424
{b1,b3}	1	0.1	1	0.2	1	0.28	1
{b2,b3}	0.2	1	1	1	1	1	1
{b1,b2,b3}	1	1	1	1	1	1	1

Table 2-8 – Plausibility functions corresponding to bpa's table.

2.4.8 Generalized Bayesian theorem

As was mentioned disjunctive rule of combination allows forward propagation, so it can be used for predictive purposes. For propagation of belief from consequences to hypothesis Generalized Bayesian theorem is used. In target identification this theorem plays crucial role because there is always a need for explaining received measurements. The following formulas are defined in Generalized Bayesian theorem:

$$Bel(A_x \mid B_y) = \prod_{a_i \in A_x} (Bel(\overline{B_y} \mid a_i) + m(\varnothing \mid a_i)) - \prod_{a_i \in U_A} (Bel(\overline{B_y} \mid a_i) + m(\varnothing \mid a_i)) \tag{8}$$

$$Pl(A_x \mid B_y) = 1 - \prod_{a_i \in A_x} (1 - Pl(B_y \mid a_i)) \tag{9}$$

For CPT from the previous section we receive next table of bpa's.

	{a1}	{a2}	{a1,a2}	{a3}	{a1,a3}	{a2,a3}	{a1,a2,a3}	{∅}
{b1}	0.8				0		0	0.2
{b2}		0.18		0.08	0	0.72	0	0.02
{b1,b2}	0.016	0.036	0.144	0.016	0.064	0.144	0.576	0.004
{b3}	0.144	0.064	0.016	0.144	0.036	0.016	0.004	0.576
{b1,b3}	0.72	0	0.08	0	0.18	0	0.02	0
{b2,b3}	0	0	0	0	0	0.8	0.2	0
{b1,b2,b3}	0	0	0	0	0	0	1	0

Table 2-9 – Basic probability assignments, which are received from Table 2-5 with Generalized Bayesian theorem.

Again belief and plausibility functions can be received by applying Fast Möbius Transformation.

	{a1}	{a2}	{a1,a2}	{a3}	{a1,a3}	{a2,a3}	{a1,a2,a3}
{b1}	1	0.2	1	0.2	1	0.2	1
{b2}	0.02	0.2	0.2	0.1	0.1	1	1
{b1,b2}	0. 02	0.04	0.2	0.02	0.1	0.2	1
{b3}	0.72	0.64	0.8	0.72	0.9	0.8	1
{b1,b3}	0.72	0	0.8	0	0.9	0	1
{b2,b3}	0	0	0	0	0	0.8	1
{b1,b2,b3}	0	0	0	0	0	0	1

Table 2-10 – Belief functions in case of applying Generalized Bayesian theorem.

Nowadays Dempster-Shafer theory is explored as an alternative to Bayesian probability, because it shows good results, when working with uncertain data. One major advantage of Dempster-Shafer theory is that different sensors can give information, which should be fused in different level of abstraction. It means that every sensor can provide information at its own level of representation. Also hypothesis don’t have to be mutually exclusive and exhaustive, as in Bayesian reasoning. *A priori* knowledge can be represented in varying forms, because sensor data can be represented in different level of abstraction. Dempster-Shafer Theory enables switching from probabilistic methods to logical techniques, when hypothesis become almost entirely true or false [Hughes 1989]

2.4.9 Pignistic transformation

The TBM lies on the assumption that beliefs manifest themselves at two levels: “credal” level and the “pignistic” level (from ‘credo’ I believe and “pignus” a bet, both in Latin) [Smets 2001].

When a decision must be made, beliefs held at the credal level induce the probability in the pignistic level. The transformation between belief function and pignistic function is so-called the pignistic transformation.

When a decision must be made that depends on the actual value of hypothesis h_0 where $h_0 \in H$, probability function in order to select optimal decision might be constructed. The constriction is achieved by the pignistic transformation. $BetP^H$ denote the pignistic probability function used to bet the alternatives in H.

$$BetP^H(h) = \sum \frac{m^H(A)}{|A|(1 - m^H(0))}, \quad \forall h \in H \tag{10}$$

where $|A|$ represents the number of elements of H in A.

In order to have clear picture of what the pignistic transformation is, simple example is presented. Assume that we have the following table of basic belief assignments [Smets 2001]:

H	∅	F	M	B	F, M	F, B	M, B	F, M, B
$m^H[x]$	0.162	0.378	0.108	0.018	0.252	0.041	0.012	0.028
$m^H[y]$	0.16	0.16	0.039	0.24	0.04	0.24	0.06	0.06
$m^H[x,y]$	0.562	0.302	0.048	0.035	0.026	0.019	0.003	0.001

Table 2-11 - Basic belief assignments.

Table 2-12 represent pignistic probabilities computed on H given observed data x, y, and (x, y).

	F	M	B
x	0.638	0.298	0.065
y	0.381	0.131	0.488
(x, y)	0.714	0.163	0.109

Table 2-12 - Pignistic probabilites computed on H.

In order to illustrate how the formula 10 is applied, let's compute $BetP^H[x](F)$

$$BetP^H[x](F) = (0.378 + 0.252/2 + 0.041/2 + 0.028/3)/(1 - 0.162) = 0.638$$

Variables F, M and B represent F-15, Mig-27 and Boeing 747 respectively. Variables x and y mean the opinion we receive by two sensors, in the example these are electronic support measure and radar sensor [Smets 2001].

2.5 Kalman filter

Kalman filter was introduced by R. E. Kalman in his famous work "A new approach to Linear Filtering and Prediction problems" [Kalman 1960]. This filter simply is optimal linear recursive data processing algorithm in least square means [Maybeck 1979]. In this case word filter means a data processing algorithm, and recursive says that this filter doesn't require all previous measured data to be kept in the storage and be reprocessed every time when we have new measurements to input to this system. This plays a crucial role in implementing Kalman filter, which isn't overloaded with often-unnecessary data.

The goal of a Kalman filter is to estimate state of some system from measurements, which may contain random errors. The majority of Kalman Filter applications are related to problem of tracking some target in space. Actually this method is a set of mathematical equations, which provide an effective computational solution of the least square algorithm. All Kalman computations can be thought as some manipulations on the normally distributed probabilities.

2.5.1 Kalman filter foundations

Simple target tracker, based on Kalman filter has an assumption that the target motion can be modelled as a point target moving in a straight line with permanent velocity. In real life, target velocity can change quite often, for example by performing some maneuver, and to allow filter to be used, a noise component is always included in model. If we mark the position and velocity of a target at time t by the state vector $x(t)$, then under a permanent velocity assumption the state of the system at time $t+T$ can be denoted as

$$x(t+T)=\phi(T)x(t)+w(t) \tag{11}$$

where $\phi(t)$ is the state transition matrix for the time interval T, and $w(t)$ is a noise component, which is assumed to be known before filtering and which should be normally distributed. If we describe motion of a target in space with coordinates (x_1, x_2, x_3), then in matrix form the state of the system at time $t+T$ can be shown this way:

$$x(t+T)=\begin{pmatrix} x_1(t+T) \\ x_2(t+T) \\ x_3(t+T) \\ \frac{\partial x_1(t+T)}{\partial(t)} \\ \frac{\partial x_2(t+T)}{\partial(t)} \\ \frac{\partial x_3(t+T)}{\partial(t)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \frac{\partial x_1(t)}{\partial(t)} \\ \frac{\partial x_2(t)}{\partial(t)} \\ \frac{\partial x_3(t)}{\partial(t)} \end{pmatrix} + \begin{pmatrix} w_1(t) \\ w_2(t) \\ w_3(t) \\ \frac{\partial w_1(t)}{\partial(t)} \\ \frac{\partial w_2(t)}{\partial(t)} \\ \frac{\partial w_3(t)}{\partial(t)} \end{pmatrix} \tag{12}$$

Let's denote the measurement vector of the target parameters as $y(t)$, which is vector supposed to be related with state vector $x(t)$ via equation

$$y(t) = Hx(t) + v(t) \tag{13}$$

where H is the measurement matrix and $v(t\,)$ is the measurement noise, which is supposed to be independent from the state noise *w(t)* and which is also normally distributed. In the case when velocity is considered to be permanent in space, the H will be an identity matrix of dimension 6 if both position and velocity are measured. If we receive as an input to our system only the position of some target then H will have the form of 3*6 matrix, like in the following example:

$$y(t)=\begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \frac{\partial x_1(t)}{\partial(t)} \\ \frac{\partial x_2(t)}{\partial(t)} \\ \frac{\partial x_3(t)}{\partial(t)} \end{pmatrix} + \begin{pmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{pmatrix} \tag{14}$$

The target model noise *w(t)* is supposed to have a known covariance matrix

$$Q(t) = E[w(t)w(t)^T], \tag{15}$$

where superscript T says that matrix is transposed and E[] represent statistical expectation.

As a target model noise *w(t)*, the measurement noise *v(t)* is also supposed to have a known covariance matrix:

$$R(t) = E[v(t)v(t)^T] \tag{16}$$

Now we have all the components, so the mathematical model for the simple Kalman filter can be summarised by:

1) State equations: formulas (11) and (12).
2) Measurement equations: formulas (13) and (14).

As was mentioned earlier the Kalman filter processing has as its goal estimation of the state vector $x(t_k)$ in time $t = t_k$ based on the received measurements taken in time $t = t_n$ for $n=0, 1, 2, .., k$. It is a step-by-step process, in the way that each new measurement updates previous estimate $\hat{x}(t_{k-1})$ via computing new measurement information $y(t_k)$ to form new estimate $\hat{x}(t_k)$. This is finished by computing the innovation $z(t_k)$ as the difference between actual measurement $y(t_k)$ and a prediction of it based on previous measurements. μ_w is a movement noise, $w(t) \sim N(\mu_w, Q)$. μ_v is the mean of measurement noise, it means that $v(t) \sim N(\mu_v, R)$. Now we can summarize equations for Kalman filter. For the sake of simplicity the notation $x_k = x(t_k)$ will be used further.

Computing of innovation is denoted as:

$$z_k = y_k - \mu_v - H_k \tilde{x}_k \tag{17}$$

Estimate $\hat{x}_k = \tilde{x}_k + K_k z_k$ (18)

Prediction $\tilde{x}_{k+1} = \phi_k \hat{x}_k + \mu_w$ (19)

Kalman gain $K_k = \tilde{P}_k H_k^T [H_k \tilde{P}_k H_k^T + R_k]^{-1}$ (20)

Estimate covariance $\hat{P}_k = [I - K_k H_k]\tilde{P}_k$ (21)

Prediction covariance $\tilde{P}_{k+1} = \phi_k \hat{P}_k \phi_k^T + Q_k$ (22)

Actually Kalman filter is a predictor corrector algorithm, which is shown in the following picture:

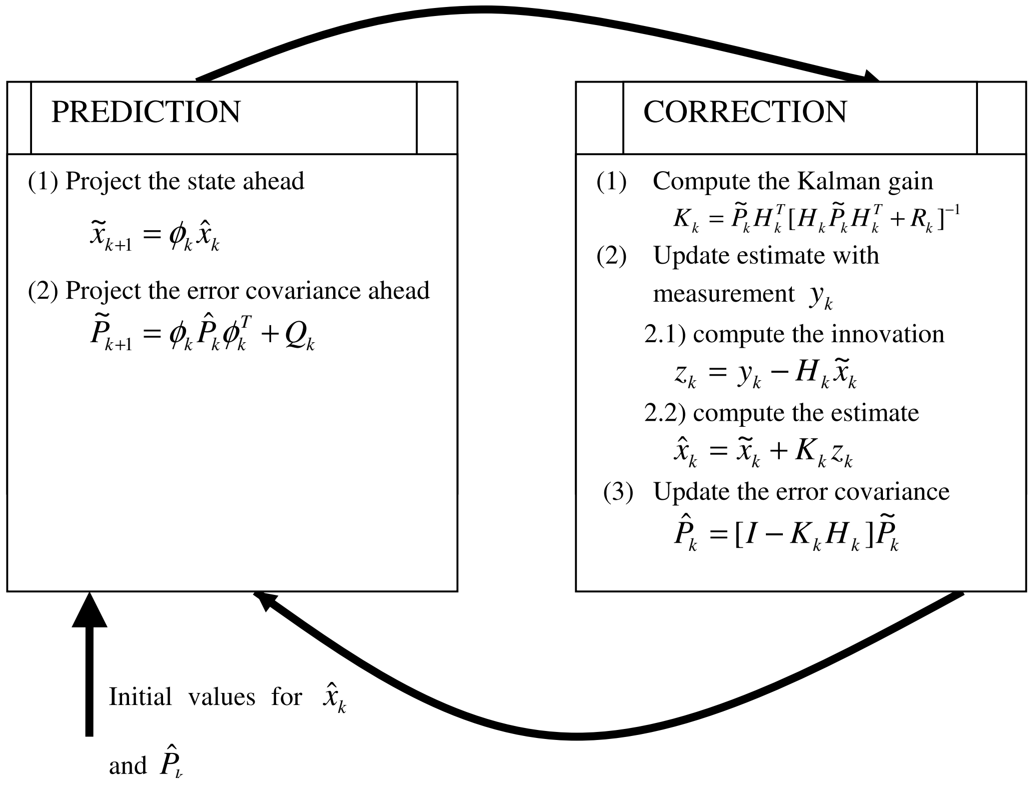

PREDICTION
(1) Project the state ahead
$\tilde{x}_{k+1} = \phi_k \hat{x}_k$
(2) Project the error covariance ahead
$\tilde{P}_{k+1} = \phi_k \hat{P}_k \phi_k^T + Q_k$
CORRECTION
(1) Compute the Kalman gain
$K_k = \tilde{P}_k H_k^T [H_k \tilde{P}_k H_k^T + R_k]^{-1}$
(2) Update estimate with measurement y_k
2.1) compute the innovation
$z_k = y_k - H_k \tilde{x}_k$
2.2) compute the estimate
$\hat{x}_k = \tilde{x}_k + K_k z_k$
(3) Update the error covariance
$\hat{P}_k = [I - K_k H_k]\tilde{P}_k$
Initial values for $\hat{x}_k$ and $\hat{P}_k$

Figure 2-3 - Kalman filter processing.

Now let's consider a simple example. Imagine that we have a very simple target, which can move in one-dimensional random way, and which adds an increment to its position between observations that is normal with mean 1 kilometer and standard deviation 2 kilometers. In other words now our state matrix ϕ is scalar and ϕ=1, Q=4, μ_w=1. The position of the target will increase in time, and the random component will lead to some exception where the state will decrease instead of increasing. Now assume that H=1, μ_v=0 and R=9 kilometers^2 what is to say that unbiased measurements of X are available with accuracies about $\sqrt{R}$ = 3 kilometers (standard deviation). The initial guess of target is $\hat{x}_k$ = 0, $\hat{P}_k$ =10000 kilometers^2, large value of $\hat{P}_k$ says that we basically have no idea where the target is. Let's assume that 3 first measurements are 84, 83, and 88, from these measurements we can assume that target is somewhere in 80's even without help of Kalman filter. Table 1 summarizes applying Kalman filter technique for tracking target from this example.

Step	Prediction		Measurement	Correction		
	$\tilde{x}_{k+1}$	$\tilde{P}_{k+1}$	y_k	K_k	$\hat{x}_k$	$\hat{P}_k$
1	0	1000	84	1	84	9
2	85	13	83	0.59	83.82	5.33
3	84.82	9.33	88	0.51	86.44	
4	87.44	8.58		0.49		4.39

Table 2-13 - Kalman filtering applied for target tracking.

Because of first Kalman gain equals to 1, filter forgets the initial guesses as soon as it gets the first measurement, and then it guesses that target position is $\hat{x}_1$=84, with assumed accuracy as the same as the measurement's accuracy. After that it adds 1-kilometer increment to $\hat{x}_1$ and 4 to $\hat{P}_1$, to receive $\tilde{x}_2$ and $\tilde{P}_2$. It's comes from that the best estimate of target's position before second measurement is increment to its best guess after the first measurement. According to $\tilde{P}_2$ $\tilde{x}_2$ is worse estimate than $\hat{x}_1$, because of outguessed part of the target motion. It is easy to see that $\tilde{P}_{k+1}$, K_k, and $\hat{P}_k$ can be calculated before the measurement, and that is because these variables are independent of measurements.

"Only easy and simple algorithms are effective" might some experienced designers say, so it may be that the simplicity has been the key for a wide success of Kalman filter, not the complexity. Thus, Kalman filter is a tool that allows making optimal estimates according to received measurements and on the basis of mathematical model of the system. Defining the right model for the modeling systems is very important step, which should be performed precisely. Advantages of this algorithm are first of all its recursive nature, which allows to work in real life applications, and also it's ability to estimate *a priori* the accuracy of received results via algorithm itself.

For further reading good tutorials on Kalman filtering can be found in [Banks 1986], [Bar-Shalom 1993], [Brookner 1998], [Catlin 1989].

3 TRACKING TOOL - EYE OF RA

3.1 Overview

For evaluation of data fusion methods research object was implemented. Kalman filter was selected for target tracking, and transferable belief model for target identification. These algorithms were implemented as part of tracking tool “Eye of Ra” using Java programming language.

User interface of “Eye of Ra” for controlling the settings input and displaying the results in case of TBM is shown on the Figure 3-1.

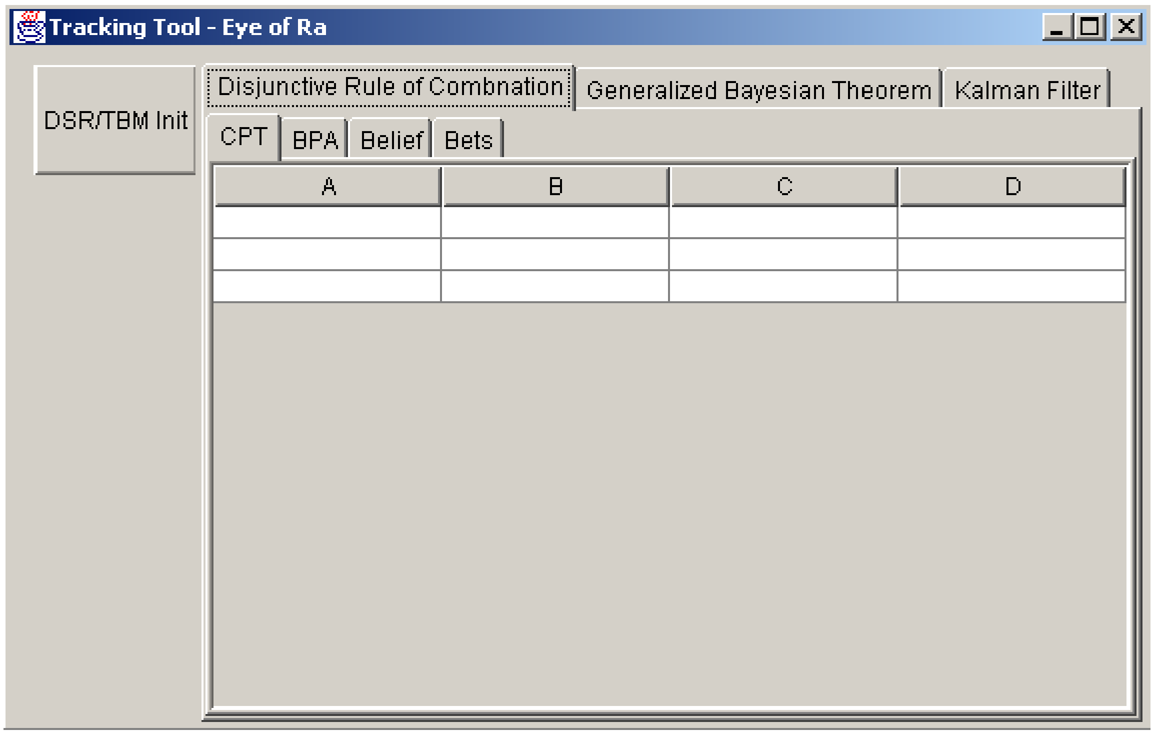

Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets
A
B
C
D

Figure 3-1 - User interface of tracking tool ”Eye of Ra” in case of TBM.

Button “DSR/TBM init” is used for initialization and beginning of TBM algorithms work. Tabbed panes “Disjunctive Rule of Combination” and “Generalized Bayesian Theorem” are used for presentation of calculation results with DRC and GBT respectively. Tabbed panes “CPT”, “BPA”, “Belief” and “Bets” are used for presentation of different steps involved in TBM.

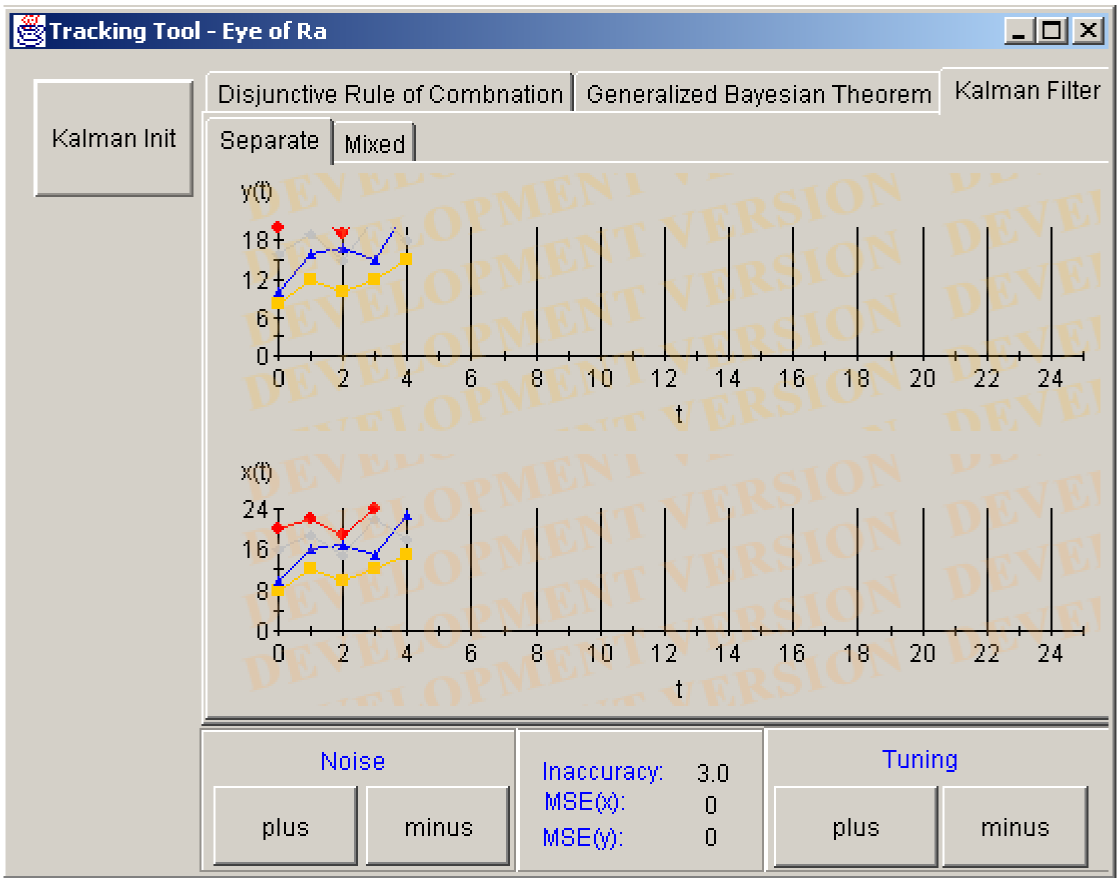


Figure 3-2 - User interface of tracking tool ”Eye of Ra” in case of Kalman filtering.

Button “Kalman init” is used for initialization of inputs and starting Kalman filter works. Four buttons in the bottom of application depicted as “Noise” and “Tuning” are responsible for increasing-decreasing noise parameters (that affect on the accuracy of measurements, represented by label “Inaccuracy”) and tuning covariance matrix of Kalman filter respectively. Tabbed panes are used for representing inputs and results. Labels “MSE(x)” and MSE(y) represent value of Mean Square Error between estimated value and real value on *x* and *y* coordinated respectively.

3.2 Evaluation of transferable belief model

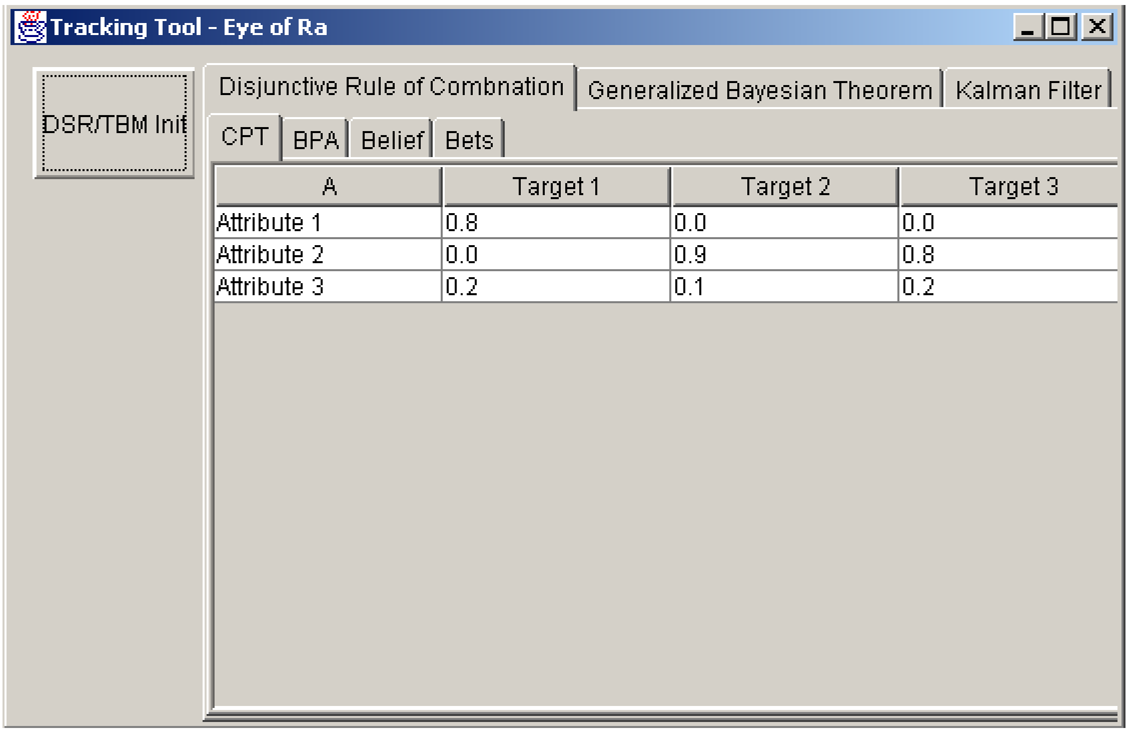
A sensor can be seen as a piece of equipment that observes some data *x* and transmits some ‘opinion’ about the actual value of a parameter of interest *h*. The simplest form arises when *x* and *h* are in one-to-one correspondence. In that case the sensor observes *x* and communicates the corresponding *h*. In more complex cases, the relation between *x* and *h* is not that simple, and the relation between *x* and *h* is represented by a set of probability distributions on *x* given *h*, one distribution for each possible value of *h*.

The idea of using belief functions to represent quantified uncertainty was introduced by Shafer [Shafer]. The TBM accepts that uncertainty is represented by belief functions. From the first point of view it might seems that TBM represents belief in a static way. But it’s not true, for example, TBM can represent how beliefs change:

- When data are collected by several sensors and must be combined (combination rules).
- When beliefs must be inverted in order to transform a belief, for example not only target identification but also data association (generalized Bayesian Theorem).

It might seem odd that sensors would represent their report in form of a “belief”, because the term belief is strongly human oriented. But it just means that sensor has collected some data and according to this data produces its opinion.

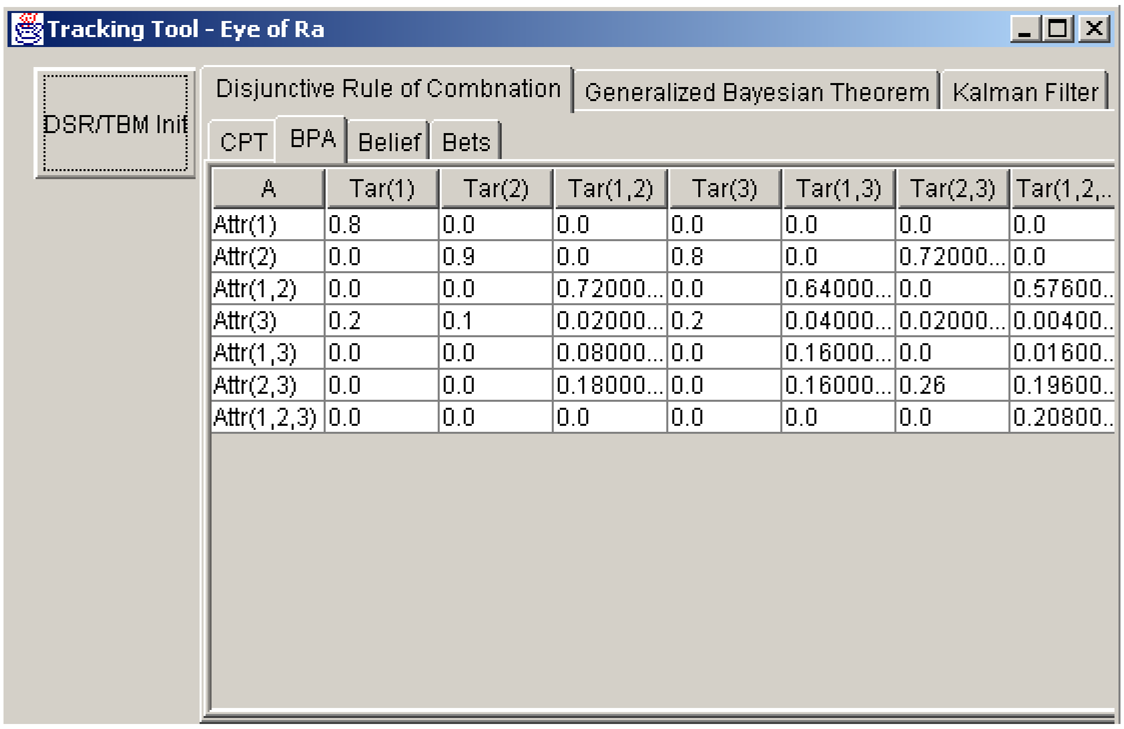
In case when sensors give information not about certain target type but as set of attributes, and probabilities of certain target have certain attribute are known, application that provides calculating involved in TBM has been developed as part “Eye of Ra”. The input for the application is conditional probability table Figure 3-9. The input has been selected to provide results which are easy to handle and track. The same value of conditional probability table was presented in [Korpisaari 2001].



Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

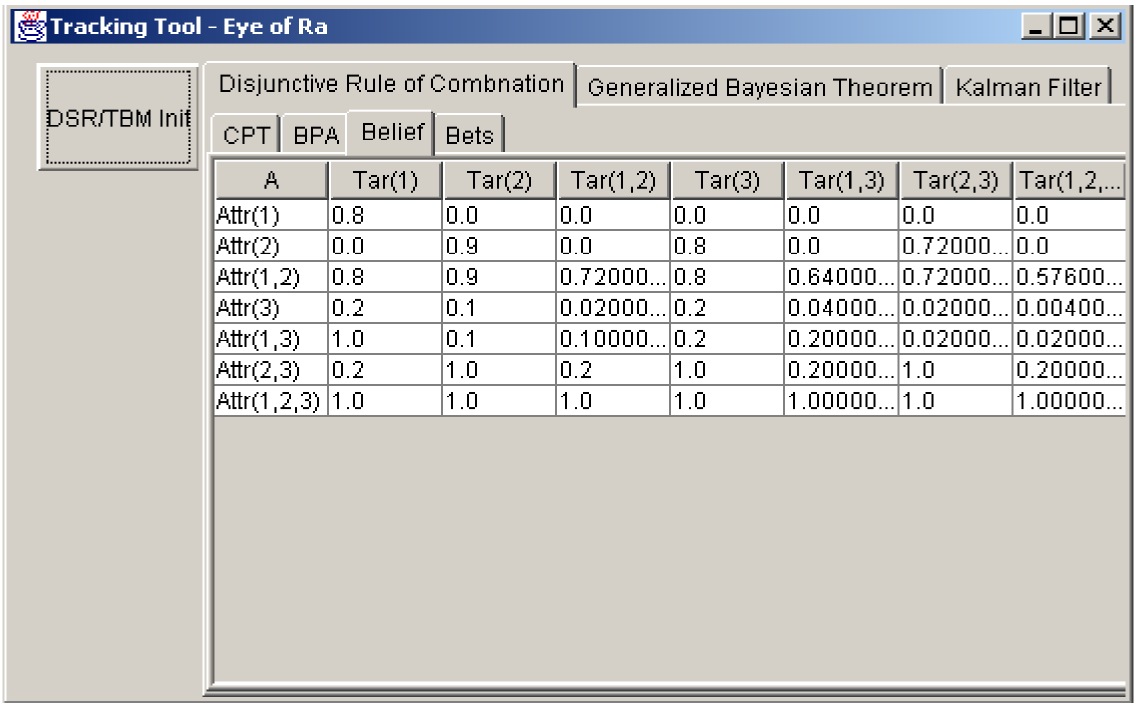
A	Target 1	Target 2	Target 3
Attribute 1	0.8	0.0	0.0
Attribute 2	0.0	0.9	0.8
Attribute 3	0.2	0.1	0.2

Figure 3-3 - Conditional probability table.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

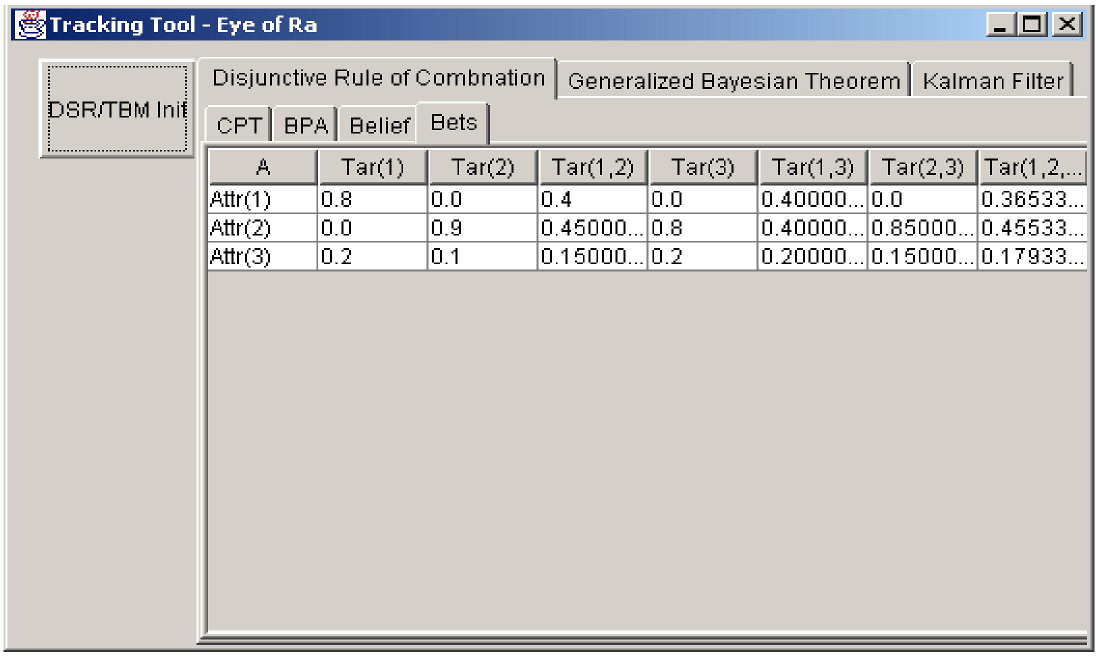
A	Tar(1)	Tar(2)	Tar(1,2)	Tar(3)	Tar(1,3)	Tar(2,3)	Tar(1,2,..
Attr(1)	0.8	0.0	0.0	0.0	0.0	0.0	0.0
Attr(2)	0.0	0.9	0.0	0.8	0.0	0.72000...	0.0
Attr(1,2)	0.0	0.0	0.72000...	0.0	0.64000...	0.0	0.57600..
Attr(3)	0.2	0.1	0.02000...	0.2	0.04000...	0.02000...	0.00400..
Attr(1,3)	0.0	0.0	0.08000...	0.0	0.16000...	0.0	0.01600..
Attr(2,3)	0.0	0.0	0.18000...	0.0	0.16000...	0.26	0.19600..
Attr(1,2,3)	0.0	0.0	0.0	0.0	0.0	0.0	0.20800..

Figure 3-4 - BPAs received from CPT by applying DRC.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

A	Tar(1)	Tar(2)	Tar(1,2)	Tar(3)	Tar(1,3)	Tar(2,3)	Tar(1,2,...
Attr(1)	0.8	0.0	0.0	0.0	0.0	0.0	0.0
Attr(2)	0.0	0.9	0.0	0.8	0.0	0.72000...	0.0
Attr(1,2)	0.8	0.9	0.72000...	0.8	0.64000...	0.72000...	0.57600...
Attr(3)	0.2	0.1	0.02000...	0.2	0.04000...	0.02000...	0.00400...
Attr(1,3)	1.0	0.1	0.10000...	0.2	0.20000...	0.02000...	0.02000...
Attr(2,3)	0.2	1.0	0.2	1.0	0.20000...	1.0	0.20000...
Attr(1,2,3)	1.0	1.0	1.0	1.0	1.00000...	1.0	1.00000...

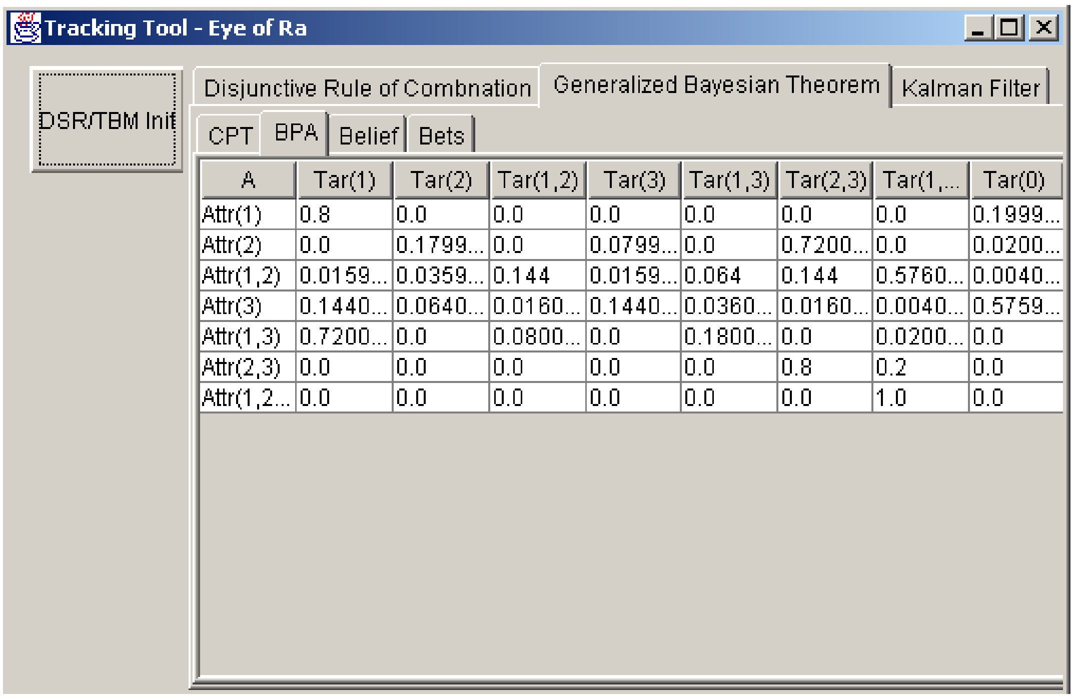
Figure 3-5 - Beliefs received from BPAs with FMT.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

A	Tar(1)	Tar(2)	Tar(1,2)	Tar(3)	Tar(1,3)	Tar(2,3)	Tar(1,2,...
Attr(1)	0.8	0.0	0.4	0.0	0.40000...	0.0	0.36533...
Attr(2)	0.0	0.9	0.45000...	0.8	0.40000...	0.85000...	0.45533...
Attr(3)	0.2	0.1	0.15000...	0.2	0.20000...	0.15000...	0.17933...

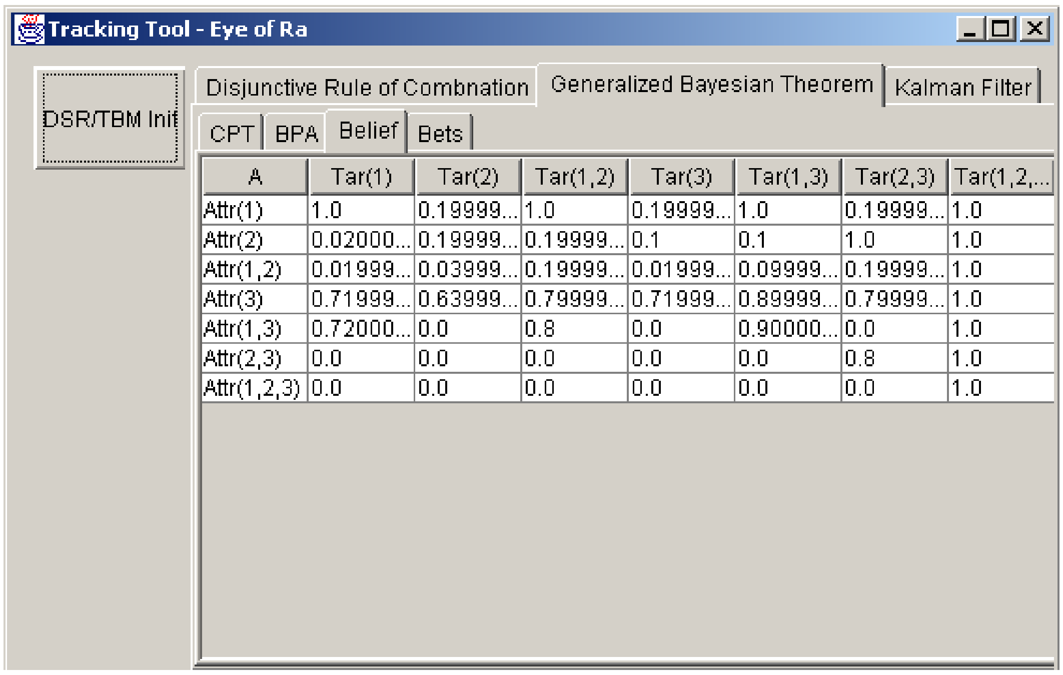
Figure 3-6 - Pignistic transformation, target conditioned on attribute (identification).

As was mentioned earlier generalized Bayesian theorem is used to invert beliefs, that is represented in Figures 3-7 – 3-9.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

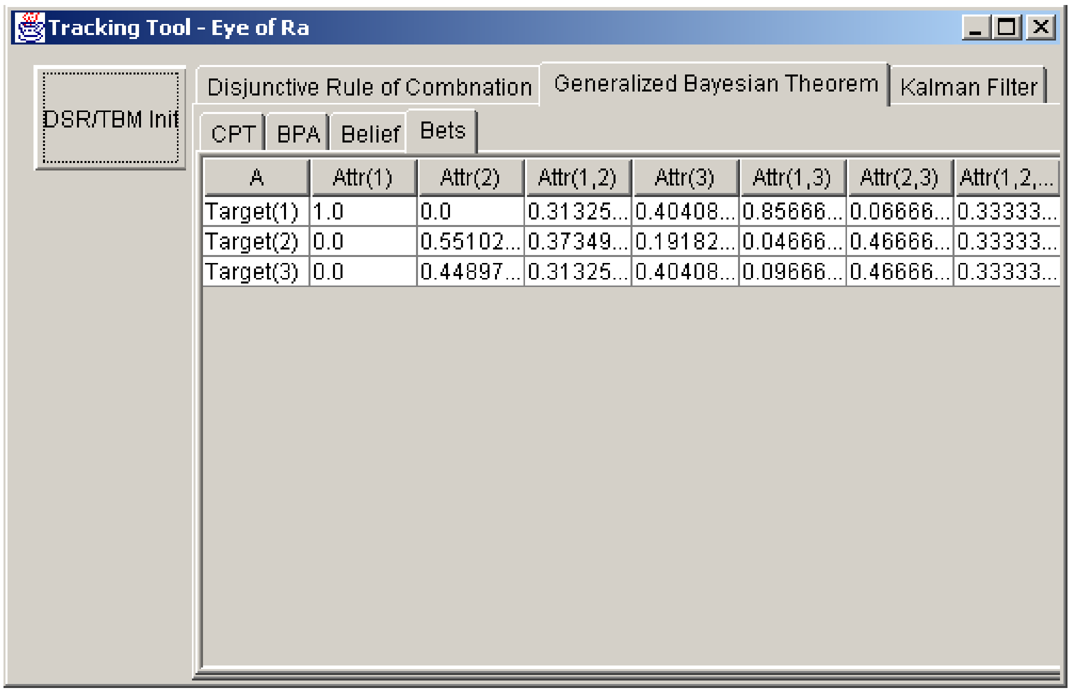
A	Tar(1)	Tar(2)	Tar(1,2)	Tar(3)	Tar(1,3)	Tar(2,3)	Tar(1,...	Tar(0)
Attr(1)	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.1999...
Attr(2)	0.0	0.1799...	0.0	0.0799...	0.0	0.7200...	0.0	0.0200...
Attr(1,2)	0.0159...	0.0359...	0.144	0.0159...	0.064	0.144	0.5760...	0.0040...
Attr(3)	0.1440...	0.0640...	0.0160...	0.1440...	0.0360...	0.0160...	0.0040...	0.5759...
Attr(1,3)	0.7200...	0.0	0.0800...	0.0	0.1800...	0.0	0.0200...	0.0
Attr(2,3)	0.0	0.0	0.0	0.0	0.0	0.8	0.2	0.0
Attr(1,2...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

Figure 3-7 - BPAs received from CPT by applying GBT.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

A	Tar(1)	Tar(2)	Tar(1,2)	Tar(3)	Tar(1,3)	Tar(2,3)	Tar(1,2,...
Attr(1)	1.0	0.19999...	1.0	0.19999...	1.0	0.19999...	1.0
Attr(2)	0.02000...	0.19999...	0.19999...	0.1	0.1	1.0	1.0
Attr(1,2)	0.01999...	0.03999...	0.19999...	0.01999...	0.09999...	0.19999...	1.0
Attr(3)	0.71999...	0.63999...	0.79999...	0.71999...	0.89999...	0.79999...	1.0
Attr(1,3)	0.72000...	0.0	0.8	0.0	0.90000...	0.0	1.0
Attr(2,3)	0.0	0.0	0.0	0.0	0.0	0.8	1.0
Attr(1,2,3)	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 3-8 - Beliefs received from BPAs with FMT.


Tracking Tool - Eye of Ra
DSR/TBM Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
CPT
BPA
Belief
Bets

A	Attr(1)	Attr(2)	Attr(1,2)	Attr(3)	Attr(1,3)	Attr(2,3)	Attr(1,2,...
Target(1)	1.0	0.0	0.31325...	0.40408...	0.85666...	0.06666...	0.33333...
Target(2)	0.0	0.55102...	0.37349...	0.19182...	0.04666...	0.46666...	0.33333...
Target(3)	0.0	0.44897...	0.31325...	0.40408...	0.09666...	0.46666...	0.33333...

Figure 3-9 - Pignistic transformation, attribute conditioned on target (identification).

In other words transferable belief model can be used not only for target identification but also for data association, which also has significant meaning, particularly in case of multitarget tracking.

To make clear picture about pignistic transformation and bets, let's consider column depicted as *Attr(1, 3)* in Figure 3-9. This notation means, that combining two attributes (Attribute1 and Attribute 3) we can assume with probability 0.85 that we track target type 1, with the probability 0.05 that it's target type 2, and with probability 0.1 it is target 3. The picture might become even clearer if we substitute labels "Attr" by label "Sensor", in this case it means that combining pieces of evidence given by different sensors we have got some decisions.

The results achieved during the second simulation are almost similar to results presented in [Korpisaari 2001]. Exception is the calculation of beliefs in case of using generalized Bayesian theorem. It seems that for some reasons the value of empty set $\varnothing$ has not been involved in Korpisaari's calculation. Results achieved during second simulation has been checked using Matlab [Matlab] sources, which can be found in [SmetsPage] and they seem to be correct.

3.3 Evaluation of Kalman filter

At this point it's useful to look at assumption behind the use of Kalman filter, that have significant meaning in the practical application of this filter.

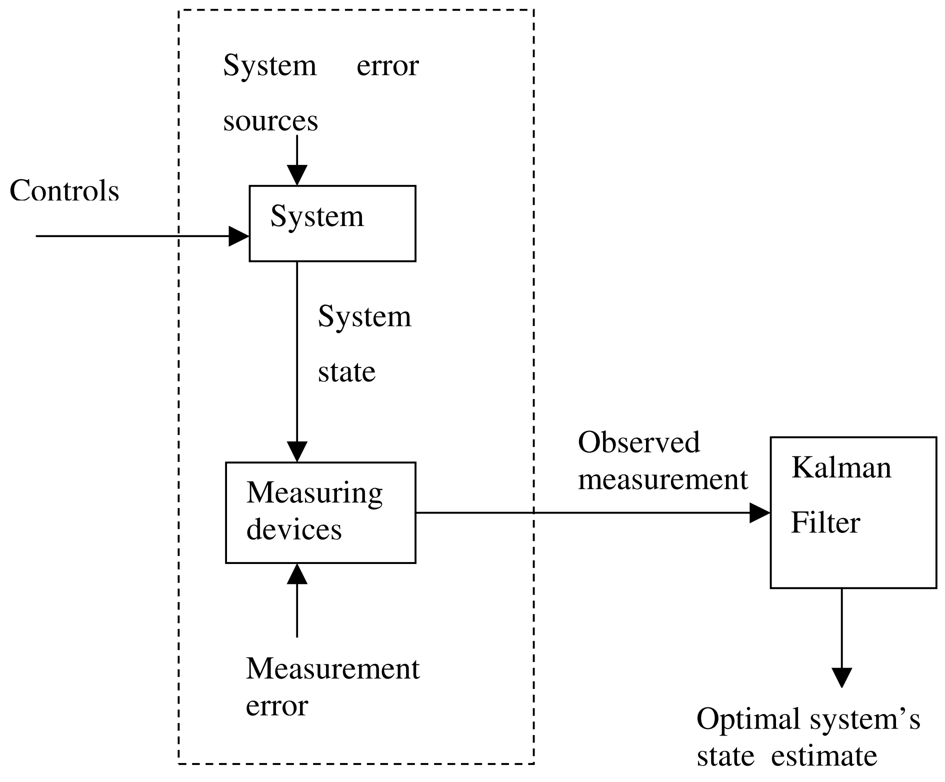


Figure 3-10 - Typical Kalman filter application.

A linear system model is justifiable for number of reasons. Often such models are adequate for many purposes, and even if nonlinearities exist, the typical engineer's approach is to linearize about some nominal point or trajectory. Linear systems are very good in sense that they are more easily manipulated, and linear system (or differential equation) theory is much more complete than nonlinear. There are also means to extend the Kalman filter concept to some nonlinear applications or developing nonlinear filter directly, but these are usually done only in case if linear models are considered inadequate [Maybeck 1979].

Second assumption is “whiteness” it means that the noise value is not correlated in time. More simply, if you know what the value of noise is now it doesn’t give you any advantages to predict noise value in the next time stamp. It also means that the noise has equal power in all frequencies. White noise cannot really exist – it is artificial entity. White noise makes mathematics involved in filter vastly simplified (in fact, made tractable), when we replace the wideband noise with white noise, which from the system point of view is identical. So that is the reason why this model is used.

Last assumption is that noise is represented as Gaussian probability density. The mathematical properties of Gaussian function has been described earlier.

In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. Measuring the measurement error covariance R is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise [Bishop 2002].

The determination of the process noise covariance Q is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainty into the process via the selection of Q. Certainly in this case one would hope that the process measurements are reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often superior filter performance (statistically speaking) can be obtained by tuning the filter parameters Q and R. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to as system identification, it doesn’t refer to target identification in this sense it means we have to identify our system to get better result from applying Kalman filter.

It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting beacons in optoelectronic tracker ceiling panels, the noise in measurements of nearby beacons will be smaller than that in far-away beacons. Also, the process noise Q_k is sometimes changed dynamically during filter operation in order to adjust to different dynamics

When Kalman filter is used for tracking the motion of some entity based on measurements of its position and/or velocity, the applied acceleration may not be known. Under these conditions, one often used techniques to solve such kind of problem is to model excitation as white stochastic random process with zero mean and a normal distribution. This process can be modeled and then included into covariance Q_k. If Kalman filter is applied to the problem of a moving target whose acceleration is unknown this form of noise covariance is often used:

$$Q = \sigma^2 T \begin{pmatrix} \frac{1}{20}T^4 & \frac{1}{8}T^3 & \frac{1}{6}T^{24} \\ \frac{1}{8}T^3 & \frac{1}{3}T^2 & \frac{1}{2}T \\ \frac{1}{6}T^2 & \frac{1}{2}T & 1 \end{pmatrix}$$

If the target is known to be highly maneuverable, then a large value of σ is appropriate. But for targets, which maneuverability is low a small value of σ is appropriate. The derivative of this kind of noise covariance can be found in [Stansfield].

In order to illustrate how the Kalman filter works an application extension for “Eye of Ra” was developed. For the sake of simplicity and in order to get results, which are easier to track, our model had these parameters:

$$x(t+T)=\begin{pmatrix} x_1(t+T) \\ x_2(t+T) \\ \dfrac{\partial x_1(t+T)}{\partial(t)} \\ \dfrac{\partial x_2(t+T)}{\partial(t)} \end{pmatrix}=\begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_1(t) \\ x_2(t) \\ \dfrac{\partial x_1(t)}{\partial(t)} \\ \dfrac{\partial x_2(t)}{\partial(t)} \end{pmatrix}+\begin{pmatrix} w_1(t) \\ w_2(t) \\ \dfrac{\partial w_1(t)}{\partial(t)} \\ \dfrac{\partial w_2(t)}{\partial(t)} \end{pmatrix}$$

Suppose that we don't measure velocity and measure only position in this case our measurement equation is:

$$y(t)=\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} x_1(t) \\ x_1(t) \\ \dfrac{\partial x_1(t)}{\partial(t)} \\ \dfrac{\partial x_2(t)}{\partial(t)} \end{pmatrix}+\begin{pmatrix} v_1(t) \\ v_2(t) \end{pmatrix}$$

Trajectory data was created randomly using Matlab [Matlab]. The data contained one track with thirty (*x, y*) points in order to illustrate how Kalman filter behaves this data was fed as an input to the algorithm. The simulations were made for different kinds of noise, and also for different accuracies of sensors. Scheme of Kalman filter application is shown in Figure 3-3. As input we have true trajectory values which are summarized with noise component. Then these noise corrupted values are transferred to Kalman filter and as output we have estimated trajectory. The default value of inaccuracy is 3 units. The default value of covariance matrix is selected from research literature [Bishop 2002].

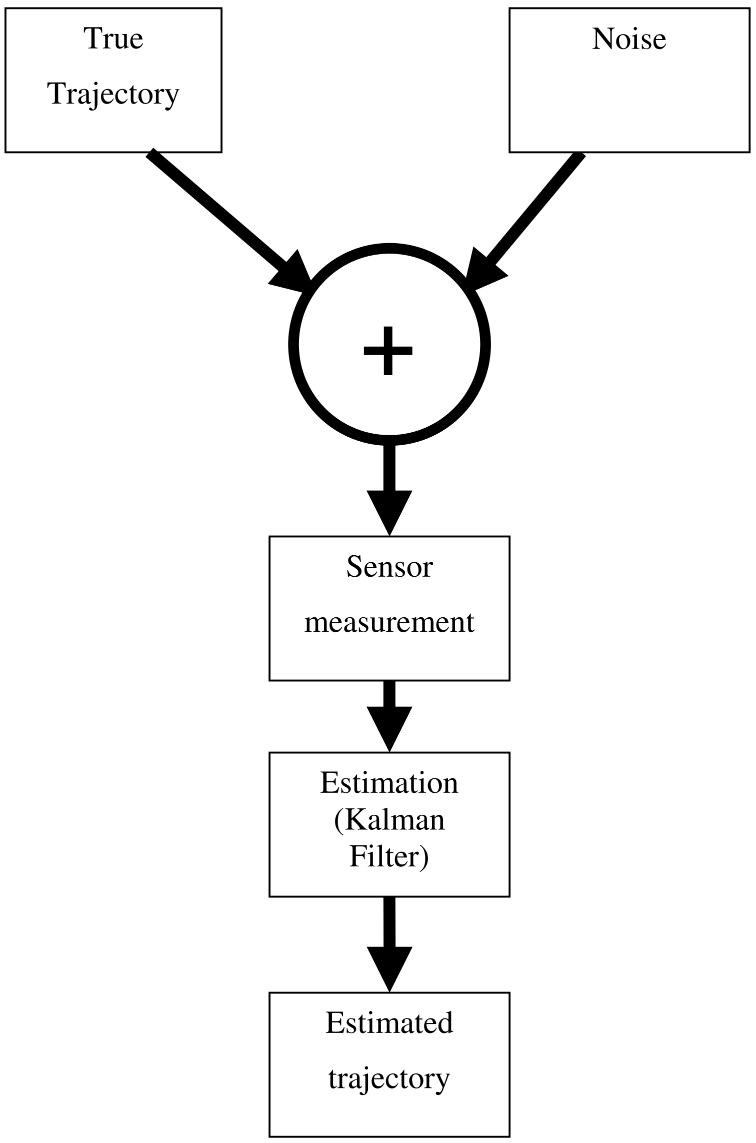
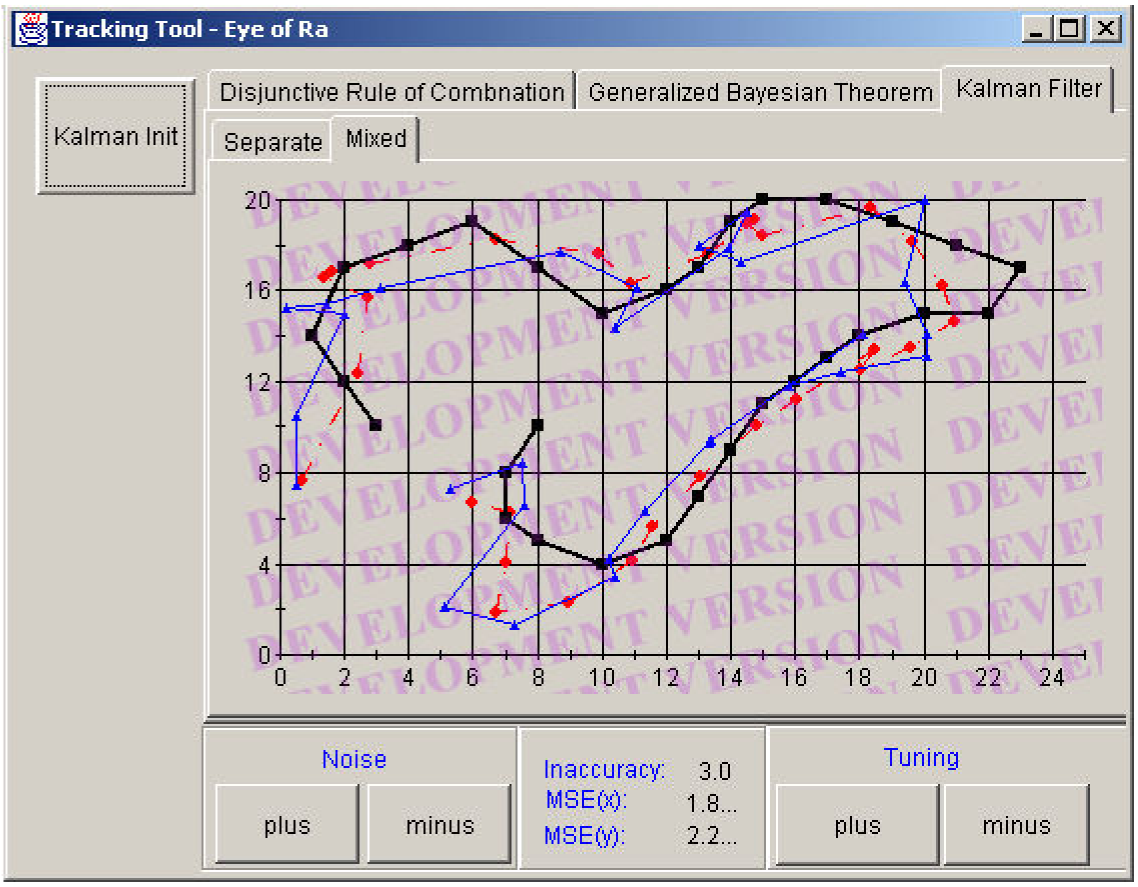


Figure 3-11 - Scheme of developed application.

Tracking procedure using default parameters of Kalman filter depicted on Figure 3-12. Solid bold line corresponds to the true trajectory of the target, normal line depicts the measured trajectory, which is corrupted by noise, and dashed line represents estimated trajectory that has been found applying Kalman filter technique.

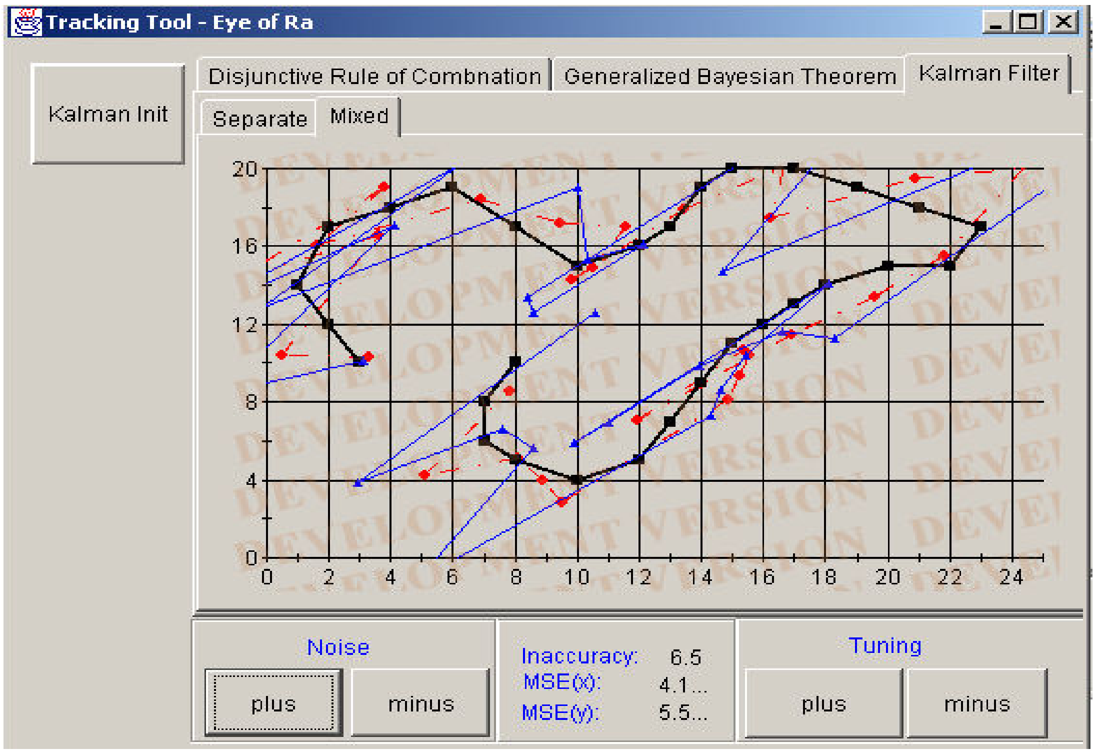


Inaccuracy: 3.0
MSE(x): 1.8...
MSE(y): 2.2...

Figure 3-12 - Tracking procedure using defaults parameters.

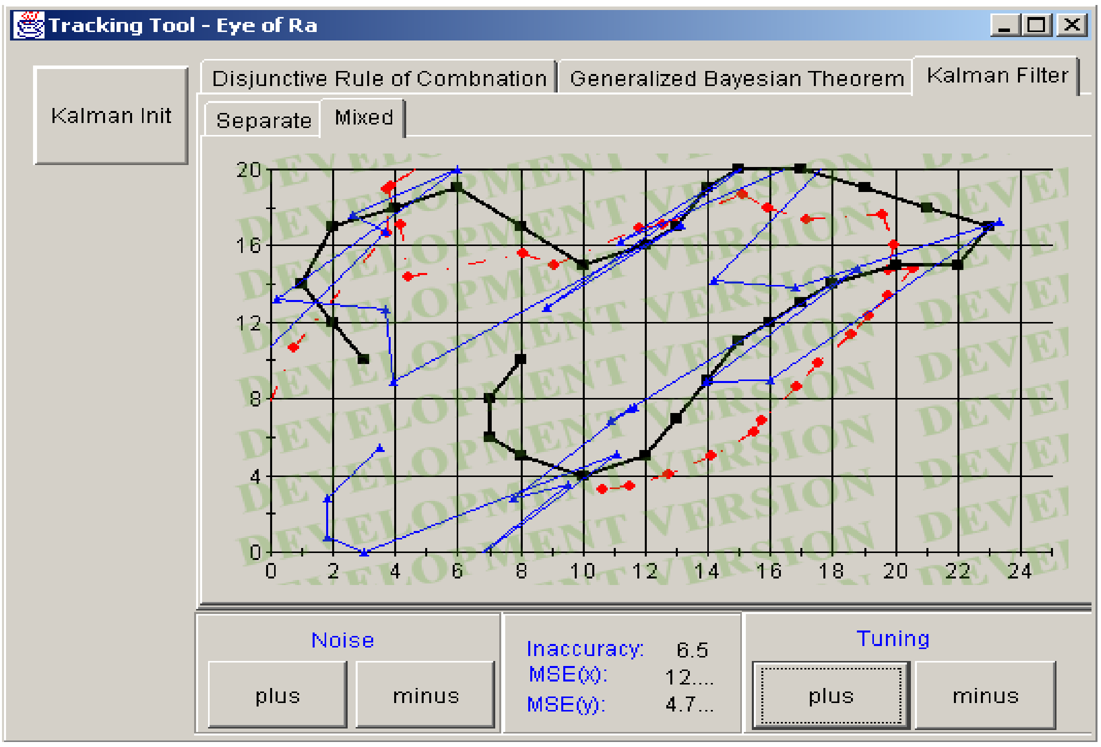
Four buttons which denote as “Noise” and “Tuning” represent the measured data which is corrupted by noise and covariance matrix Q_k. MSE labels represent mean square error between real trajectory and estimated trajectory. Now let’s change the “Noise” parameter. Result is shown on Figure 3-13.

Now we have quite noisy corrupted measurements. In order to get better results even when we have very big uncertainties in measurements we should tune parameters of our filter in our case these are the values in covariance matrix Q_k. In other words increasing noise covariance we can tell the filter that we are not very certain about quality of received measurements.


Tracking Tool - Eye of Ra
Kalman Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
Separate
Mixed
Noise
plus
minus
Tuning
plus
minus

Inaccuracy: 6.5
MSE(x): 4.1...
MSE(y): 5.5...

Figure 3-13 - Noise corrupted measurements (inaccuracy 6.5 units).

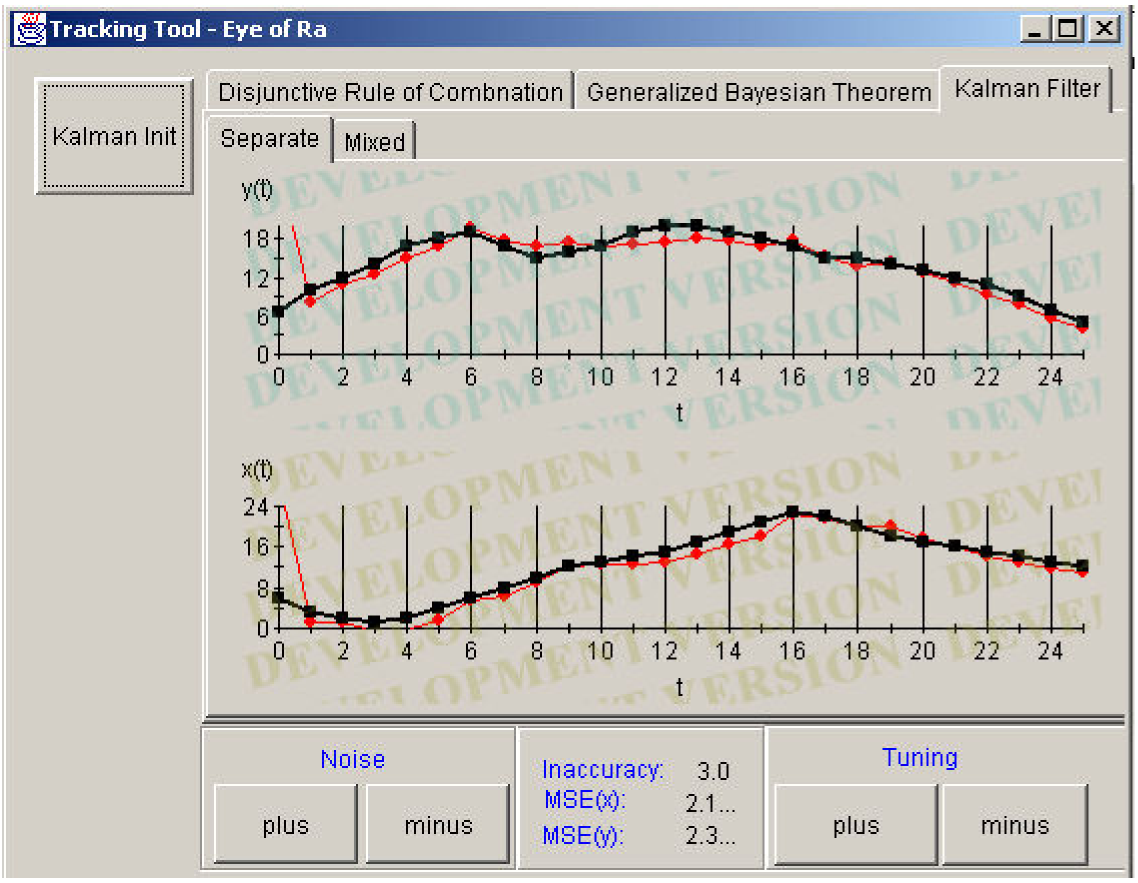

Tracking Tool - Eye of Ra
Kalman Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
Separate
Mixed
Noise
plus
minus
Tuning
plus
minus

Inaccuracy: 6.5
MSE(x): 12....
MSE(y): 4.7...

Figure 3-14 - Noise corrupted measurements, with grater value of noise covariance.

From the first point of view the estimated value seemed not to be very good, but we have to take into account the inaccuracy of measurements because dispersion of 6.5 units of measured values is very big according to scale where we have maximum value of 20 units.

One more thing to mention is bold dots on the lines – they represent the time stamps. According to our model we measure position of target in two coordinates *x* and *y*. So, if we want to represent this with time we need a 3D environment. To solve this problem it is possible to show tracking procedure independently by each coordinate.

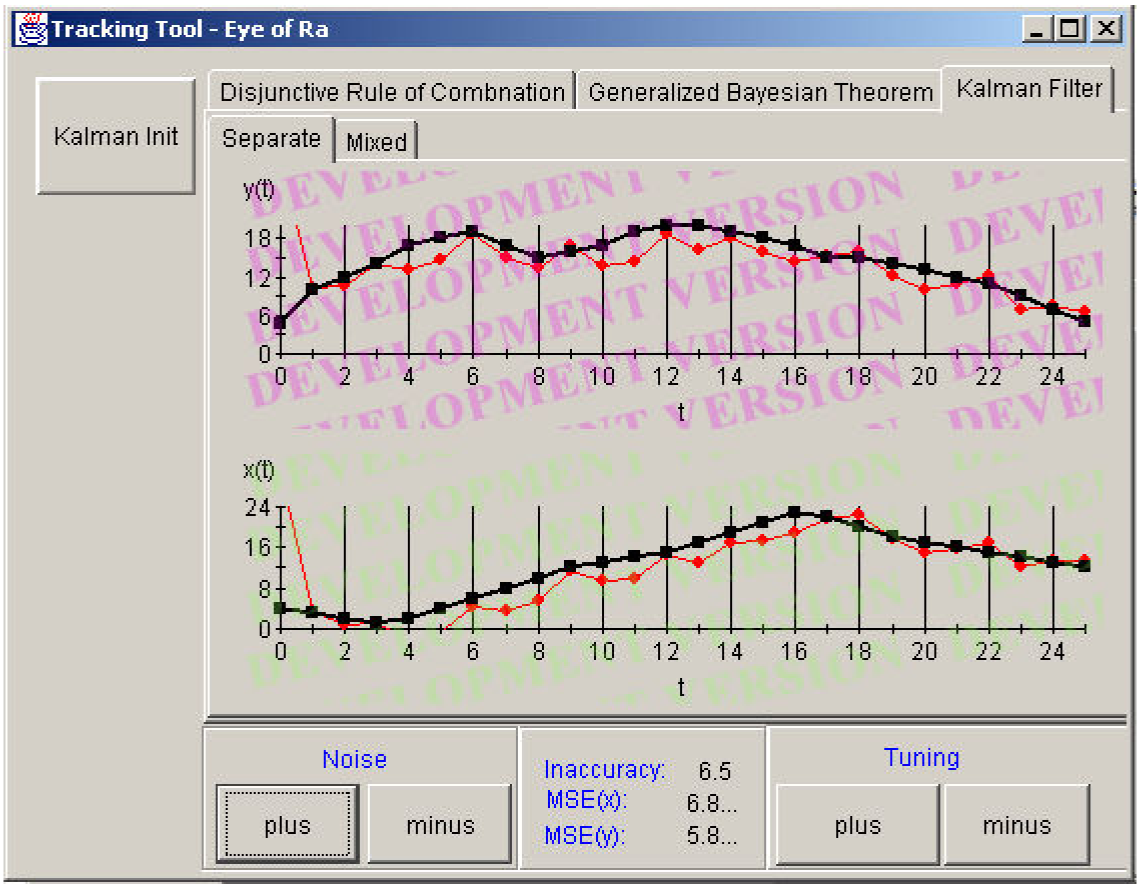

Tracking Tool - Eye of Ra
Kalman Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
Separate
Mixed
y(t)
x(t)
t
Noise
plus
minus
Tuning
plus
minus

Inaccuracy: 3.0
MSE(x): 2.1...
MSE(y): 2.3...

Figure 3-15 - Separate representation of each coordinate with default parameters.

So now it's easier to track how the real trajectory (bold line) differs from estimated trajectory (normal line). Even when we have very noisy corrupted measurements the results seems to be better according to new representation as in Figure 3-16.

The results obtained during this simulation are seemed to be quite similar to results, described in [Bishop 2002]. In their case they made a measurements of voltage and real value of voltage was constant, and they measure only one value. In our case the measured values assumed to have dynamics.


Tracking Tool - Eye of Ra
Kalman Init
Disjunctive Rule of Combnation
Generalized Bayesian Theorem
Kalman Filter
Separate
Mixed
y(t)
x(t)
t
Noise
plus
minus
Tuning
plus
minus

Inaccuracy: 6.5
MSE(x): 6.8...
MSE(y): 5.8...

Figure 3-16 - Separated representation of noisy measurements (accuracy 6.5 units).

Of course for the sake of simplicity, the model simulated in our application is quite simple. In case of real application of Kalman filter every matrix could be somehow transformed during the process of target tracking. For example U.S. Navy makes use of a Kalman filter called the maneuvering target statistical tracking (MTST). MTST has four components state vector, the first two being target location and the last two - target velocity. The two components of velocity are assumed to be independent Ornstein-Uhlenbeck process with the same parameters [Gelb 1988]. They have following state transition matrix:

$$\phi = \begin{bmatrix} 1 & 0 & \delta & 0 \\ 0 & 1 & 0 & \delta \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & c \end{bmatrix}$$

Description and derivative of parameters c and δ can be found in [Gelb 1988].

In practice there exists a lot of filters that can be used for target tracking, for example the Benedict-Bordner filter, which is to say almost identical to Kalman filter. So a question is raised why we should use the Kalman filter. The advantage of Kalman filter comparing the Benedict-Bordner filter is that calculations of the accuracy of filter prediction can be made. This prediction is needed in military application for a weapon deliver system to determine if the predicted position of the target is known accurately enough for a target kill. It is also needed to accurately predict where a detected Subsonic Cruise Unarmed Decoy (SCUD), intermediate-range ballistic missile (IRBM), or intercontinental ballistic missile (ICBM) would land. It is also needed for determining where an artillery shell, mortar shell, SCUD, IRBM or ICBM was launched. The Kalman filter allows making estimates of the launcher location (reason is predictive nature of algorithm), and estimate the accuracy of these estimates [Brookner 1998].

The Kalman filter makes optimal use of the target measurements by adjusting the filter parameters to take into account the accuracy of some measurements. For example if in some measurements the signal-to-noise ratio is very good so that a very accurate target position measurement is obtained, then parameters are automatically adjusted to take this into account.

Also the Kalman filter optimally makes use of a priori information. Such a priori information could come from example from another radar that had been previously tracking the target and from which the target is being handed over. Data from previous radar can be used to optimally setup the filter's parameters for the new radar. Finally the addition of the random-velocity variable, forces the Kalman filter to be always stable [Brookner 1998].

3.4 Summary

Question might arise why so different methods have been implemented to evaluate the data fusion techniques. The answer is simple, in practice target tracking and target identification procedures are often run in parallel. Because knowledge of certain target's type gives the possibility to tune parameters of Kalman filter according to given model. And vice versa motion of some unknown target (for example speed, acceleration) can be used for correct target identification. Effective target identification and target tracking play crucial role in case of multitarget environment and multitarget tracking.

Results achieved during research process showed the correctness of developed application and were expected. Results have been checked using research literature. Evaluation was successful – Kalman filter and TBM implemented in "Eye of Ra" are worked correctly.

4 CONCLUSION

Data fusion techniques and algorithms have important meaning particularly in area of Artificial Intelligence, estimation, and data processing. Nowadays, data fusion is successfully used in many areas. For example now in medical diagnostics exist many decision support systems based on data fusion algorithms. These systems help doctor in giving diagnosis to the patient, of course final decisions are always made by human.

In this work the idea of data fusion and problems related to it have been presented. Additionally, two techniques have been considered in more depth. They are Kalman filtering and transferable belief model. After implementation of selected algorithms within "Eye of Ra" research process was started. Values of input during research process have been selected from research literature, and achieved results showed the correctness of developed application. The biggest contribution of the thesis is the tracking tool "Eye of Ra" which can be further used as research environment for data fusion studies. Within this tool Kalman filtering and transferable belief model has been implemented as an extension. This tool was developed using Java programming language, and in future can be transformed to work in P2P network, based on the Chedar platform [Smirnova 2002].

Kalman filter and transferable belief model were not selected randomly; the main cause is that these techniques are going to be used to support each other. These can be achieved for example by fusing them together within hybrid Bayesian network as in [Korpisaari 2001].

REFERENCES

[Banks 1986]	Stephen P. Banks, “Control systems engineering : modelling and simulation, control theory and microprocessor implementation”, Englewood Cliffs, N.J. : Prentice-Hall International, 1986, 614 s.
[Bar-Shalom 1993]	Bar-Shalom Yaakov, Xiao-Rong Li “Estimation and tracking : principles, techniques, and software”, Boston (Mass.) : Artech House, 1993 xxii, 511 s.
[Bayes]	Bayes Theorem, http://members.tripod.com/~Probability/bayes01.htm
[Bishop 2002]	http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
[Brookner 1998]	Brookner, Eli. “Tracking and Kalman filtering made easy” New York: Wiley, cop. 1998, 477 s.
[Catlin 1989]	Catlin, Donald E. “Estimation, control, and the discrete Kalman filter”, New York : Springer, 1989, 274 s.
[CheeseFactory]	Cheese Factory Project, http://tisu.it.jyu.fi/cheesefactory/
[CLT]	Central Limit Theorem, http://tigger.uic.edu/~hroberts/CO346su00.html
[Gelb 1988]	Gelb, A., "Applied Optimal Estimation," MIT Press, 1988.
[Hughes 1989]	Hughes , T.J. “Sensor Fusion in a military Avionics Environment” Measurement and Control. Sept. 1989 (203-205).

[Kalman 1960]	Kalman, R.E. ”A New Approach to Linear Filtering and Prediction Problems”, Transaction of the ASME – Journal of basic Engineering, pp 35-45 (March 1960).
[Kennes 1991]	Kennes, R., & Smets, P. Computational aspects of the Möbius transform. In P. P. Bonissone, M. Henrion, L. N. Kanal, & J. F. Lemmer (Eds.), Uncertainty in Artificial Intelligence 6 (pp. 401-416).
[Korpisaari 2001]	Studies on Data Fusion Techniques for Multitarget Tracking, Tampere University of Technology, 2001.
[Liu 1992]	Liu, L.J., Gu, Y.G., and J.Y. Yang. “Inference for Data Fusion.” Neural and Stochastic Methods in Image and Signal Processing. Proceedings of the SPIE – The International Society for Optical Engineering. 20-23 July 1992: San Diego, CA. SPIE, 1992 (670-677).
[Matlab]	Matlab 6.01, www.mathworks.com
[Maybeck 1979]	Maybeck, Peter S. “Stochastic Models, Estimation, and Control”, Volume 1, 1979, Academic press, Inc.
[Murphy]	Kevin Murphy’s page, http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html
[Shafer 1976]	Shafer G.,“A mathematical theory of evidence” Princeton Univ. Press. Princeton, NJ.
[Smets 1993]	Smets Ph., “An axiomatic justification for the use of belief function to quantify beliefs”, in Int. Joint Conf. on Artificial Intelligence, ILCAI-93, Ed. 1993, pp. 598-603, Morgan Kaufman, San Mateo, Ca.

[Smets 2001]	Smets Ph., “Matrix calculus for Belief Functions”. IRIDIA, November 10, 2001.
[SmetsPage]	Home page of Philip Smets, http://iridia.ulb.ac.be/~psmets/#G
[Smets 2001]	Ph. Smets, “Target identification based on the Transferable Belief Model interpretation of Dempster-Shafer Model Pars1: methodology”, IRIDIA, 2001.
[Smirnova 2002]	Smirnova V., ”Multi-agent system for distributed data fusion in peer-to-peer environment”, November 2002.
[Swinburne 2002]	Swinburne R., “Bayes’ theorem”, Oxford University Press, 2002
[Ruthven 1999]	Ruthven I., Lalmas M. and C.J. van Rijsbergen. “Retrieval through explanation: an abductive inference approach to relevance feedback.” 10th Irish Conference on Artificial Intelligence and Cognitive Science. Cork. 1999
[Yan 1995]	Jun Yan, Michael Ryan and James Power, Using Fuzzy Logic: Towards Intelligent Systems. Prentice-Hall, 1995

APPENDICES

Appendix 1. Extended table of Kalman filter output

Step#	Real value		Measured value		Estimated value		Predicted value	
	x	**y**	**x**	**y**	**x**	**y**	**x**	**y**
1	3	10	0.6	7.6	0.7998	7.7998	100.3999	107.3985
2	2	12	3.1	13.1	4.97	14.9133	11.4466	24.2479
3	1	14	2.1	15.1	3.4803	16.4509	5.34.87	21.2757
4	2	17	3.1	18.1	3.6507	18.8777	4.8373	22.7397
5	4	18	2.4	16.4	3.1531	18.359	3.8292	20.8928
6	6	19	7.4	20.4	6.146	20.573	7.3829	23.029
7	8	17	5.6	14.6	6.2765	17.7986	7.2904	19.201
8	10	15	8	13	7.7169	15.4736	8.8040	16.2296
9	12	16	13.4	17.4	11.504	16.917	13.0057	17.7785
10	13	17	13.6	17.6	13.3489	17.6754	14.8982	18.5226
11	14	19	11.9	16.9	13.1885	17.5973	15.5197	18.3265
12	15	20	12.9	17.9	13.605	18.0856	14.8271	18.7860
13	17	20	17.1	20.1	16.1013	19.5227	17.467	20.306
14	19	19	20	20	18.879	20.135	20.3967	20.9005
15	21	18	21.7	18.7	21.1199	19.6795	22.7127	20.3179
16	23	17	24	18.1	23.4798	19.0915	25.1497	19.6066
17	22	15	22	15	23.4130	17.0666	24.9127	17.3327
18	20	15	18.2	13.2	21.2198	15.0591	22.3647	15.1068
19	18	14	15.9	11.9	18.8146	13.3458	19.6242	13.2271
20	17	13	17.5	13.5	18.4594	13.3767	19.1605	13.272
21	16	12	16	12	17.4295	12.5753	17.9712	12.4064
22	15	11	12.5	8.5	14.9774	10.2689	15.2459	9.9049
23	14	9	11.8	6.8	13.3618	8.2073	13.4597	7.6896
24	13	7	11.1	5.1	12.1703	6.2746	12.1521	5.6295
25	12	5	13.1	6.1	12.6698	5.8865	12.698	5.2644
26	10	4	10	4	11.2251	4.5741	11.1219	3.8906
27	9	5	6.2	3.2	8.4358	3.5137	8.094	2.7966
28	7	6	4.2	3.2	5.9695	3.0167	5.4393	2.3191
29	7	8	7.7	8.7	6.6724	7.7997	6.2513	5.4099
30	8	10	5.8	7.8	6.0052	5.7134	5.5623	6.4386

Appendix 2. Source code of TBM algorithm

```
package kalmanfilter;
import java.awt.Dimension;

/**
 * <p>Title: Eye of Ra - TBM algorithm</p>
 * <p>Description: Implementation of Kalman Filter algorithm </p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: University of Jyväskylä</p>
 * @author Begemot
 * @version 1.0
 */

public class DSR {
  Matrix CPT;
  Matrix BPA;
  Matrix PLS;
  Matrix Belief, BeliefGBT;
  Matrix belGBT,bpaGBT;
  Matrix BET,BETGBT;
  int bpaDimI, bpaDimJ;
  public DSR() {
  CPT=new Matrix(3,3);
  CPT.set(0,0,0.8);
  CPT.set(0,1,0.0);
  CPT.set(0,2,0.0);
  CPT.set(1,0,0.0);
  CPT.set(1,1,0.9);
  CPT.set(1,2,0.8);
  CPT.set(2,0,0.2);
  CPT.set(2,1,0.1);
  CPT.set(2,2,0.2);
  bpaDimI=0; bpaDimJ=0;
  for(int i=0;i<CPT.getColumnDimension();i++){
    bpaDimI=bpaDimI*2+1;
  }
  for(int j=0;j<CPT.getColumnDimension();j++){
    bpaDimJ=bpaDimJ*2+1;
  }
  BPA=new Matrix(bpaDimI, bpaDimJ);

  }
 double rWQ(double value){



  return value;
  }

  Matrix fillBPA(){
  BPA.set(0,0, CPT.get(0,0));
  BPA.set(0,1,CPT.get(0,1));
  BPA.set(0,3,CPT.get(0,2));
  BPA.set(1,0,CPT.get(1,0));
  BPA.set(1,1,CPT.get(1,1));
  BPA.set(1,3,CPT.get(1,2));
  BPA.set(3,0,CPT.get(2,0));
  BPA.set(3,1,CPT.get(2,1));
  BPA.set(3,3,CPT.get(2,2));
  BPA.set(0,2,CPT.get(0,0)*CPT.get(0,1));
  BPA.set(0,4,CPT.get(0,0)*CPT.get(0,2));
  BPA.set(0,5,CPT.get(0,1)*CPT.get(0,2));
  BPA.set(0,6,CPT.get(0,0)*CPT.get(0,1)*CPT.get(0,2));

  BPA.set(1,2,CPT.get(1,0)*CPT.get(1,1));
  BPA.set(1,4,CPT.get(1,0)*CPT.get(1,2));
  BPA.set(1,5,CPT.get(1,1)*CPT.get(1,2));
  BPA.set(1,6,CPT.get(1,0)*CPT.get(1,1)*CPT.get(1,2));

  BPA.set(2,0,0);
  BPA.set(2,1,0);
  BPA.set(2,2,CPT.get(0,0)*CPT.get(1,1)+CPT.get(0,1)*CPT.get(1,0));
  BPA.set(2,3,0);
  BPA.set(2,4,CPT.get(0,0)*CPT.get(1,2)+CPT.get(0,2)*CPT.get(1,0));
  BPA.set(2,5,CPT.get(0,1)*CPT.get(1,2)+CPT.get(0,2)*CPT.get(1,1));

BPA.set(2,6,CPT.get(0,0)*CPT.get(0,1)*CPT.get(1,2)+CPT.get(1,0)*CPT.get(1
,1)*CPT.get(0,2)+

CPT.get(0,0)*CPT.get(1,1)*CPT.get(1,2)+CPT.get(1,0)*CPT.get(0,1)*CPT.get(
0,2)+

CPT.get(0,0)*CPT.get(1,1)*CPT.get(0,2)+CPT.get(1,0)*CPT.get(0,1)*CPT.get(
1,2));

  BPA.set(3,2, CPT.get(2,0)*CPT.get(2,1));
  BPA.set(3,4,CPT.get(2,0)*CPT.get(2,2));
  BPA.set(3,5,CPT.get(2,1)*CPT.get(2,2));
  BPA.set(3,6,CPT.get(2,0)*CPT.get(2,1)*CPT.get(2,2));

  BPA.set(4,0,0);
  BPA.set(4,1,0);
  BPA.set(4,2,CPT.get(0,0)*CPT.get(2,1)+CPT.get(0,1)*CPT.get(2,0));
  BPA.set(4,3,0);
  BPA.set(4,4,CPT.get(0,0)*CPT.get(2,2)+CPT.get(0,2)*CPT.get(2,0));
  BPA.set(4,5,CPT.get(0,1)*CPT.get(2,2)+CPT.get(0,2)*CPT.get(2,1));

BPA.set(4,6,CPT.get(0,0)*CPT.get(0,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(2
,1)*CPT.get(0,2)+

CPT.get(0,0)*CPT.get(2,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(0,1)*CPT.get(
0,2)+

CPT.get(0,0)*CPT.get(2,1)*CPT.get(0,2)+CPT.get(2,0)*CPT.get(0,1)*CPT.get(
2,2));

  BPA.set(5,0,0);
  BPA.set(5,1,0);
  BPA.set(5,2,CPT.get(1,0)*CPT.get(2,1)+CPT.get(1,1)*CPT.get(2,0));
  BPA.set(5,3,0);
  BPA.set(5,4,CPT.get(1,0)*CPT.get(2,2)+CPT.get(1,2)*CPT.get(2,0));
  BPA.set(5,5,CPT.get(1,1)*CPT.get(2,2)+CPT.get(1,2)*CPT.get(2,1));

BPA.set(5,6,CPT.get(1,0)*CPT.get(1,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(2
,1)*CPT.get(1,2)+

CPT.get(1,0)*CPT.get(2,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(1,1)*CPT.get(
1,2)+
CPT.get(1,0)*CPT.get(2,1)*CPT.get(1,2)+CPT.get(2,0)*CPT.get(1,1)*CPT.get(
2,2));

  BPA.set(6,0,0);
  BPA.set(6,1,0);
  BPA.set(6,2,0);
  BPA.set(6,3,0);
  BPA.set(6,4,0);
  BPA.set(6,5,0);

BPA.set(6,6,CPT.get(0,0)*CPT.get(1,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(1
,1)*CPT.get(0,2)+

CPT.get(0,0)*CPT.get(2,1)*CPT.get(1,2)+CPT.get(1,0)*CPT.get(2,1)*CPT.get(
0,2)+

CPT.get(1,0)*CPT.get(0,1)*CPT.get(2,2)+CPT.get(2,0)*CPT.get(0,1)*CPT.get(
1,2));
  return BPA;
  }
  Matrix prob2(int first, int second){
  Matrix  pr =new Matrix(1,CPT.getColumnDimension());
    for(int j=0;j<CPT.getColumnDimension();j++){
    pr.set(0,j,CPT.get(first,j)+CPT.get(second,j));
    }
    return pr;
  }

  Matrix GBTBPA(){
    Matrix GBT=new Matrix(bpaDimI,bpaDimJ+1);
     Matrix probability=null;
     double value=0;
     int t=0;
     for(int i =0;i<bpaDimI;i++){
      if(i==0||i==1||i==3){
      if(i==0||i==1)t=i;
      if(i==3)t=i-1;
      GBT.set(i,0,rWQ(CPT.get(t,0)*(1-CPT.get(t,1))*(1-CPT.get(t,2))));
      GBT.set(i,1,rWQ(CPT.get(t,1)*(1-CPT.get(t,0))*(1-CPT.get(t,2))));
      GBT.set(i,2,rWQ(CPT.get(t,0)*CPT.get(t,1)*(1-CPT.get(t,2))));
      GBT.set(i,3,rWQ(CPT.get(t,2)*(1-CPT.get(t,0))*(1-CPT.get(t,1))));
      GBT.set(i,4,rWQ(CPT.get(t,0)*CPT.get(t,2)*(1-CPT.get(t,1))));
      GBT.set(i,5,rWQ(CPT.get(t,1)*CPT.get(t,2)*(1-CPT.get(t,0))));
      GBT.set(i,6,rWQ(CPT.get(t,0)*CPT.get(t,1)*CPT.get(t,2)));
      for(int l=0;l<7;l++){
      value=value+GBT.get(i,l);
      }
      GBT.set(i,7,rWQ((1-value)));
      value=0;
      }
      if(i==2||i==5||i==4){
      if(i==2)probability=prob2(0,1);
      if(i==4)probability=prob2(0,2);
      if(i==5)probability=prob2(1,2);
      GBT.set(i,0,rWQ(probability.get(0,0)*(1-probability.get(0,1))*(1-
probability.get(0,2))));
      GBT.set(i,1,rWQ(probability.get(0,1)*(1-probability.get(0,0))*(1-
probability.get(0,2))));
      GBT.set(i,2,rWQ(probability.get(0,0)*probability.get(0,1)*(1-
probability.get(0,2))));
      GBT.set(i,3,rWQ(probability.get(0,2)*(1-probability.get(0,0))*(1-
probability.get(0,1))));
```

```
      GBT.set(i,4,rWQ(probability.get(0,0)*probability.get(0,2)*(1-
probability.get(0,1))));
      GBT.set(i,5,rWQ(probability.get(0,1)*probability.get(0,2)*(1-
probability.get(0,0))));

GBT.set(i,6,rWQ(probability.get(0,0)*probability.get(0,1)*probability.get
(0,2)));
      for(int l=0;l<7;l++){
      value=value+GBT.get(i,l);
      }
      GBT.set(i,7,rWQ(1-value));
      value=0;
      }
      if(i==6){
      for(int j=0;j<7;j++){
      if(j==6)GBT.set(i,j,1);
      else
      GBT.set(i,j,0.0);

      }
      }
     }
     return GBT;
  }

  Matrix fillBelief(Matrix bpa){
  Belief = new Matrix(bpa.getRowDimension(),bpa.getColumnDimension());
  for(int j=0;j<bpa.getColumnDimension();j++){
    Belief.set(0,j,bpa.get(0,j));
    Belief.set(1,j,bpa.get(1,j));
    Belief.set(2,j,bpa.get(0,j)+bpa.get(1,j)+bpa.get(2,j));
    Belief.set(3,j,bpa.get(3,j));
    Belief.set(4,j,bpa.get(0,j)+bpa.get(3,j)+bpa.get(4,j));
    Belief.set(5,j,bpa.get(1,j)+bpa.get(3,j)+bpa.get(5,j));
    Belief.set(6,j,bpa.get(0,j)+bpa.get(1,j)+bpa.get(2,j)+bpa.get(3,j)+
    bpa.get(4,j)+bpa.get(5,j)+
    bpa.get(6,j));
  }


  return Belief;
  }
 Matrix flip( Matrix row1){
  Matrix result;
  result= new Matrix(row1.getRowDimension(),1);
  for(int i =0, j=row1.getRowDimension()-1;
 (i<row1.getRowDimension())&&(j>=0);i++,j--){
  result.set(i,0,row1.get(j,0));
  }
  return result;
 }
 Matrix plaus(Matrix a){

  Matrix one= new Matrix (a.getRowDimension(),a.getColumnDimension(),
a.get(a.getRowDimension()-1,0));
  Matrix fl=flip(a);
  one=one.minus(fl);
  Matrix result =new Matrix(a.getRowDimension(),1);
  result.setMatrix(0,5,0,0,one.getMatrix(1,6,0,0));
  result.set(6,0, a.get(a.getRowDimension()-1,0));
  return result;
 }
 Matrix fillPlausibility(Matrix Bel){
 Matrix prom;
 Matrix result=new
Matrix(Bel.getRowDimension(),Bel.getColumnDimension());
 for(int i=0;i<Bel.getColumnDimension();i++){
  prom=Bel.getMatrix(0,6,i,i);
  result.setMatrix(0,6,i,i,plaus(prom));
 }
 return result;
 }
 Matrix fillBelGBT(Matrix bpa){
  BeliefGBT =new Matrix (bpa.getRowDimension(),bpa.getColumnDimension()-
1);
  for(int i=0;i<bpa.getRowDimension();i++){
    BeliefGBT.set(i,0,bpa.get(i,7)+bpa.get(i,0));
    BeliefGBT.set(i,1,bpa.get(i,7)+bpa.get(i,1));

BeliefGBT.set(i,2,bpa.get(i,7)+bpa.get(i,0)+bpa.get(i,1)+bpa.get(i,2));
    BeliefGBT.set(i,3,bpa.get(i,7)+bpa.get(i,3));

BeliefGBT.set(i,4,bpa.get(i,7)+bpa.get(i,0)+bpa.get(i,3)+bpa.get(i,4));

BeliefGBT.set(i,5,bpa.get(i,7)+bpa.get(i,1)+bpa.get(i,3)+bpa.get(i,5));

BeliefGBT.set(i,6,bpa.get(i,7)+bpa.get(i,0)+bpa.get(i,1)+bpa.get(i,2)+
    bpa.get(i,3)+bpa.get(i,4)+bpa.get(i,5)+bpa.get(i,6));

  }
  return BeliefGBT;
  }
  Matrix Pignistic(Matrix bpa){
    Matrix bets=new Matrix(3,7);
      //for(int i=0;i<bpa.getRowDimension();i++){
        for(int j=0; j<bets.getColumnDimension();j++){

bets.set(0,j,bpa.get(0,j)+(bpa.get(2,j)/2)+(bpa.get(4,j)/2)+(bpa.get(6,j)
/3));

bets.set(1,j,bpa.get(1,j)+(bpa.get(2,j)/2)+(bpa.get(5,j)/2)+(bpa.get(6,j)
/3));

bets.set(2,j,bpa.get(3,j)+(bpa.get(4,j)/2)+(bpa.get(5,j)/2)+(bpa.get(6,j)
/3));
        }
      //}
      return bets;
  }
  Matrix PignisticGBT(Matrix bpa){
    Matrix bets=new Matrix(7,3);
      for (int i=0;i<bets.getRowDimension();i++){
        bets.set(i,0,(bpa.get(i,0)/(1-bpa.get(i,7)))+(bpa.get(i,2)/(2*(1-
bpa.get(i,7))))+
         (bpa.get(i,4)/(2*(1-bpa.get(i,7))))+(bpa.get(i,6)/(3*(1-
bpa.get(i,7)))));
        bets.set(i,1,(bpa.get(i,1)/(1-bpa.get(i,7)))+(bpa.get(i,2)/(2*(1-
bpa.get(i,7))))+
         (bpa.get(i,5)/(2*(1-bpa.get(i,7))))+(bpa.get(i,6)/(3*(1-
bpa.get(i,7)))));
        bets.set(i,2,(bpa.get(i,3)/(1-bpa.get(i,7)))+(bpa.get(i,4)/(2*(1-
bpa.get(i,7))))+
         (bpa.get(i,5)/(2*(1-bpa.get(i,7))))+(bpa.get(i,6)/(3*(1-
bpa.get(i,7)))));
      }
      return bets;
  }
  void DSRinWork(){
    Matrix BA,bel;

    BA=fillBPA();
    BET=Pignistic(BA);
    Belief=fillBelief(BA);
    PLS=fillPlausibility(Belief);
    bpaGBT=GBTBPA();
    BETGBT=PignisticGBT(bpaGBT);
    belGBT=fillBelGBT(bpaGBT);
    System.out.println("result");
}
```

Appendix 3. Source code of Kalman filter algorithm

```
package kalmanfilter;
import java.math.*;
import java.util.*;
/**
 * <p>Title:Eye of Ra - Kalman Filter algorithm</p>
 * <p>Description: Implementation of Kalman Filter</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: University of Jyväskylä</p>
 * @author Begemot
 * @version 1.0
 */

public class Kalman {
  //State transition Matrix
private static Matrix STM;
public  String  forChart,  forChartXReal,forCharty,forChartYReal,  counter,
onemore,
measurX, measurY;

boolean nextStep=true;
public Matrix state ;
public Matrix H;
    int accuracy;
    double []x;
    double []y;
    double []xm;
    double []ym;
    double ips;
    Matrix xPred,xEstimated;
    public double T;
    public Matrix KalmanGain;
    public Matrix I;
    Matrix covPk,updateCovPk;
    Matrix Q;
    Matrix R;
    Matrix Y;
    Matrix Z;
//initialization
  public Kalman(int acc) {
    ips=0.1;
    forChart=new String();
    measurX=new String(); measurY=new String();
    forChartXReal=new String();
    forCharty= new String();
    forChartYReal=new String();
    counter=new String("0 ");
    onemore=new String();
    x = new  double [30];
    y = new  double [30];
    xm = new  double [30];
    ym = new  double [30];
    x[0]=3;
    x[1]=2;
    x[2]=1;
    x[3]=2;
    x[4]=4;
    x[5]=6;
    x[6]=8;
```

```
x[7]=10;
x[8]=12;
x[9]=13;
x[10]=14;
x[11]=15;
x[12]=17;
x[13]=19;
x[14]=21;
x[15]=23;
x[16]=22;
x[17]=20;
x[18]=18;
x[19]=17;
x[20]=16;
x[21]=15;
x[22]=14;
x[23]=13;
x[24]=12;
x[25]=10;
x[26]=8;
x[27]=7;
x[28]=7;
x[29]=8;
y[0]=10;
y[1]=12;
y[2]=14;
y[3]=17;
y[4]=18;
y[5]=19;
y[6]=17;
y[7]=15;
y[8]=16;
y[9]=17;
y[10]=19;
y[11]=20;
y[12]=20;
y[13]=19;
y[14]=18;
y[15]=17;
y[16]=15;
y[17]=15;
y[18]=14;
y[19]=13;
y[20]=12;
y[21]=11;
y[22]=9;
y[23]=7;
y[24]=5;
y[25]=4;
y[26]=5;
y[27]=6;
y[28]=8;
y[29]=10;
accuracy=acc;
Random r=new Random();
for(int i=0;i<x.length;i++){
int k;
k=r.nextInt(accuracy);
System.out.println("ganerated"+k);
if(k<(accuracy/2)){
  xm[i]=x[i]+k/10.;
ym[i]=y[i]+k/10.;
System.out.println("xm"+i+':'+xm[i]);
}else{
    xm[i]=x[i]-k/10.;
    ym[i]=y[i]-k/10.;
    System.out.println("xm"+i+':'+xm[i]);
    }
    }

    Z=new Matrix(4,1);
    Y=new Matrix(2,1);
    xPred=new Matrix(4,1);
    xEstimated=new Matrix(4,1);
    for(int p=0;p<xPred.getRowDimension();p++)
    xPred.set(p,0,1000);
    R= new Matrix(2,2);
    R.set(0,0,0.2);
    R.set(1,1,0.2);
   // R.set(2,2,0);
   // R.set(3,3,0);
    Q=new Matrix (4,4);

    for(int k=0;k<Q.getRowDimension();k++)Q.set(k,k,ips);
    H= new Matrix(2, 4);
    H.set(0,0,1);
    H.set(1,1,1);
    KalmanGain=new Matrix(4,4,0);
    KalmanGain=Matrix.identity(4,4);
    covPk=new Matrix(4,4);
    updateCovPk=new Matrix(4,4);
    for(int i=0;i<covPk.getColumnDimension();i++)covPk.set(i,i,1000);
    double [][] newMatrix, forState;
    T=0.1;
    STM=new Matrix(4,4);
    state= new Matrix (4,1);
      for(int i=0;i<4;i++){
       // state.elements[0][i]=0;
        for(int j=0;j<4;j++){
          if(i==j)STM.set(i,j,1);
           else
          if(i==2&&j==0||i==3&&j==1)
            STM.set(i,j,T);
          else
          STM.set(i,j,0);
      }

  }
  }

  //Prediction Phase
Matrix projectAHeadState(Matrix stm,Matrix st){
  return stm.times(st);
}
Matrix projectErrorAhead(Matrix st, Matrix esq, Matrix cov){
  return ((st.times(esq).times(st.transpose()).plus(cov)));
}

//Correction Phase
Matrix kalmanGain(Matrix P, Matrix H, Matrix R){
Matrix r1=H.times(P);
   return
((P.times(H.transpose())).times((((H.times(P)).times(H.transpose())).plus
(R)).inverse()));
}
Matrix computeInnovation(Matrix y, Matrix H, Matrix x){
  return y.minus(H.times(x));
}
```

```
Matrix computeEstimate(Matrix x, Matrix K, Matrix z){
  return x.plus(K.times(z));
}
Matrix updateError(Matrix I,Matrix K,Matrix H,Matrix P){
return (Matrix.identity(4,4).minus(K.times(H))).times(P);
}
 void kalmanInWork(){
  for(int i=0;i<30;i++){
    KalmanGain=kalmanGain(covPk,H,R);
    Y.set(0,0,xm[i]);
    Y.set(1,0,ym[i]);
    Z=computeInnovation(Y,H,xPred);
    xEstimated=computeEstimate(xPred,KalmanGain,Z);
    updateCovPk=updateError(I,KalmanGain,H,covPk);
    //Prediction phase
    xPred=projectAHeadState(STM,xEstimated);
    covPk=projectErrorAhead(STM,updateCovPk,Q);
    System.out.println("x:"+x[i]+"y:"+y[i]);
    System.out.println("xEstimated:"+xEstimated.get(0,0));
    System.out.println("yEstimated:"+xEstimated.get(1,0));
    forChart=forChart+(new Double(xEstimated.get(0,0))).toString()+" ";
    forChartXReal=forChartXReal+(new Double(x[i])).toString()+" ";
    measurX=measurX+(new Double(xm[i])).toString()+" ";
    forCharty=forCharty+(new Double(xEstimated.get(1,0))).toString()+" ";
    forChartYReal=forChartYReal+(new Double(y[i])).toString()+" ";
    measurY=measurY+(new Double(ym[i])).toString()+" ";
    counter=counter+(new Integer(i+1)).toString()+" ";
    onemore=(new Integer(i+1)).toString();
  }
}
}
```