16th International Conference on Information Fusion
Istanbul, Turkey, July 9-12, 2013

Cross-layer Utility-based System Optimization

Maarten Ditzel, Leon Kester, Sebastiaan van den Broek, Martin van Rijn
TNO, Oude Waalsdorperweg 63, 2597 AK The Hague, The Netherlands
Email: maarten.ditzel@tno.nl

***Abstract*—Multilevel fusion systems need provisions to optimally schedule scarce processing and communication resources. To this end, we explore the idea of using utility-based metrics to optimize the run-time operation of a computation and communication constrained multilevel system, including automatic decision support measures such as course of action planning.**

In a simplified case study, we extend a processing chain with local utility-based run-time feedback mechanisms. This allows the processes in the distributed chain to improve their behavior towards a common measure of effectiveness using local interactions only. Simulation results show an increased performance of utility-based optimization over constant rate or random techniques.

***Index Terms*—system design; system architecture; system modeling; distributed fusion; autonomous systems; fusion management**

I. Introduction

The situations in which observation systems have to operate are becoming more complex and increasingly dynamic. None withstanding, these systems are expected to seamlessly work together to provide their users with actionable information automatically. Moreover, they are expected be able to adapt their behavior to the evolving situation without operator intervention. The dynamics of the situation may cause changes in the capability of sensors, communication and computation resources, e.g., due to changing weather conditions. Also, the actuality of the situation may result in a gradual or sudden alteration of what important information is. Therefore the next generation fusion systems need to (i) automatically process information up to higher abstraction levels; (ii) operate distributedly; and (iii) have provisions to optimally manage sensors, processing and communication resources to cope with the changing circumstances.

Modern systems apply different optimization techniques for sensor management and for selecting which data needs to be scheduled, using for instance first-come, first-served, priority-based, or processor sharing (e.g., weighted fair queuing) techniques. However, these optimization techniques do not take into account the utility of the data for the ultimate effectiveness of the system. They either use low level information theoretic measures [1]–[3], or do not use information related measures at all. Measures based on expected utility are more appropriate, because they more directly relate to the effectiveness or goal of the system. In [4], [5] such measures are applied for sensor management. In [6] the authors derive utility metrics for classes of data flows, for instance the transmission of an image over a network channel. However, they do not take into account the value of the actual data element in a particular situation. [7] recognizes the value of using situation dependent utility measures, but does not provide a method to derive these during operation (run-time) of a system.

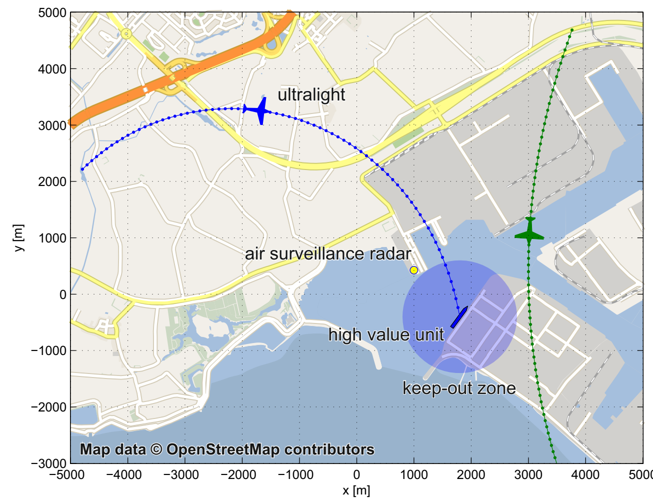


Figure 1: Overview of the high value unit protection scenario: two targets approach a high value unit (a moored tanker), one friendly approaching from the northeast and one hostile approaching from the northwest.

In [8]–[10], utility-based metrics are used in a distributed system to build a common operational picture. The authors have shown that the use of utility-based metrics to optimize the communication and the information exchange between similar functions at object assessment level gives substantial performance gains. In this paper, we further explore the idea of using utility-based metrics to optimize the run-time operation of a computation and communication constrained multilevel system, including automatic decision support measures such as course of action planning.

To analyze and assess the potential benefits of utility-based cross-layer system optimization, we explore the optimization of a processing chain in a case study. The chain consists of three functions: course of action planning, goal estimation and state estimation. Since the system is distributed and the communication resources are constrained also the optimization needs to be distributed. For better comprehension of the interactions and the results, the functions have simplified implementations which, in our opinion, do not limit the validity of the results. For each function we implement a (local) utility-based run-time feedback mechanism. The added mechanisms enable the individual processes forming the distributed chain

978-605-86311-1-3 ©2013 ISIF

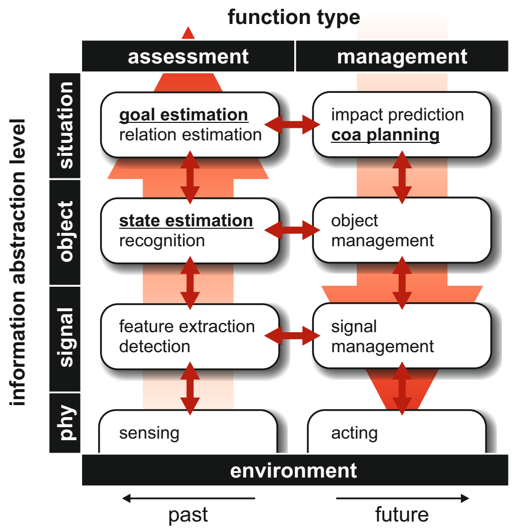


Figure 2: Modified information abstraction model using internal (run-time) feedback.

to improve their behavior towards a common global metric using local interactions only.

The paper is organized as follows. First in Section II a scenario is described (see Fig. 1) in which an air surveillance radar has to schedule its resources to effectively classify different airborne targets. Next, in Section III, a simplified processing model of a processing chain is described, consisting of three separate functions starting state estimation, up until course of action planning. Then, in Section IV, an optimization approach is explained, that enables these functions to optimize their behavior locally to a global performance metric, in response to the actual changes in the situation. In Section V results are presented and compared to different optimization methods. Finally, in Section VI, we conclude this paper.

II. Scenario Description

To assess the performance of utility-based optimization of a processing chain, we analyze its workings in an exemplary scenario where a high value unit has to be protected. In this scenario several assets are available to prevent an aerial attack of a low flying ultralight aircraft on a moored tanker carrying hazardous chemicals. An air surveillance radar is available to scan the airspace for potential threats. Moreover, an armed helicopter is at standby, first to intercept and deter a detected threat, or ultimately to engage a hostile target. Apart from potential hostile ultralights, also other non-hostile planes may enter the airspace under observation. Of course, engaging one of these ignorant bystanders by mistake should be avoided at all times.

Sensing, detection and state estimation take place at the surveillance radar. Higher level functions such as situation assessment and course of action planning are located at a command center unit. Here, it is also possible to issue the order to dispatch the helicopter. The decision to engage a target is made when a target enters a keep-out zone. The size of the zone corresponds to a threshold on a function of range, velocity and heading, similar to a time of impact, such that the imminent attack can still be countered effectively. As a consequence the zone's boundaries depend on the target's actual range, heading and velocity, relative to the position of the tanker.

Due to communication and/or processing constraints, it is unfeasible to send all plot data to the command center. As all processes are automated, the processing chain itself has to optimize it behavior such that the limited resources are put to maximum use. Therefore, as already argued, each element in the processing chain needs quantitative metrics to be able to decide which data to send or process.

Fig. 1 depicts the scenario graphically. In this scenario two ultralights approach the tanker. One of the planes (green trajectory) approaches the keep-out zone, but slowly turns away and subsequently passes by. The other (blue trajectory) initially seems to pass by, but gradually turns towards the tanker to attack it.

III. Processing Model

In this paper we investigate the optimization of a processing chain layered according to an updated version of the information abstraction model using internal feedback introduced in [9] (see Fig. 2). The modifications involve the reordering of the functionality originally found in the impact layer, and the subsequent removal of this layer. Its functions are now moved to situation management (impact prediction). The benefit of this restructuring is a clear temporal distinction between the assessment side and the management side.

As case study we have selected three functions of the modified model: course of action (CoA) planning, goal estimation and state estimation. Our prime interest is to study the effects of utility-based decentralized optimization of a processing chain. Therefore we intentionally use simple implementations for the three functions. Subsequently, for each function we implement an internal run-time feedback mechanism, such that each function is able to optimize its own behavior to improve the overall effectiveness of the processing chain as a whole in a structured manner.

A. CoA Planning

For the planning model we choose a simple function to select one of three possible actions. The function takes a single parameter as input: the estimated probability p that an observed target actually is about to perform a hostile act. The output of the function is one of the three possible actions that can be taken:

- *action* A_1 – do nothing,
- *action* A_2 – intercept and deter, and
- *action* A_3 – engage and destroy the target.

Each action to choose from has an associated reward $R_{A_i}(p)$ given by

$$R_{A_i}(p) = V_{A_i}(p) - C_{A_i}(p),\ i = 1 \ldots 3. \qquad (1)$$

Here, $V_{A_i}(p)$ is the anticipated value of taking action A_i given probability p

$$V_{A_1}(p) = -V_T\ p, \tag{2}$$
$$V_{A_2}(p) = -\frac{V_T}{E_H}p, \tag{3}$$
$$V_{A_3}(p) = 0, \tag{4}$$

where V_T is the net worth of the tanker, and E_H the effectiveness of the deterrence. $C_{A_i}(p)$ is the expected cost associated with action A_i

$$C_{A_1}(p) = 0, \tag{5}$$
$$C_{A_2}(p) = C_H, \tag{6}$$
$$C_{A_3}(p) = C_H + C_W + C_F(1-p), \tag{7}$$

taking into account the cost of deploying a helicopter C_H, the cost of deploying a weapon C_W, and the cost of inadvertently destroying a non-hostile (friendly) target C_F.

Substituting Eqn. (2) to (7) into Eqn. (1) gives

$$R_{A_1}(p) = -V_T\ p, \tag{8}$$
$$R_{A_2}(p) = -\frac{V_T}{E_H}p - C_H, \tag{9}$$
$$R_{A_3}(p) = -C_H - C_W - C_F(1-p). \tag{10}$$

The resulting rewards are depicted in Fig. 3. Finally, the output of the decision function is the action $A_d(p)$ that maximizes the expected reward, given the estimated probability p, that is

$$A_d(p) = \arg\max_{A_i} R_{A_i}(p). \tag{11}$$

Substituting Eqn. (8) to (10) and solving Eqn. (11) gives

$$A_d(p) = \begin{cases} A_1 & \text{if } p \leq P_{12}, \\ A_2 & \text{if } P_{12} < p \leq P_{23}, \\ A_3 & \text{otherwise,} \end{cases} \tag{12}$$

where $P_{12} = \frac{C_H E_H}{V_T(E_H-1)}$ and $P_{23} = \frac{E_H(C_W+C_F)}{V_T+E_H C_F}$.

B. Goal Estimation

As exemplary implementation for the goal estimator we take a function that takes the individual targets' states as its inputs. A target's state includes the target's position, course over ground and velocity. Combined with the location of the tanker, the goal estimator calculates an estimate of the probability p that a target poses a threat to the moored tanker. Ad hoc calculation of p is given by

$$p(r, v, \phi) = p_r(r) \cdot p_v(v) \cdot p_\phi(\phi), \tag{13}$$

with r the range of a target, v the radial velocity and ϕ the heading of target, all with respect to the tanker's position, i.e., a heading of 0 degrees signals that the target is heading straight towards the tanker. $p_r(r)$, $p_v(v)$ and $p_\phi(\phi)$ respectively indicate the range, velocity and heading dependent partial probabilities. These are given by

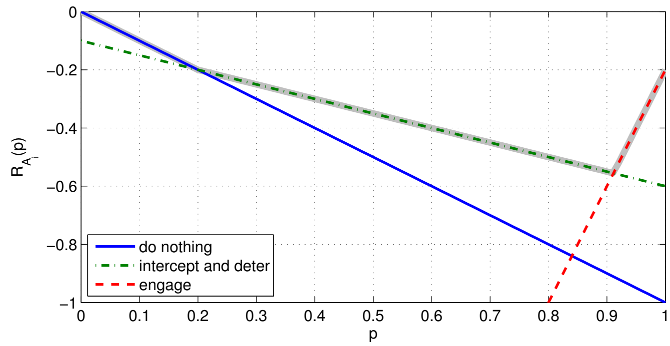


Figure 3: Normalized expected reward R_{A_i} for taking action A_i (with $V_T = 1$, $C_F = 4$, $E_H = 2$, $C_H = 0.1$, and $C_W = 0.1$).

$$p_r(r) = \begin{cases} \frac{R_{max}-r}{R_{max}} & \text{if } r \leq R_{max}, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$
$$p_v(v) = \begin{cases} \frac{v}{v_{max}} & \text{if } v \leq v_{max}, \\ 1 & \text{otherwise,} \end{cases} \tag{15}$$
$$p_\phi(\phi) = 1 - \left|\frac{\phi}{\pi}\right|. \tag{16}$$

Again, this particular implementation of goal estimation is intentionally kept simple for better comprehension of the optimization mechanism.

Goal estimation is done using the available state estimates (r, v and ϕ). When no current update from the state estimation component is obtained, the state is predicted using the latest update and a prediction model to estimate the change of state and its possible error. For example, a Kalman Filter could be used with a constant velocity model. In this paper, a simple constant velocity model is used (i.e., velocity and heading stay constant, while range changes with time), with errors increasing linearly in time to a maximum uncertainty.

C. State Estimation

State estimation is the final component of the processing chain. It is foreseen that a kinematic tracker processes the contacts delivered by the feature extraction and detection components. Hence, it is able to provide the goal estimator with the required position, direction and velocity information. Moreover, it has additional internal information on the accuracy of these information elements, for instance presented by a covariance matrix. In the current example, no explicit tracking is applied, and measurements are passed on with a set accuracy.

IV. Utility-based Optimization

For the optimization of a processing chain, two approaches are possible: centralized or decentralized optimization. In centralized optimization there is a single optimization entity that reasons about the optimal working of the whole chain.

The entity does however require information on the individual states of each of the constituting functions of the processing chain. In decentralized optimization each component in the chain is capable of optimizing its own function. Moreover, restrictions apply to what functions know or are able to learn about each others internal behavior, mostly because of communication constraints and delays.

Irrespective of the approach taken, both the central and the distributed optimization functions need clues to what is to be achieved by the processing chain as a whole, i.e., an expression of the chain's effectiveness. Once that is known, an optimization function can reason about how it may improve the behavior of the function(s) under its control to better meet the chain's overall goal. Alternatively phrased, the question is how to tune the functions' individual performances to get to an overall increased effectiveness of the chain. To that end, we investigate the use of utility functions to express the value of information for the overall effectiveness.

A commonly heard objection concerning the use of the concept of utility for system optimization is that it is hard to derive meaningful utility functions or utility metrics, particularly during operation of the system. In this paper, we do not intend to state that this is actually an easy job. We do however show that a system's effectiveness can already be improved with only very basic models and approximations to calculate the effect of new information. Ultimately, only if a system designer is able to explicitly express what is utile, purposeful automatic optimization of a system is possible.

In case also an estimate is available for the cost of processing and sending the data to the receiving functions (in addition to the anticipated value of new information), we are able to evaluate the condition

$$V > C, \tag{17}$$

where V represents the value of the new data and C the cost to compute and communicate the data. If this condition is met, it is profitable to process and exchange the data and have the receiving functions update their internal state and estimates. Evidently, in this case the value and cost metrics should have the same unit. Unfortunately, this is often not the case or even possible. However, the value and cost metrics can then still be used to determine the relative gain for each data element

$$\frac{V}{C}. \tag{18}$$

Those elements with the largest relative gains, can then be selected for further processing and communication.

A. Value of Information

At the heart of the aforementioned approach is the calculation of the value of new information. We propose to calculate the value of a new information element based on the effect it has on the process that uses that information. Thereto, we first introduce the potential gain in effect $\Gamma(x_n, x_o)$ of new information x_n given prior information x_o. It should be noted that both x_n and x_o possibly represent multi-dimensional information elements. Then, the value V of the new information

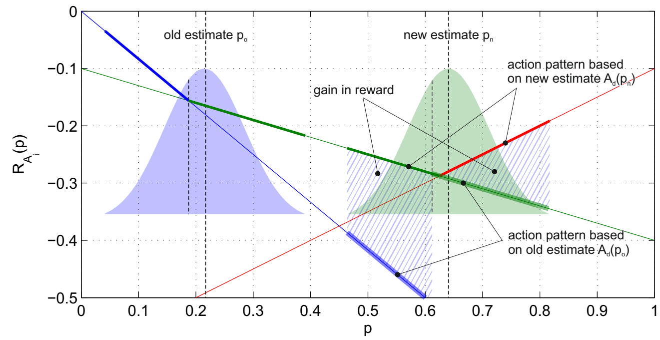


Figure 4: Reward of actions and probability density functions of old and new estimates of p depicting the calculation of V_p.

given the old information can be calculated using

$$V = \int_{X_o}\int_{X_n} \Gamma(x_n, x_o)P(x_n|x_o)P(x_o)dx_n dx_o, \tag{19}$$

where $P(x_n|x_o)$ and $P(x_o)$ denote the probability distributions of x_n (conditional on x_o) and x_o, respectively. X_n and X_o contain the range of all possible values for x_n and x_o.

Once the potential value of new information is known, it can be used to select what information elements are important enough to communicate or process. One way, is to communicate the full expression that describes the value of information to the information providers. Another approach is to calculate the value's sensitivity to a change in information and use this to steer the selection process. The latter is the approach taken in this paper.

B. CoA Planning

To express the need of the CoA planning module for new information provided by the goal estimation function that calculates p, we choose to quantify this need in the form of a utility function U. As stated, we derive the utility function of the planning module from the extent to which new information influences the decision making process. Therefore, we first define the potential gain in effect due to new information as the potential gain in reward as

$$\Gamma(p_n, p_o) = R_{A_d(p_n)}(p_n) - R_{A_d(p_o)}(p_n). \tag{20}$$

Here, p_o and p_n are the old and new estimated threat probabilities, respectively. Then using Eqn. (19), the value V_p of new information p_n (given the old information p_o) can be calculated using

$$V_p = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Gamma(p_n, p_o)P(p_n|p_o)P(p_o)dp_n dp_o, \tag{21}$$

where $P(p_n|p_o)$ and $P(p_o)$ are the probability density functions of p_n (conditional on p_o) and p_o respectively. It corresponds to proposing a new estimated probability p_n – given the prior p_o – and then calculating the integral over the uncertainty distributions of the old and new estimates, keeping the action set corresponding to the old p_o.

Given the progressive nature of the underlying processes that jointly provide the threat probability, it is reasonable to expect that p_n will be an update of the previous estimate p_o. Moreover, if we assume $P(p_o)$ to be normally distributed then we can expect p_n and p_o to have fully correlated Gaussian distributions such that

$$p_n = f(p_o) = \frac{\sigma_{p_n}(p_o - \mu_{p_o})}{\sigma_{p_o}} + \mu_{p_n} \tag{22}$$

and

$$P(p_n|po) = \delta(f(p_o) - p_o), \tag{23}$$

where $\delta(.)$ denotes the Dirac delta function. In this equation, σ_{p_o}, μ_{p_o}, and σ_{p_n} and μ_{p_n} indicate the standard deviation and mean of the old and new distributions, respectively. Substitution in Eqn. (21) gives

$$V_p(\mu_{p_n}, \sigma_{p_n}, \mu_{p_o}, \sigma_{p_o}) = \int_{-\infty}^{\infty} \Gamma\left(\frac{\sigma_{p_n}(p_o - \mu_{p_o})}{\sigma_{p_o}} + \mu_{p_n}, p_o\right) \frac{\sigma_{p_n}}{\sigma_{p_o}} P(p_o) dp_o, \tag{24}$$

with

$$P(p_o) = \frac{1}{\sqrt{2\pi}\sigma_{p_o}} \exp\left\{-\frac{1}{2}\left(\frac{p_o - \mu_{p_o}}{\sigma_{p_o}}\right)^2\right\}. \tag{25}$$

This approach is graphically depicted in Fig. 4.

Once we have the above expression for the value of new information, it becomes possible to determine the sensitivity of the CoA function to new information by calculating its partial derivatives. New information consists of two elements: the mean value of the new estimated distribution of μ_{p_n} and it standard deviation σ_{p_n}. Therefore, we arrive at two utility functions given by

$$U_{\mu_p}(\mu_{p_o}, \sigma_{p_o}) = \frac{\partial}{\partial \mu_{p_n}} V_p(\mu_{p_n}, \sigma_{p_o}, \mu_{p_o}, \sigma_{p_o}), \tag{26}$$

and

$$U_{\sigma_p}(\mu_{p_o}, \sigma_{p_o}) = \frac{\partial}{\partial \sigma_{p_n}} V_p(\mu_{p_o}, \sigma_{p_n}, \mu_{p_o}, \sigma_{p_o}), \tag{27}$$

The values of these derivatives at operating point $(\mu_{p_o}, \sigma_{p_o})$ are the metrics that will be fed back to the goal estimation function. With these, the goal estimator will be able to optimize its behavior to provide the most useful information for the top level CoA planning function.

In Fig. 5 the feedback metrics are plotted for different operating points. Fig. 5a shows that sensitivity for changes in μ_p is high if σ_p is small and μ_p is near a change of action (see also Fig. 3). Sensitivity to σ_p is higher in the same cases, but even more so when μ_p is near 0.5 and σ_p is large, as seen in Fig. 5b. This suggests information has more value if it makes the uncertainty smaller (i.e., decreases σ_p), or if μ_p varies near values where the best action changes.

C. Goal Estimation

In general, any component may independently reason about the information it has to deliver to other components based on their interest expressed in the form of utility. In its term, it will combine and translate the given requests to express its own information needs to subsequent modules. Since, in our example, the goal estimation only serves the CoA planning module, the translation is straightforward. First, the approximate expected value of a change in μ_p or σ_p may be calculated using

$$\hat{V}_p(\Delta\mu_p, \Delta\sigma_p) = U_{\mu_p}(\mu_{p_o}, \sigma_{p_o})\Delta\mu_p + U_{\sigma_p}(\mu_{p_o}, \sigma_{p_o})\Delta\sigma_p, \tag{28}$$

where $\Delta\mu_p$ and $\Delta\sigma_p$ indicate the differences of the current estimates of the mean and deviation with respect to the last estimate of p_o sent.

Taking Eqn. (14) to (16), it takes little effort to calculate the partial derivatives of $p(r, v, \phi)$. Multiplying these by the feedback metrics U_{μ_p} and U_{σ_p} gives an approximation of the sensitivity of the goal estimation to new information. For instance

$$U_{\mu_r}(\mu_{r_o}) = \frac{\partial \hat{V}_p(\Delta\mu_p, \Delta\sigma_p)}{\partial \mu_{r_o}} \approx \left.\frac{\partial p(r, v, \phi)}{\partial r}\right|_{r=\mu_{r_o}} U_{\mu_p}(\mu_{p_o}, \sigma_{p_o}) \tag{29}$$

and

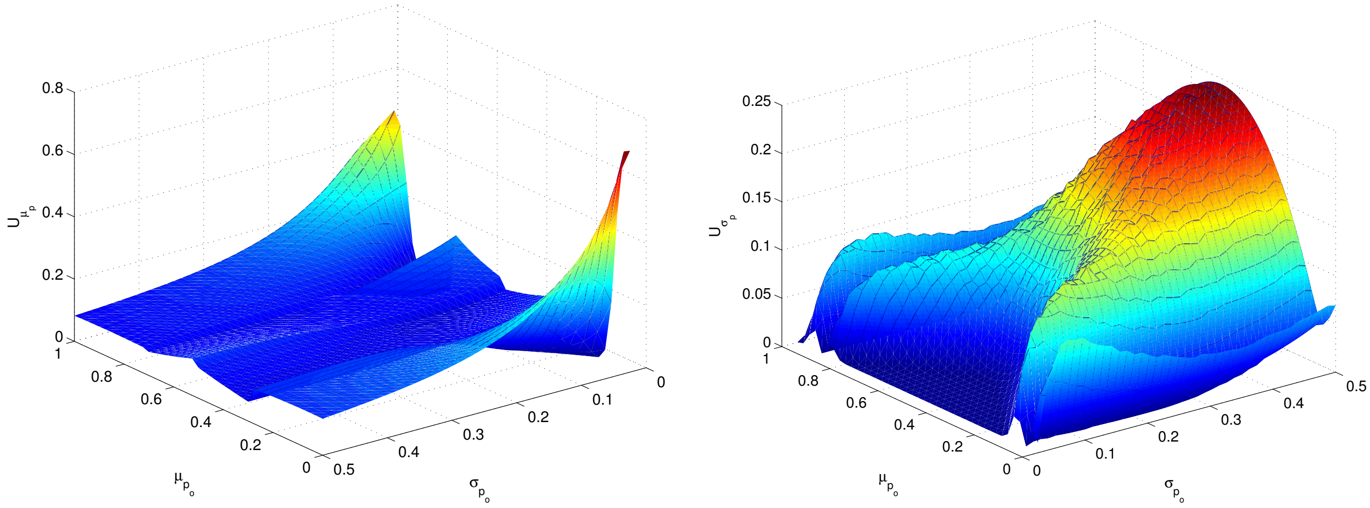
$$U_{\sigma_r}(\sigma_{r_o}) = \frac{\partial \hat{V}_p(\Delta\mu_p, \Delta\sigma_p)}{\partial \sigma_{r_o}} \approx \left.\frac{\partial p(r, v, \phi)}{\partial r}\right|_{r=\mu_{r_o}} U_{\sigma_p}(\mu_{p_o}, \sigma_{p_o}) \tag{30}$$

give the utility functions for new range information (its mean and standard deviation). Here, we use that the derivatives of the deltas in the value equation, equal the derivatives of p, using p_o as operating point.

An equivalent approach is used to calculate the sensitivity to new values of a target's velocity v and heading ϕ. As before, the sensitivity to new information is used to calculate feedback metrics, but now for the state estimation module. Predictions of the state are used when no new values are obtained from the state estimation component, as explained in Section III-B. This yields best estimates of values, with a possible error that increases in time. Since the partial derivatives of p to the state values have cross-dependencies (e.g., sensitivity to heading decreases with range), worst case sensitivity values are computed as feedback metrics for the state estimation function.

D. State Estimation

As in this example the state estimation is the final component in the processing chain, we can directly calculate the differences in mean and standard deviation of the range, velocity and heading given the last values sent to the goal estimation function, and using the same state prediction as used in goal estimation. With the feedback metrics of the goal



(a) (b)

Figure 5: Plot of the sensitivity of the expected value V_p of the CoA function to (a) changes in the estimated mean μ_{p_n} and (b) changes in the standard deviation σ_{p_n}, used as a feedback metrics U_{μ_p} and U_{σ_p} for the goal estimation function.

estimation it is possible to estimate the value of new range information for the goal estimation

$$\hat{V}_r(\Delta\mu_r, \Delta\sigma_r) = U_{\mu_r}(\mu_{r_o})\Delta\mu_r + U_{\sigma_r}(\sigma_{r_o})\Delta\sigma_r \qquad (31)$$

where $\Delta\mu_r$ and $\Delta\sigma_r$ indicate the deltas of the current estimates of the mean and deviation with respect to the last range estimate r_o sent. Again, the value gain for a target's velocity and heading can be calculated in an equivalent manner.

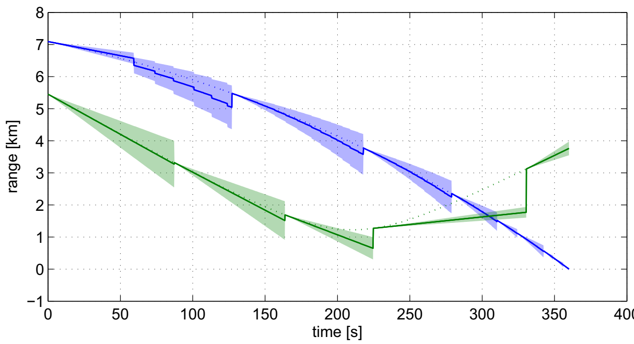
Overall, we now are able to value the contribution of individual information elements to the overall performance of the processing chain. As a consequence, we can now decide which information to process or communicate in an educated manner.

V. Results

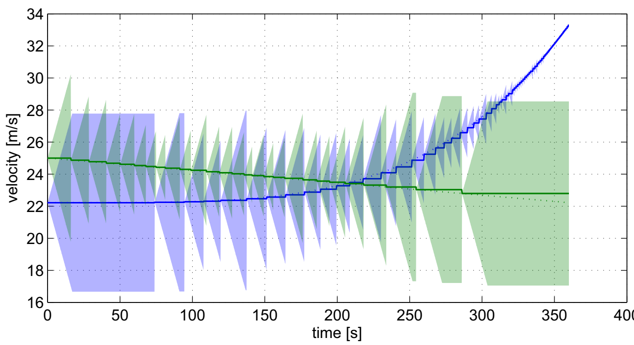
The utility-based strategy as described in the previous section has been applied to optimize the processing chain for the scenario as presented in Section II. Fig. 6 plots the internal estimates of the range, velocity, heading and the threat probability per target versus the actual values. The update results are depicted for the case that 2.8% of the measurements are communicated and processed. The graphs show that the uncertainty in heading increases faster than that in range and velocity. This is in line with the motion profile of an ultralight: it can change heading fast, but cannot suddenly be much closer. This results in more frequent updates in heading. Consequently, as long as the heading is known not to be directed to the tanker, range predictions will remain accurate and updates in range are not made. This changes if the range is small, and all values are updated more often. This is the case at the very end of the scenario for the blue (hostile) ultralight when the probability of threat becomes high and half-way the trajectory of the green ultralight. For the blue ultralight it is interesting to see that all values are updated less often, when the ultralight is far away, not directed towards the tanker. In this case, the sensitivity to for instance a change in velocity is reduced by the low contribution (close to zero) to Eqn. (13) of the range and heading values. The last pane shows the resulting updates for the threat probability of each target.

Finally, Fig. 7 shows (i) the reward of p in the case all the contacts would have been used, (ii) the predicted reward, which is the best estimate of the system about the reward of p, and (iii) the actual reward of p. We see that a reasonable update strategy is followed, despite the (sometimes large) differences between the estimated reward and the actual reward. Overall we observe data to be updated more often when the situation becomes more threatening or when parameters become more dominant. For instance, heading updates are sent most often for the friendly target around half of the scene duration. This corresponds to the situation where the target is closest to the tanker and a change in heading may result in a rapid change in estimated threat level. Likewise, the heading updates for the hostile target are most frequent and the end of the scenario, when the target becomes a definite threat.

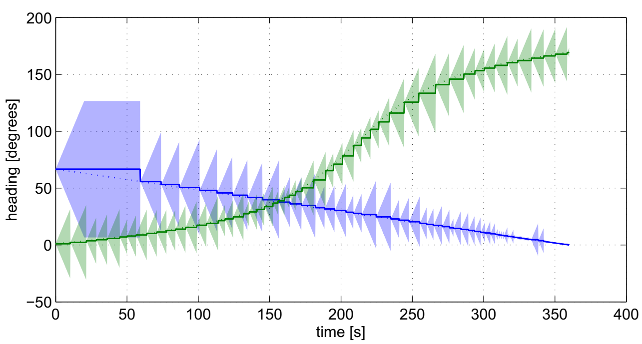
In order to get a fair comparison of the performance of the optimization method Fig. 8 plots the average missed reward for fifty ultralights with randomly generated trajectories. In total six methods are compared. The first one is the utility-based cross-layer method with individual updates (per value) for range, velocity and heading as discussed above for the two ultralight scenario. The second method is the similar, but now the updates for range, velocity and heading are sent simultaneously (per measurement). In addition, the figure also contains results for randomly selecting updates and for transmitting updates at a predefined constant rate, both methods both per value and per measurement. The results show that the random updates per value and per measurement both perform significantly worse than updates at constant rate and updates based on utility. At low update rates the utility-based method also outperforms the constant rate method.



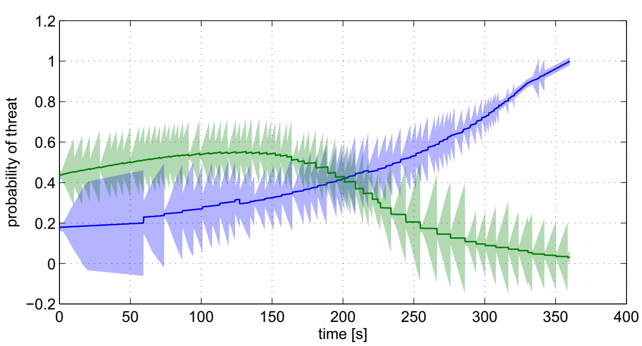
(a) Range r.



(b) Velocity v.



(c) Heading ϕ.



(d) Threat probability p.

Figure 6: Actual values (dotted lines) and internal estimates (solid lines) of the range, velocity, heading and the probability of threat for each target. The colored areas indicate the 1-σ deviation from the estimate mean.

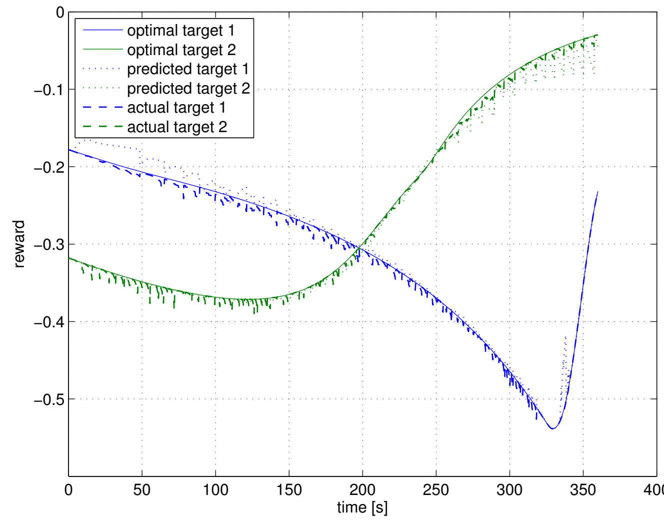


Figure 7: Optimal, predicted and actual rewards for each target. The predicted rewards indicate the erroneously assumed reward from the estimated state, based on which actions are calculated. The dashed lines show the actual reward of the estimated actions, using the optimal state as ground truth. The differences between the actual and optimal lines represent the missed reward.

In Fig. 9 the maximum missed reward is plotted. This measure is a good indication for the robustness of the method. Here the differences between the methods are even more distinctive. In this case the utility per value method performs slightly better than the utility per measurement method, most likely because the update per value method has an extra of degree of freedom to update the most utile value.

VI. Conclusions

Overall, the experiments give a good indication of how to perform cross-layer optimization with resource constraints. The simulations show that the utility based method outperforms the method which updates at random times and the method that updates at constant rate, requiring less updates for similar results. This shows that the use of utility measures allows automatic reasoning about the actual run-time value of data to improve the overall effectiveness of the system, using local interaction only. Making decisions for each value of range, velocity and heading separately yields better results than deciding for all three values at the same time, except at very low data rates, where the latter produced the best results. Results depend on the evaluated trajectories. For some, a constant rate would describe the situation with similar result as the utility-based one. However, an important difference between the utility-based method with respect to the other fixed rate methods is that it is able to adjust to the actual trajectories, where the fixed rate methods are unable to do so.

Acknowledgment

This research was conducted as part of the Sensor Technology Applied in Reconfigurable Systems (STARS) project funded by the Dutch Government.

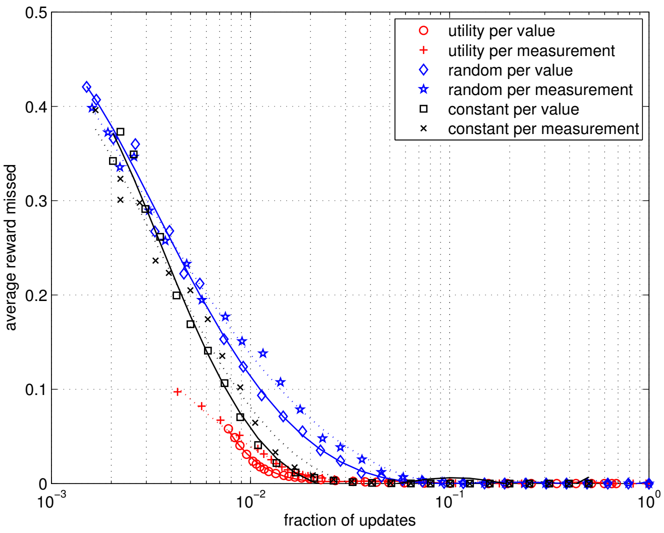


Figure 8: Average value missed per assessment plotted as function of the fraction of updates sent for utility-based, random and constant rate selection of updates.

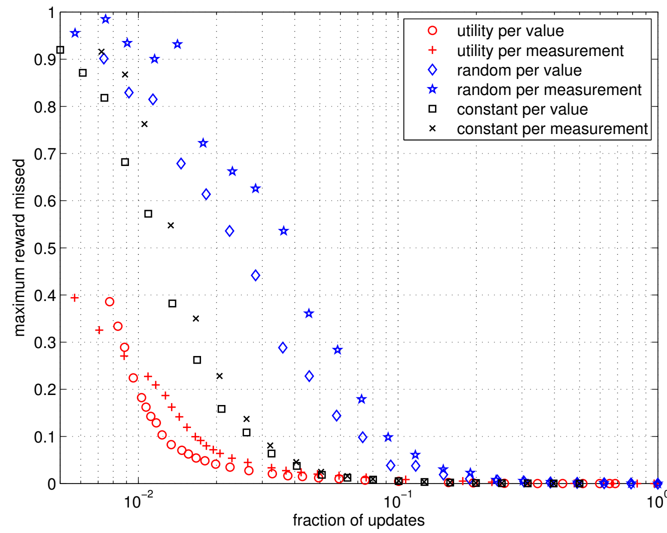


Figure 9: Maximum value missed plotted as function of the fraction of updates sent for utility-based, random and constant rate selection of updates.

REFERENCES

[1] C. M. Kreucher, K. Kastella, and A. O. H. III, "Sensor management using an active sensing approach," *Signal Processing*, vol. 85, no. 3, pp. 607–624, 2005.

[2] J. L. Williams, "Information theoretic sensor management," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[3] P. A. Ortega and D. A. Braun, "A conversion between utility and information," in *3rd Conference on Artificial General Intelligence*, March 2010.

[4] F. Bolderheij, "Mission-driven sensor management, analysis, design, implementation and simulation," Ph.D. dissertation, Delft University of Technology, 2007.

[5] A. Charlish, "Autonomous agents for multi-function radar resource management," Ph.D. dissertation, University College London, 2011.

[6] S. Eswaran, D. Shur, and S. Samtani, "A metric and framework for measuring information utility in mission-oriented networks," *Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 416–433, August 2011.

[7] P. Velagapudi, O. Prokopyev, K. Sycara, and P. Scerri, "Maintaining shared belief in a large multiagent team," in *10th International Conference on Information Fusion*, 2007, pp. 1–8.

[8] E. van Foeken, L. Kester, and M. van Iersel, "Real-time common awareness in communication constrained sensor systems," in *12th International Conference on Information Fusion*, July 2009, pp. 118 – 125.

[9] L. Kester, "Method for designing networking adaptive interactive hybrid systems," in *Interactive Collaborative Information Systems*, R. Babuska and F. Groen, Eds. Heidelberg: Springer-Verlag Berlin, 2010, vol. SCI 281, pp. 401 – 421.

[10] E. van Foeken and L. Kester, "Adaptive team formation for shared situation awareness," in *15th International Conference on Information Fusion*, July 2012, pp. 361–368.