A System to Support Attention Allocation: Development and Application*

Tibor Bosse[1], Rianne van Lambalgen[1], Peter-Paul van Maanen[1,2], and Jan Treur[1]
[1] *Vrije Universiteit Amsterdam, Amsterdam, The Netherlands*
{tbosse,rm.van.lambalgen,treur}@cs.vu.nl
[2] *TNO Human Factors, Soesterberg, The Netherlands*
peter-paul.vanmaanen@tno.nl

Abstract. This paper discusses and evaluates an agent model that is able to manipulate the visual attention of a human, in order to support naval crew. The agent model consists of four submodels, including a model to reason about a subject's attention. The model was evaluated based on a practical case study which was formally analysed and verified using automated checking tools. Results show how a human subject's attention is manipulated by adjusting luminance, based on assessment of the subject's attention. These first evaluations of the agent show a positive effect.

*Parts of this article appeared in the Proceedings of the Ninth IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'09) and the Proceedings of the Fifth International Workshop on Attention in Cognitive Systems (WAPCV'08).

Keywords: Human Experimentation, Attention, Interface Agent, Formal Analysis.

1. Introduction

In the domain of naval warfare, it is crucial for the crew of the vessels involved to be aware of the situation in the field. One of the crew members is usually assigned the task to identify and classify all entities in the environment (e.g., [10]). This task determines the type and intent of a multiplicity of contacts on a radar screen. Attention is typically directed to one bit of information at a time [21], [23], [25]. A supporting software agent may alert the human about a contact if it is ignored. To this end the agent has to maintain a model of the cognitive state of the human including the human's distribution of attention. Existing cognitive models on attention show that it is possible to predict a person's attention based on a saliency map, calculated from features of a stimulus, like luminance, colour and orientation [13], [20]. In this study, a Theory of Mind (or ToM, e.g., [6]) model is exploited within the agent model to analyse attention of the human. Attention can then be influenced (or 'manipulated') by changing features of stimuli, e.g., its contrast with stimuli at other locations [13], [15], [19], its luminance [24], [26], or its form [26].

Some approaches in the literature address the development of software agents with a Theory of Mind (e.g., [6], [16], [17]), but only address a model of the epistemic (e.g., beliefs), motivational (e.g., desires, intentions), and/or emotional states of other agents. For the situation sketched above, attribution of attentional states has to be addressed. In the current paper, an agent model has been developed, which uses four specific (sub)models. The first is a representation of a dynamical model of human attention, for estimation of the locations of a person's attention, based on information about features of objects on the screen and the person's gaze. The second model is a reasoning model which the agent uses to reason through the first model, to generate beliefs on attentional states at any point in time. With a third model the agent compares the output of the second model with a normative attention distribution and determines the discrepancy. Finally, a fourth model is used to direct the person's attention to relevant contacts based on the output of the third model.

Initial versions of the first two models were adopted from earlier work [7]. The current paper focuses on the use of the last two models, where input from [5] was

adopted. Section 2 gives a literature review on the manipulation of attention, Section 3 describes a formalisation of the different models, and in Section 4 the global behaviour of the model is tested by simulation experiments. In Section 5, the model is implemented in the context of a case study where a software agent is used to manipulate a subject's attention. Based on this case study, Section 6 addresses experimental validation of the results, and Section 7 addresses automated verification of different important properties of the submodels used in the agent. In Section 8, a formal mathematical analysis of the model is given. Finally, Section 9 is a discussion.

2. Manipulation of Attention

Typically, a person's attention is influenced both by top-down and by bottom-up processes. The former means that observers orient their attention in a goal-directed manner, as a consequence of their expectations or intentions [21]. For example, when searching for a friend in the crowd, attention is guided top-down [23]. In contrast, the latter means that attention is elicited by a (highly salient) trigger from the environment. For example, one green circle among several blue circles will "pop-out" and attention will be directed to this object [25]. In this project the focus is primarily on adjusting the features of a specific location, such that only bottom-up attention is manipulated. Features that are mainly known to influence attention are intensity (luminance), colour and orientation. Previous research shows that attention can be elicited both by the contrast with stimuli at other locations [13], [15], [19] and the abrupt change of a feature, like luminance [24], [26] or form [26].

Several cognitive models on attention have been proposed and show that it is possible to predict attention allocation based on a saliency map, calculated from features of a stimulus, like luminance, colour and orientation [14], [20]. Furthermore, other features like effort and expectancy have been incorporated in attention models [12], [27]. These proposed models are not dynamic in the sense that they take changes of information from the environment into account. However, if indeed the change of a specific feature (like luminance) can cause an attention shift in the human performing a task considered, a *support model* can be used to realise this change. This way, humans who have to direct their attention to a large number of locations in parallel can be supported to adequately perform their task.

Although much is known on the features that guide attention [13], [23], there are few other attempts to design a system for attention allocation support. Automated attention guidance has been investigated, by providing either a tactical cue [22] or a visual cue to a relevant location [11]. However, this automated cueing is based on features of the task (i.e. threat of an object) and not on the human's actual distribution of attention.

3. A Theory of Mind for Attention

3.1. Overall Setting

A Theory of Mind enables an agent to analyze another agent's mind, and to act according to the outcomes of such an analysis and its own goals. For the general case such processes require some specific facilities.

A *representation of a dynamical model* is needed describing the relationships between different mental states of the other agent. Such a model may be based on qualitative causal relations, but it may also concern a numerical dynamical system model that includes quantitative relationships between the other agent's mental states. In general such a model does not cover all possible mental states of the other agent, but focuses on certain aspects, for example on beliefs and desires, on emotional states, on the other agent's awareness states, or on attentional states as in this paper.

Furthermore, *reasoning methods to generate beliefs on the other agent's mental state* are needed to draw conclusions based on the dynamical model in (1) and partial information about the other agent's mental states. This may concern deductive-style reasoning methods performing forms of simulation based on known inputs to predict certain output, but also abductive-style methods reasoning from output of the model to (possible) inputs that would explain such output.

Moreover, when in one way or the other an estimation of the other agent's mental state has been found out, it has to be *assessed whether there are discrepancies* between this state and the agent's own goals. Here also the agent's self-interest comes in the play. It is analyzed in how far the other agent's mental state is in line with the agent's own goals, or whether a serious threat exists that the other agent will act against the agent's own goals.

Finally a *decision reasoning model* is needed to decide how to act on the basis of all of this information. Two types of approaches are possible. A first approach is to take the other agent's state for granted and prepare for the consequences to compensate for them as far as these are in conflict with the agent's own goals, and to cash them as far as they

can contribute to the agent's own goals (*anticipation*). For the navy case, an example of anticipation is when it is found out that the other agent has no attention for a dangerous object, and it is decided that another colleague or computer system will handle it (dynamic task reallocation). A second approach is not to take the other agent's mental state for granted but to decide to try to get it adjusted by affecting the other agent, in order to obtain a mental state of the other agent that is more in line with the agent's own goals (*manipulation*). This is the case addressed in this paper.

The general pattern sketched above is applied in this paper to the way in which a (software) agent can attempt to adjust the other (human) agent's attention, whenever required. To this end the software agent uses the following four different models: Dynamic Attention Model, Model for Beliefs about Attention, Model to Determine Discrepancy and Decision Model for Attention Adjustment. In this section, each of these models are described in detail. The agent and its interaction with the environment (involving a complex task and an eye-tracker, see Section 5) are schematically displayed in Figure 1.

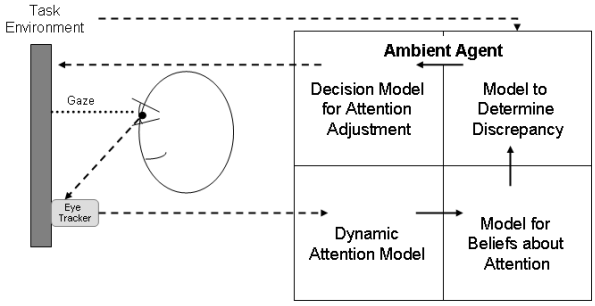

Figure 1. Overview of the ambient agent and its environment

3.2. Dynamic Attention Model

This model is taken over from [7] and is only briefly summarised in this section. The model uses three types of input: information about the human's *gaze direction*, about *locations* (or spaces) and about *features* of objects on the screen. Based on this, it makes an estimation of the *current attention distribution* at a time point: an assignment of attention values $AV(s,t)$ to a set of attention spaces at that time. The attention distribution is assumed to have a certain persistency. At each point in time the new *attention level* is related to the previous attention, by

$$AV(s,t) = \lambda \cdot AV(s,t-1) + (1-\lambda) \cdot AV_{norm}(s,t)$$

Here, λ is the decay parameter for the decay of the attention value of space s at time point $t-1$, and $AV_{norm}(s,t)$ is determined by *normalisation* for the total amount of attention $A(t)$, described by:

$$AV_{\text{norm}}(s,t) = \frac{AV_{\text{new}}(s,t)}{\sum_{s'} AV_{\text{new}}(s',t)} \cdot A(t)$$

$$AV_{\text{new}}(s,t) = \frac{AV_{\text{pot}}(s,t)}{1 + \alpha \cdot r(s,t)^2}$$

Here $AV_{new}(s,t)$ is calculated from the potential attention value of space s at time point t and the relative distance of each space s to the gaze point (the centre). The term $r(s,t)$ is taken as the Euclidian distance between the current gaze point and s at time point t (in the previous formula multiplied by an importance factor α which determines the relative impact of the distance to the gaze point on the attentional state, which can be different per individual and situation):

$$r(s,t) = d_{\text{eucl}}(gaze(t), s)$$

The potential attention value $AV_{pot}(s,t)$ is a weighted sum of the features of the space (i.e., of the types of objects present) at that time (e.g., luminance, colour):

$$AV\text{pot}(s,t) = \sum_{maps\ M} M(s,t) \cdot w_M(s,t)$$

For every feature there is a saliency map M, which describes its potency of drawing attention (e.g. [8], [13], [14]). Moreover, $M(s,t)$ is the unweighted potential attention value of s at time point t, and $w_M(s,t)$ is the weight used for saliency map M, where $1 \leq M(s,t)$ and $0 \leq w_M(s,t) \leq 1$.

Figure 2 shows an overview of this model. The circles denote the italicised concepts introduced above, and the arrows indicate influences between concepts.

3.3. Model for Beliefs about Attention

This (reasoning) model is used to generate beliefs about attentional states of the other agent. The software agent uses the dynamical system model as described in Section 3.2 as an internal simulation model to generate new attentional states from the previous ones, gaze information and features of the object, with the use of a forward reasoning method (forward in time) as described in [2]. The basic specification of the reasoning model can be expressed by the representation leads_to_after(I, J, D) (belief that I leads to J after duration D). Here, I and J are both information elements (i.e., they may correspond to any concept from Figure 2, e.g., gaze_at(1, 2) or has_value(av(1,2), 0.68).

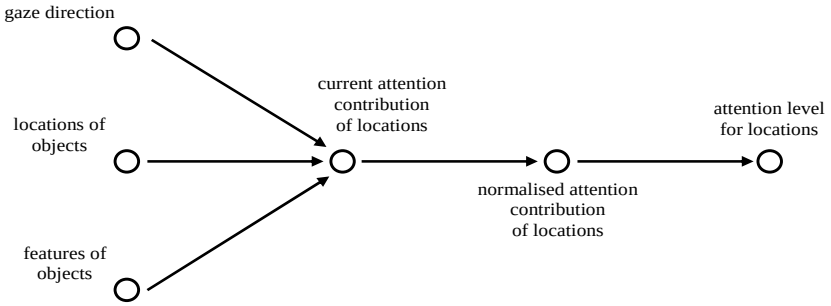

Figure 2. Overview of attention model

In addition, the representation at(I, T) gives information on the world (including human processes) at different points in time. It represents a belief that state I holds at time point T. For example, at(gaze_at(1,2), 53) expresses that at time point 53, the human's gaze is at the space with coordinates {1,2}.

3.4. Model to Determine Discrepancy

With this model the agent determines the discrepancy between actual and desirable attentional states and to what extent the attention distribution has to change. This is based on a model for the desirable attention distribution (prescriptive model). For the case addressed this means an assessment of which objects deserve attention (based on features as distance, speed and direction). To be able to make such assessments, the agent is provided with some tactical domain knowledge, in terms of heuristics (also see Section 5)

3.5. Decision Model for Attention Adjustment

The model for adjustment of the attention distribution has as input the discrepancy determined by the model described in Section 3.4, and also makes use of the explicitly represented dynamical model as described in Section 3.2. The general idea is that the relations between variables within this model are followed in a backward manner, thereby propagating the desired adjustment from the attentional state variable to the features of the object at the screen. The general pattern behind this operation on a dynamical model representation is illustrated in Figure 3. Here v_1 is the (desired) output of a model, and by branches the variables on which this depends are depicted, until the leaves where actual adjustments can be made. [1]

[1] For the moment, deterministic relationships between variables are assumed. However, in a later stage, the agent might learn such relationships.

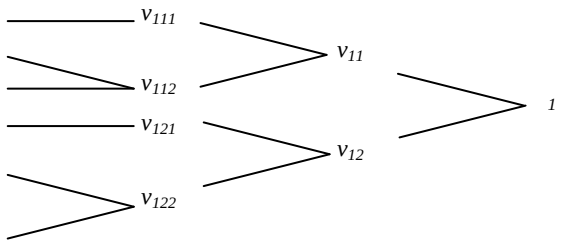

Figure 3. Dependencies between variables in a dynamical system model

This is a form of *desire refinement*: starting from the root variable, by a step-by-step process a desire on adjusting a parent variable is refined to desires on adjustments of the children variables, until the leave variables are reached. The starting point is the desire on the root variable, which is the desired adjustment of the attentional state; this is determined by.

belief(av(s)<h) ∧ desire(a(v)>h) ∧ belief(has_value(av(s), v)) → desire(adjust_by(av(s), (h-v)/v)

Note that here the adjustment is taken relative (expressed by division of the difference h-v by v). Suppose as a point of departure (given the discrepancy assessment) an adjustment Δv_1 is desired, and that v_1 depends on two variables v_{11} and v_{12} that are adjustable (the non-adjustable variables can be left out of consideration). Then by elementary calculus as a linear approximation the following relations between required adjustments can be obtained:

$$\Delta v_1 = \frac{\partial v1}{\partial v11} \Delta v_{11} + \frac{\partial v1}{\partial v12} \Delta v_{12}$$

This formula is used to determine the desired adjustments Δv_{11} and Δv_{12}, where by weight factors μ_{11} and μ_{12} the proportion can be indicated in which the variables should contribute to the adjustment: $\Delta v_{11}/\Delta v_{12} = \mu_{11}/\mu_{12}$.

$$\Delta v_1 = \frac{\partial v1}{\partial v11} \Delta v_{12}\, \mu_{11}/\mu_{12} + \frac{\partial v1}{\partial v12} \Delta v_{12} =$$

$$\left(\frac{\partial v1}{\partial v11}\,\mu_{11}/\mu_{12} + \frac{\partial v1}{\partial v12}\right)\Delta v_{12}$$

So the adjustments can be made as follows:

$$\Delta v_{12} = \frac{\Delta v1}{\frac{\partial v1}{\partial v11}\,\mu11/\mu12 + \frac{\partial v1}{\partial v12}}$$

$$\Delta v_{11} = \mu_{11}/\mu_{12}\,\frac{\Delta v1}{\frac{\partial v1}{\partial v11}\,\mu11/\mu12 + \frac{\partial v1}{\partial v12}} =$$

$$\frac{\Delta v1}{\frac{\partial v1}{\partial v11} + \frac{\partial v1}{\partial v12}\,\mu12/\mu11}$$

Special cases are $\mu_{11} = \mu_{12} = 1$ (*absolute equal contribution*) or $\mu_{11} = v_{11}$ and $\mu_{12} = v_{12}$ (*relative equal contribution*: in proportion with their absolute values). As an example, consider a variable that is just the weighted sum of two other variables (as is the case, for example, for the aggregation of the effects of the features of the objects on the attentional state):

$$v_1 = w_{11}v_{11} + w_{12}v_{12}$$

For this case

$$\frac{\partial v1}{\partial v11} = w_{11} \qquad \frac{\partial v1}{\partial v12} = w_{12}$$

and

$$\Delta v_{11} = \frac{\Delta v1}{w11 + w12\,\mu12/\mu11} \quad \Delta v_{12} = \frac{\Delta v1}{w11\,\mu11/\mu12 + w12}$$

For example when $\mu_{11} = \mu_{12} = 1$ this results in

$$\Delta v_{11} = \frac{\Delta v1}{w11 + w12} \qquad \Delta v_{12} = \frac{\Delta v1}{w11 + w12}$$

Assuming $w_{11} + w_{12} = 1$ in addition, this results in $\Delta v_{11} = \Delta v_{12} = \Delta v_1$.

Another setting, which actually has been used in the model is to take $\mu_{11} = v_{11}$ and $\mu_{12} = v_{12}$. In this case the adjustments are assigned proportionally; for example, when v_1 has to be adjusted by *5%*, also the other two variables on which it depends need to contribute an adjustment of *5%*. Thus the relative adjustment remains the same through propagations:

$$\frac{\Delta v11}{v11} = \frac{\frac{\Delta v1}{w11 + w12\,v12/v11}}{v_{11}} = \frac{\Delta v1}{w11v11 + w12\,v12} = \frac{\Delta v1}{v1}$$

This shows the general approach on how desired adjustments can be propagated in a backward manner through a dynamical model. Thus a desired adjustment of the attentional state as output at some point in time can be related to adjustments in the features of the displayed objects as inputs at previous points in time. For the case study undertaken this approach has been applied, although at some points in a simplified form. One of the simplifications made is that due to the linearity of most dependencies in the model, adjustments have been used that just propagate without any modification. An example of a rule specified to achieve this propagation process is:

desire(adjust_by(u1, a)) ∧ belief(depends_on(u1, u2)) → desire(adjust_by(u2, a))

Here the adjustments are taken relative, so, this rule is based on $\Delta u_2 / u_2 = \Delta u_1 / u_1$ as derived above for the linear case. When at the end the leaves are reached, which is represented by the belief that they are directly adjustable, then from the desire an intention to adjust them is derived.

desire(adjust_by(u, a)) ∧ belief(directly_adjustable(u)) → intention(adjust_by(u, a))

If an intention to adjust a variable u by a exists with current value b, the new value b+ α*a*b to be assigned to u is determined; here α is a parameter that allows the modeller to tune the speed of adjustment:

intention(adjust_by(u, a)) ∧ belief(has_value_for(u, b)) → performed(assign_new_value_for(u, b+ α*a*b))

This rule is applied for variables that describe features f of objects at locations s, i.e., instances for u of the form feature(s, f). Note that each time the adjustment is propagated as a value relative to the overall value.

4. Simulation results

To test whether the approach described above yields the expected behaviour, it has been used to perform a number of simulation experiments in the LEADSTO simulation environment [4]. This environment takes a specification of causal relationships (in the format as shown in the previous sections) as input, and uses this to generate simulation traces. The simulations shown here address a slightly simplified case, where the radar screen has been split up in 4 locations. For the time being, it is assumed that each location contains one contact, and that these contacts stay within their locations.

The features of the contacts that are manipulated are luminance, size, and level of flashing. Initially, each contact starts with the same features, but during the simulation these features are manipulated, based on the prescribed (or desired) attention. This desired attention is generated randomly, where every 50 time units a

next location is selected where the attention should be. Furthermore, the behaviour of the human gaze is generated as follows: after each adaptation of the features, the gaze moves to one of the four locations, with a probability that is proportional to the saliency of the contact at that location.

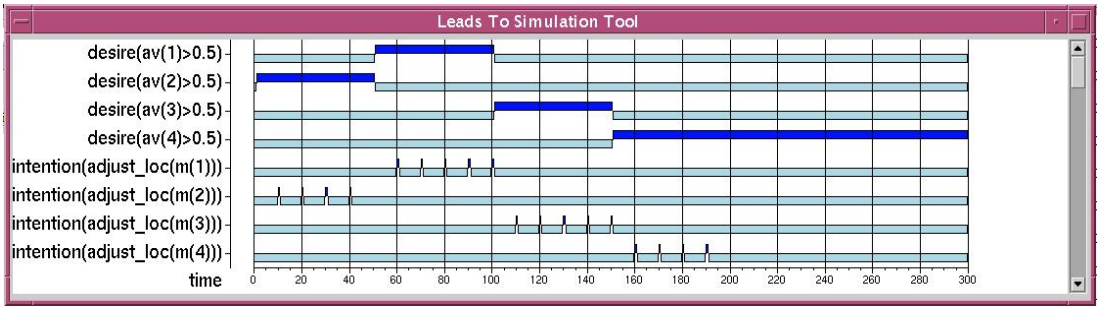

Figure 4. Model-based reasoning process. First it is intended (several times) to adjust a feature value at location 2, then at location 1, then at location 3, and finally at location 4.

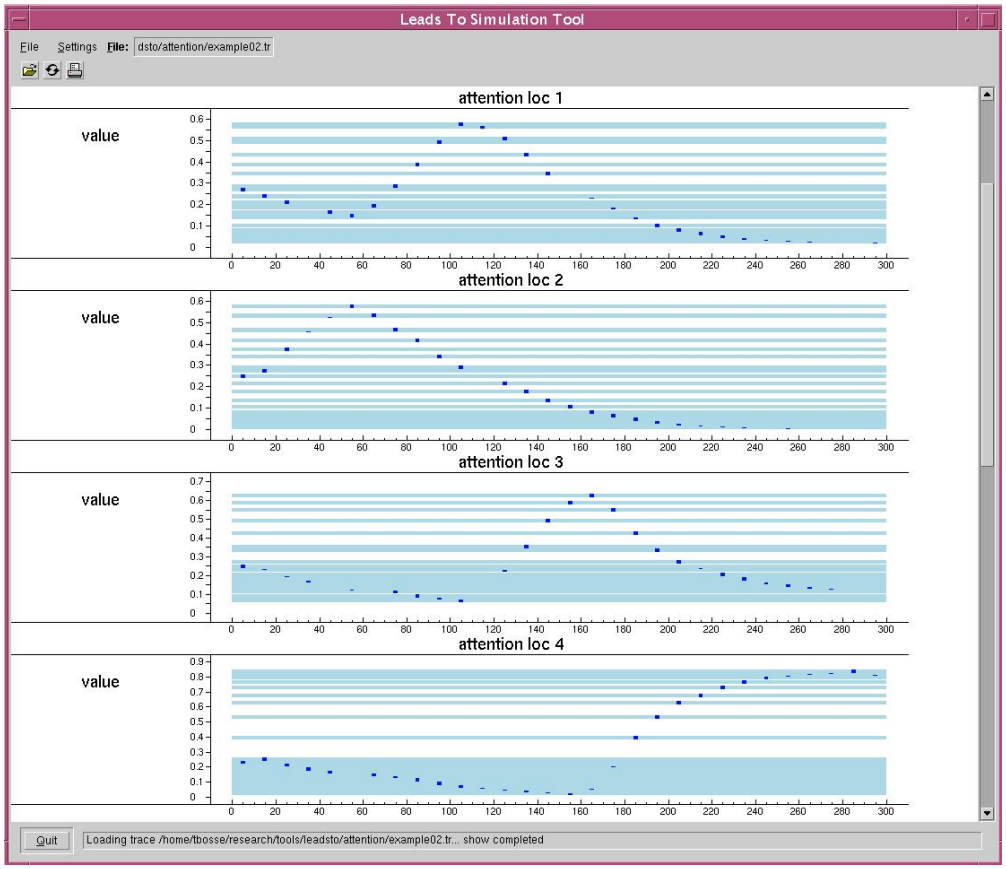

Figure 5. Estimated attention at different locations. Initially the highest attention value is estimated to be at location 2 (with a peak around time point 55), then at location 1, then at location 3, and finally at location 4.

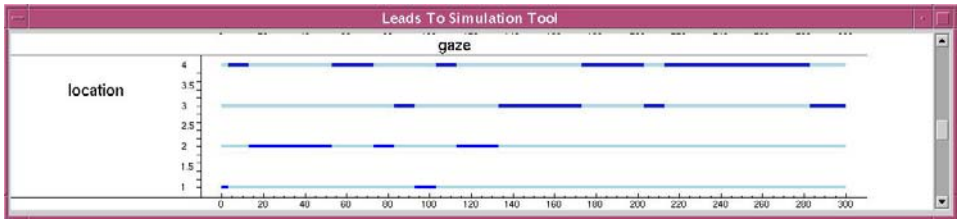

Figure 6. Dynamics of gaze. The vertical axis denotes the location of the gaze, which switches between location 1, 2, 3, and 4.

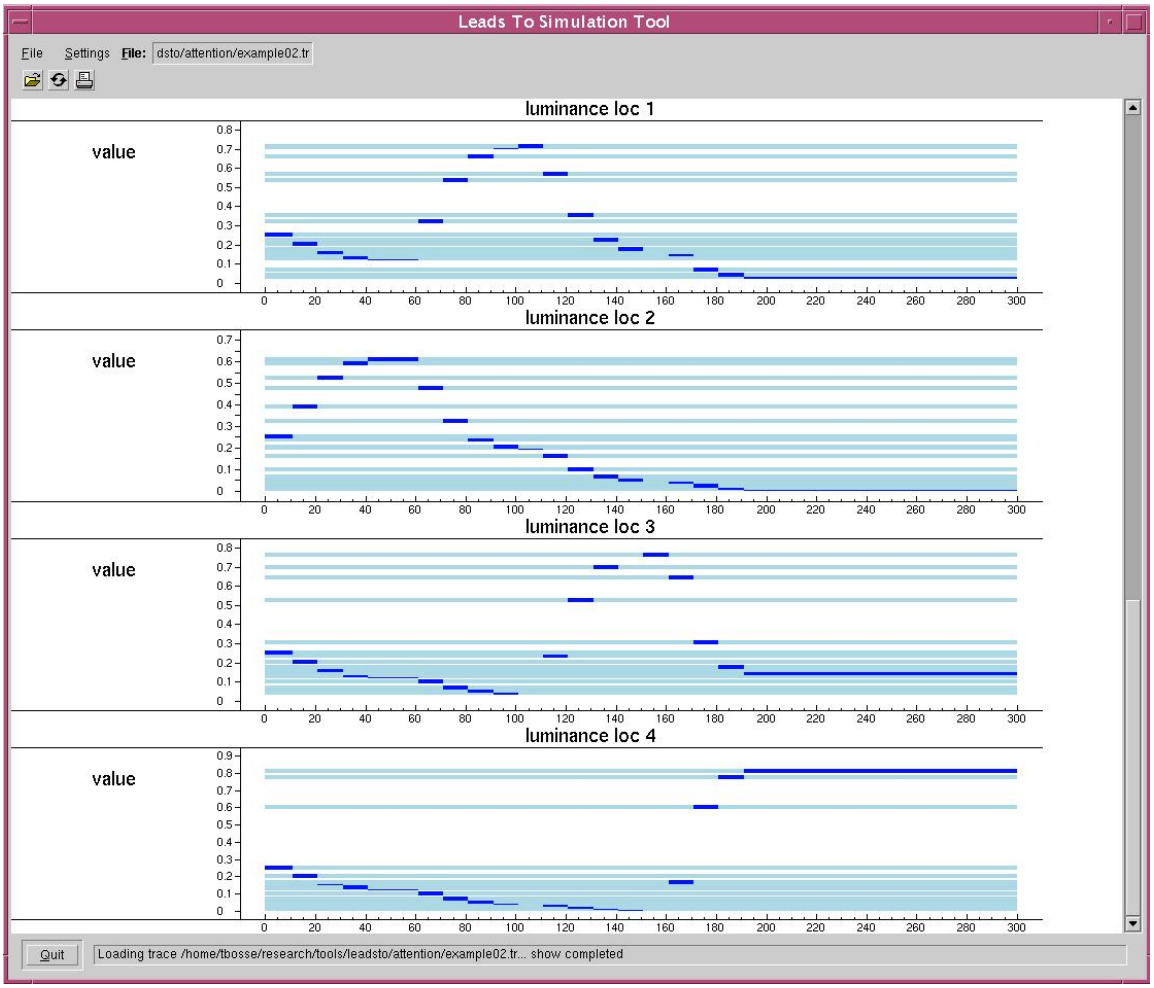

Figure 7. Values of feature 'luminance' at different locations. First the luminance at location 2 is increased, then at location 1, 3, and 4 (note that values are normalised).

The results of an example simulation run are depicted in Figures 4 to 7. In these figures, time is on the horizontal axis, and the different state of the process is shown in the vertical axis. A dark line indicates that a state is true at a certain time point. Note that some information has been omitted due to space limitations. Figure 4 shows the model-based reasoning process of the agent, in terms of desires and intentions. Figures 5, 6, and 7 show, respectively, the estimated attention, the human's gaze, and the value of the feature "luminance" at different locations over time. As shown in Figure 4, initially it is desired that at least 50% of the human's attention is at location 2 (desire(av(2)>0.5)). Since this is not the case (see Figure 5), the luminance of the contact at location 2 is increased (see Figure 6). As a result, the human's gaze shifts towards this location

(see Figure 6), which increases his attention for location 2. In the rest of the simulation, this pattern is repeated for different locations.

After successfully running simulations of the models under a number of different parameter settings, it was considered appropriate to be implemented in a real world case study. This case study is described in the next section.

5. Case Study

The different models have been implemented and tested for a case study. The used case study mimics a real-world situation, with human subjects executing the Tactical Picture Compilation Task. In Section 5.1 the environment is shortly explained. Section 5.2 discusses some implementation details of the attention manipulating agent tailored to the environment.

5.1. Environment

The task used for this case study is an altered version of the identification task described in [8] that has to be executed in order to build up a tactical picture of the situation, i.e. the Tactical Picture Compilation Task (TPCT). The implementation of the software was done in Gamemaker [29].

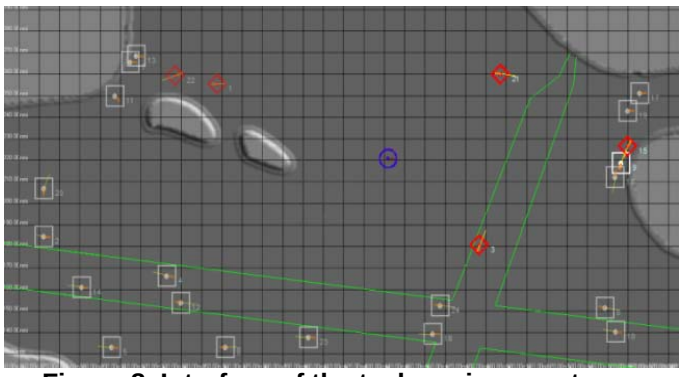

Figure 8. Interface of the task environment

In Figure 8 a snapshot of the interface of the task environment is shown. The goal is to identify the five most threatening contacts (ships). In order to do this, participants monitor a radar display of contacts in the surrounding areas. To determine if a contact is a possible threat, different criteria have to be used. These criteria are the identification criteria (idcrits) that are also used in naval warfare, but are simplified in order to let naive participants learn them more easily. These simplified criteria are the speed (depicted by the length of the tail of a contact), direction (pointer in front of a contact), distance of a contact to the own ship (circular object), and whether the contact is in a sea lane or not (in or out the large open cross). Contacts can be identified as either a threat (diamond) or no threat (square).

5.2 Implementation

The support agent was further developed and evaluated using Matlab (see Appendix A). The output of the environment described in Section 5.1 was used and consisted of a representation of all properties of the contacts visible on the screen, i.e. speed, direction, whether it is in a sea lane or not, distance to the own ship, location on the screen and contact number. In addition, data from a Tobii x50 eye-tracker [28] were retrieved from a participant executing the TPC task. All data were retrieved several times per second and were used as input for the models within the agent. Once the agent models were tailored to the TPC case study, the eventual implementation of them was done in C#.

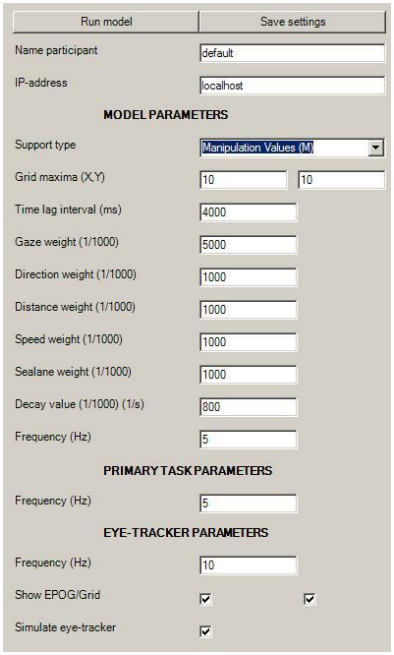

Figure 9. Interface of the attention allocation support system

In Figure 9 the interface of the implemented agent models is shown. This interface consists of four parts where parameters can be set. In first part the agent models can be run. Once the button is pushed, both the input from the TPC task environment and the eye-tracker are retrieved and the required saliency levels are communicated back to the TPC task environment. Also the current settings can be saved; the participant's name and the IP-address where the TPC task

environment is running and eye-tracker is connected can be specified here. In the second part, agent model parameters can be set. For this paper we used a type of support where feature manipulation values are to be communicated to the task environment. These values cause the saliency of the different objects on the screen to either increase or decrease, which may result in a shift of the participant's visual attention. As a result, the participant's attention is continuously manipulated in such a way that it is expected that he pays attention to the objects that are considered relevant by the agent. The increase or decrease of the saliency of objects can be done on a continuous or discrete scale, with a binary scale as being discrete. Other types of support can also be set with the support type parameter, such as the estimated threat values of each contact. Furthermore the grid size can be set here. The more fine grained the grid, the more computationally intensive the running of the agent models will be. Time lag is also set here which determines how old (in terms of milliseconds) data is allowed to be in order to be used by the agent models. This is needed because the application is run over a network (though 4 seconds is most likely never reached). The weights together with the decay are the same parameters as also described in Section 3.2. The frequency determines the amount of model loops the agent is allowed to run. The higher the frequency, the more computationally intensive the support agent will be. The third part deals with the frequency of the task environment. This specifies the amount of times the information from the task environments is communicated to the support agent. The fourth part deals with the parameters of the eye-tracker. The frequency specifies the amount of times the gaze location is retrieved. The other options are to visualise the eye-tracker information in real-time or to simulate eye-tracker information by mouse movements instead of gaze behaviour.

5.3. Results

The first results of the agent implemented for this case study are best described by a number of example snapshots of the outcomes of the models used in the agent to estimate (model 1) and manipulate (model 4) attention in three different situations over time (see Figure 10).

On the left side of Figure 10 the darker dots correspond to the agent's estimation of those contacts to which the participant is paying attention. On the right side of the figure, the darker dots correspond to those contacts where attention manipulation is initiated by the system (in this case, by increasing its saliency). On both sides of the figure a cross corresponds to the own ship, a star corresponds to the eye point of gaze, and the x- and y-axes represent the coordinates on the interface of the TPCT. In the pictures to the left, the z-axis represents the estimated amount of attention.

The darker dots on the left side are a result of the exceedance of this estimation of a certain threshold (in this case .03). Thus, a peak indicates that it is estimated that the participant has attention for that location.

Furthermore, from top to bottom, the following three situations are displayed in Figure 10:

- After 37 seconds since the beginning of the experiment, the participant is not paying attention to region A at coordinates (7.5,1.5) and no attention manipulation for region A is initiated by the system.
- After 39 seconds, the participant is not paying attention to region A, while the attention should be allocated to region A, and therefore attention manipulation for region A is initiated by the system.
- After 43 seconds, the participant is paying attention to region A, while no attention manipulation for region A is done by the system, because this is not needed anymore.

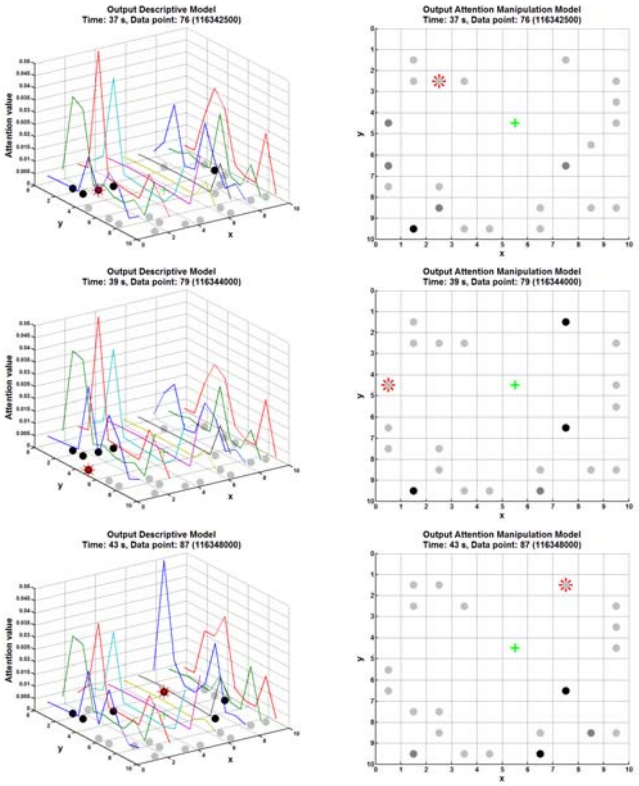

Figure 10. Estimation of the participant's attention division and the agent's reaction

The output of the attention manipulation system and the resulting reaction in terms of the allocation of the participant's attention in the above three situations, show what one would expect of an accurate system of attention manipulation. As shown in the two pictures at

the bottom of Figure 10, in this case the agent indeed succeeds in attracting the attention of the participant: both the gaze (the star in the bottom right picture) and the estimated attention (the peak in the bottom left picture) shift towards the location that has been manipulated.

6. Validation

In order to validate the agent's manipulation model, the results from the case study have been used and tested against results that were obtained in a similar setting without manipulation of attention. The basic idea was to show that the agent's manipulation of attention indeed results in a significant improvement of human performance. Human performance in selecting the five most threatening contacts was compared during two periods of 10 minutes (with and without manipulation, respectively). The type of manipulation was based on determining the saliency of the objects on a binary scale. In this way it was easy (opposed to a continuous scale) to follow the agent's advice. The performance measure took the severity of an error into account. Taking the severity into account is important, because for instance selecting the least threatening contact as a threat is a more severe error than selecting the sixth most threatening contact. This was done by the use of the following penalty function (P):

$$P_x = \frac{p_x}{\sum_k^{24} p_k} = \frac{abs(t_x - \frac{t_5 + t_6}{2})}{\sum_k^{24} p_k}$$

where p_x is the prenormalised penalty of contact x and t_x is the threat value of contact x (there are 24 contacts). Human performance is then calculated by adding all penalties of the contacts that are incorrectly selected as one of the top five threats and subtracting them from 1.

After the above alterations, the average human performance over all time points of the condition "support" was compared with the average human performance of the first condition "no support", where "support" ($M+ = .8714$, $SD+ = .0569$) was found significantly higher (i.e., $p < .05$) than "no support" ($M- = .8541$, $SD- = .0667$), with $t(df = 5632) = 10.46$, $p < 0.001$. Hence significant improvements were found comparing the first and the second condition. Finally, subjective data based on a questionnaire pointed out that the participant preferred the "support" condition above that of the "no support" condition.

7. Verification

In addition to this validation, the results of the experiment have been analysed in more detail by converting them into formally specified *traces* (i.e., sequences of events over time), and checking relevant *properties*, expressed as temporal logical expressions, against these traces. To this end, a number of properties were logically formalised in the language TTL [3]. This predicate logical language supports formal specification and analysis of dynamic properties. TTL is built on atoms referring to states of the world, time points and traces, i.e. trajectories of states over time. In addition, dynamic properties are temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner. Given a trace γ over state ontology Ont, the state in γ at time point t is denoted by state(γ, t). These states can be related to state properties via the formally defined satisfaction relation denoted by the infix predicate |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t) |= p denotes that state property p holds in trace γ at time t.

Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic, using quantifiers over time and traces and the usual first-order logical connectives such as ¬, ∧, ∨, ⇒, ∀, ∃. To give a simple example, the property 'there is a time point t in trace 1 at which the estimated attention level of space {1,2} is 0.5' is formalised as follows (see [3] for more details):

∃t:TIME state(trace1,t) |= belief(has_value(av(1,2), 0.5))

Below, a number of such dynamic properties that are relevant to check the agent's attention manipulation are formalised in TTL, in a similar manner as was done in [6][2] To this end, some abbreviations are defined:

discrepancy_at(γ:TRACE, t:TIME, x,y:COORDINATE) ≡
  ∃a,h:REAL estimated_attention_at(γ,t,x,y,a) &
  state(γ,t) |= desire(has_value(av(x,y), h)) & a<h

This predicate states that at time point t in trace γ, there is a discrepancy at space {x,y}. This is the case when the estimated attention at this space is smaller

[2] Note that the properties introduced in [6] were used mainly to check whether the attention model (as described in Section 3.2) behaved correctly, whereas the current properties aim to check for successfulness of the attention manipulation model.

than the desired attention. Next, abbreviation estimated_attention_at is defined:

estimated_attention_at(γ:TRACE,t:TIME,
x,y:COORDINATE, a:REAL)
≡ state(γ,t) |= belief(has_value(av(x,y),a))

This takes the estimated attention as calculated by the agent at runtime. This means that this definition can only be used under the assumption that this calculation is correct. Since this is not necessarily the case, a second option is to calculate the estimated attention during the checking process, based on more objective data such as the gaze data and the features of the contacts.

Based on these abbreviations, several relevant properties may be defined. An example of a relevant property is the following (note that this property assumes a given trace γ, a given time point t, and a given space {x,y}):

PP1 (Discrepancy leads to Efficient Gaze Movement)
If there is a discrepancy at {x,y} and the gaze is currently at {x2,y2}, then within δ time points the gaze will have moved to another space {x3,y3} that is closer to {x,y} (according to the Euclidean distance).

PP1(γ:TRACE, t:TIME, x,y:COORDINATE) ≡
∀x2,y2:COORDINATE
discrepancy_at(γ,t,x,y) &
state(γ,t) |= gaze_at(x2,y2) & t < LT-δ
⇒ ∃t2:TIME ∃x3,y3:COORDINATE [t<t2<t+δ &
state(γ,t2) |= gaze_at(x3,y3) &
$\sqrt{((x-x2)^2+(y-y2)^2)} > \sqrt{((x-x3)^2+(y-y3)^2)}$]

In the above property, a reasonable value should be chosen for the delay parameter δ. Ideally, δ equals the sum of 1) the time it takes the agent to adapt the features of the contacts and 2) the person's reaction time.

To enable automated checks, a special software environment for TTL exists, featuring both a Property Editor for building and editing TTL properties and a Checking Tool that enables formal verification of such properties against traces [3]. Using this TTL Checking Tool, properties can be automatically checked against traces generated from any case study. In this paper the properties were checked against the traces from the experiment described in Section 5. When checking such properties, it is useful to know not only if a certain property holds for a specific space at a specific time point in a specific trace, but also how often it holds. This will provide a measure of the successfulness of the system. To check such more statistical properties, TTL offers the possibility to test a property for all time points, and sum the cases that it holds. Via this approach, PP1 was checked against the traces of the experiment with δ = 3.0 sec. These checks pointed out that (under the "support" condition) in 88.4% of the cases that there was a discrepancy, the gaze of the person changed towards the location of the discrepancy. Under the "no support" condition, this was around 80%.

8. Formal Analysis

The results of validation and verification discussed above may ask for a more detailed analysis. In particular, the question may arise of how a difference between 80% without support and 88% with support as reported above should be interpreted. Here a more detailed formal analysis is given that supports the context for interpretation of such percentages. To this end the effect of arbitrary transitions in gaze dynamics is analysed, in particular those that occur between the time points of monitoring the gaze and adjustment of luminance.

At a given time point, the adjustment of luminance is based on the gaze at that point in time. A question is whether at the time the luminance is actually adjusted, the gaze is still at the same point. When the system is very fast in adjusting the luminance this may be the case. However, it is also possible that even in this very short time the gaze has changed to focus on another location on the screen. Here it is analysed in how many cases of an arbitrarily changed gaze the luminance adjustment by the system should still be sufficient. The general idea is that this is the case as long as the gaze transition does not increase the distance between gaze location and considered discrepancy location. The area of all locations of the screen for which this is the case is calculated mathematically below; here the worst case is analysed, the case when the considered discrepancy location is at the corner of the screen. The screen is taken as a square. The function f indicates an under-approximation of the number (measured by the area) of locations with distance at most r to O (see Figure 11, with $r = OQ$). For $r \leq d$ the area within distance r to O is a quarter of a circle: $\pi/4\, r^2$; so $f(r) = \pi/4\, r^2$, for $r \leq d$. For $r > d$ an approximation was made. The part of distance to O larger than r is approximated by two triangles as ΔPQR in Figure 10.

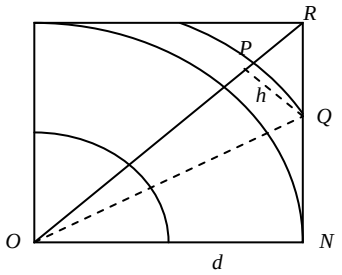

Figure 11. Gaze area approximation

$ON = d, QN = \sqrt{r^2 - d^2}$,
$QR = RN - QN = d - \sqrt{r^2 - d^2}$,
$OR = \sqrt{2} \cdot d$, $PR = OR - OP = \sqrt{2} \cdot d - r$,
$\Delta PQR = \frac{1}{2} PR \cdot h$, with h the distance of Q to OR,

$$h = \frac{1}{2}\sqrt{2} \cdot QR = \frac{1}{2}\sqrt{2} \cdot \left(d - \sqrt{r^2 - d^2}\right).$$

The whole area $-2\Delta PQR$ is

$$d^2 - \frac{1}{2}\sqrt{2} \cdot \left(d - \sqrt{r^2 - d^2}\right)\left(\sqrt{2} \cdot d - r\right)$$

Therefore, for $r > d$, it is taken

$f(r) = d^2 - \frac{1}{2}\sqrt{2} \cdot \left(d - \sqrt{r^2 - d^2}\right)\left(\sqrt{2} \cdot d - r\right)$, for $r > d$

For $d = 10$ the overall function f divided by the overall area d^2 (thus normalising it between 0 and 1) is shown in Figure 12. For example, it shows that when $r = \frac{1}{2}d$, then the covered area is around 20% of the overall screen, but when r is a bit larger, for example $r = d$, then at least around 80% is covered. Note that this is a worst case analysis with the location considered in the corner. In less extreme cases the situation can differ. When, for example, the considered location is at the center, then for distance $r = \frac{1}{2}d$, the covered area would be a full circle with radius $\frac{1}{2}d$, so an area of $\pi/4\, d^2$, which is more than 70% of the overall area.

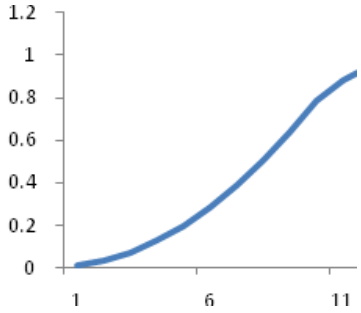

Figure 12. Function of the number of locations within distance r to O, divided by d^2, for $d = 10$

Moreover, the distance of the considered location where a discrepancy is detected to the actual gaze may not have a uniform probability distribution from 0 to $\sqrt{2} \cdot d$. Indeed, the value 0 may be very improbable, and the larger values may have much higher probabilities. Suppose $p(r)$ denotes the probability (density) that the distance between actual gaze and considered discrepancy location is r, then the expected coverage can be calculated by:

$$\int_0^{\sqrt{2} \cdot d} p(r) \cdot f(r) dr$$

For example, if a probability distribution is assumed that is increasing in a modest, linear way from $p(0) = 0$ to $p(\sqrt{2} * d) = 1/d^2$, then for $d = 10$ with $p(r) = r/100$ this becomes approximately (estimated by numerical integration):

$$\int_0^{14} \frac{r \cdot f(r)}{100} dr = 0.72$$

This means that the expected coverage would be 72%. For a bit less modest increase, for example in a quadratic manner for $d = 10$ from $p(0) = 0$ to $p(14) = 0.2$, then the expected coverage is approximately 80% (estimated by numerical integration):

$$\int_0^{14} \frac{r^2 \cdot f(r)}{1000} dr = 0.80$$

When it turns out that the gaze is often changing, then a remedy is to base the adjustment of the luminance on a larger distance for r, thus anticipating on the possible future states. The graph for f shows that if r is taken equal to distance d, then a coverage of 80% is achieved.

9. Discussion

An important task in the domain of naval warfare is the Tactical Picture Compilation Task, where persons have to deal with a lot of complex and dynamic information at the same time. To obtain an optimal performance, an intelligent agent can provide aid in such a task. This paper discussed and evaluated an initial version of such a supporting software agent. Within this type of agent an explicitly represented model of human functioning plays an important role, for the case considered here the model of the human's attention.

To obtain a software agent for these purposes, four models were used that are aimed at manipulating a person's attention at a specific location: (1) a dynamical system model for attention, (2) a reasoning model to generate beliefs about attentional states using the attention model for forward simulation, (3) a discrepancy assessment model, and (4) a decision reasoning model, again using the attention model, this time for backward desire propagation. The first two

models were adopted from earlier work [7], and the decision model in (4) from [5].

After testing the models via simulation experiments, they have been implemented within an ambient agent, in a case study where participants perform a simplified version of the Tactical Picture Compilation Task. Within this case study an experiment was conducted to validate the agent's manipulation. The participants, both in the experiment discussed in this paper as well in earlier pilot studies, reported to be confident that the agent's manipulation indeed is helpful. The results of the validation study with respect to performance improvement have also been positive.

Further investigation has to be done in order to rule out any order effects, which suggests more research with more participants. It is also expected that future improvements of the agent's submodels, based on the gained knowledge from automated verification will also contribute to the improved success of such validation experiments.

A detailed analysis and verification of the behaviour of the agent also provided positive results. Traces of the experiment were checked to see whether the agent was able to adapt the features of objects in such a way that they attracted human attention. Results show that when there was a discrepancy between the prescriptive and the descriptive model of attention, the agent indeed was able to attract the human's attention.

Note that the model in this paper assumes mainly a bottom-up influence on attention to a location (i.e. influence of saliency). An existing model that incorporates both bottom-up and top-down aspects of attention is that of [11]. Next to the saliency of a location, their model predicts attention taking into account the expectancy of seeing a valuable (important) event at a location and the effort it takes to contribute attention at that location (see also [27]).

Although top-down influences are not taken into account in the current model, previous research shows that it is possible to extend such models based on a saliency map with top-down features of attention. In [9], [18], a map is proposed that shows the relevancy of a location to the task (task-relevance map) next to the existing saliency map. As our attention model is based on the generic notion of features of a location, it can be easily extended with top-down features as well. In the future, these possibilities will be explored in detail.

Acknowledgments

This research was partly funded by the Royal Netherlands Navy (program number V524).

References

[1] Baron-Cohen, S. Mindblindness: an essay on autism and theory of mind. MIT Press, 1995.

[2] Bosse, T., Both, F., Gerritsen, C., Hoogendoorn, M., and Treur, J. Model-Based Reasoning Methods within an Ambient Intelligent Agent Model. In: M. Mühlhäuser et al. (eds.), Constructing Ambient Intelligence: AmI-07 Workshops Proceedings. LNCS, vol. 11, Springer Verlag, 2008, pp. 352-370.

[3] Bosse, T., Jonker, C.M., Meij, L. van der, Sharpanskykh, A., and Treur, J., Specification and Verification of Dynamics in Agent Models. *International Journal of Cooperative Information Systems*, vol. 18, 2009, pp. 167 - 193. Shorter version in: Nishida, T. et al. (eds.), Proc. of the Sixth Int. Conference on Intelligent Agent Technology, IAT'06. IEEE Computer Society Press, 2006, pp. 247-254.

[4] Bosse, T., Jonker, C.M., Meij, L. van der, & Treur, J. (2007). A Language and Environment for Analysis of Dynamics by Simulation. *International Journal of Artificial Intelligence Tools*, vol. 16, no. 3, pp. 435-464.

[5] Bosse, T., Lambalgen, R. van, Maanen, P.-P. van, Treur, J., Automated Visual Attention Manipulation. In: L. Paletta, Tsotsos, J.K. (eds.), *Attention in Cognitive Systems, Proceedings of the Fifth International Workshop on Attention in Cognitive Systems, WAPCV'08*. Lecture Notes in Artificial Intelligence, vol. 5395. Springer Verlag, 2009, pp. 257-272.

[6] Bosse, T., Memon, Z.A., and Treur, J., A Two-Level BDI-Agent Model for Theory of Mind and its Use in Social Manipulation. In: Proceedings of the AISB 2007 Workshop on Mindful Environments, 2007, pp 335-342.

[7] Bosse, T., Maanen, P.-P. van, Treur, J., Simulation and Formal Analysis of Visual Attention, Web Intelligence and Agent Systems Journal, vol. 7, 2009, pp. 89-105.

[8] Chen, L.Q., Xie, X., Fan, X., Ma, W.Y., Zhang, H.J., and Zhou, H.Q., A visual attention model for adapting images on small displays, *ACM Multimedia Systems Journal*, 2003.

[9] Elazari, L., & Itty, L. (2010). A Bayesian model for efficient visual search and recognition. *Vision Research 50*. 1338-1352.

[10] Heuvelink, A., Both, F.: Boa: A cognitive tactical picture compilation agent. In: Proc. of the Int. Conf. on Intelligent Agent Technology, IAT '07, IEEE Comp. Soc. Press, 2007.

[11] Horrey, W.J., Wickens, C.D., Strauss, R., Kirlik, A., & Stewart, T.R. Supporting situation assessment through attention guidance and diagnostic aiding: the benefits and cost of display enhancement on judgment skill. In: Kirlik, A. (ed.) *Human-Technology Interaction*, Oxford University Press, New York, 2006.

[12] Gore, B.F., Hooey, B.L., Wickens, C.D., & Scott-Nash, S. A computational implementation of a human attention guiding mechanism in MIDAS v5. In: Duffy, V.G. (Ed.) *Digital Human Modeling, HCII 2009*, LNCS 5620, 237-246, 2009. Springer-Verlag Berlin Heidelberg, 2009.

[13] Itti, L. and Koch, C., Computational Modeling of Visual Attention, Nature Reviews Neuroscience, Vol. 2, No. 3, 2001, pp. 194-203.

[14] Itti, L., Koch, U., and Niebur, E. A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 1998, pp. 1254-1259.

[15] Levitt, J.B., and Lund, J.S. Contrast dependence of contextual effects in primate visual cortex. Nature, 387, 1997, 73-76.

[16] Marsella, S.C., Pynadath, D.V., and Read, S.J., PsychSim: Agent-based modeling of social interaction and influence. In: Lovett, M., Schunn, C.D., Lebiere, C., and Munro, P. (eds.), Proc. of the Int. C. on Cognitive Modeling, ICCM 2004, pp. 243-248 Pittsburg, Pensylvania, USA.

[17] Memon, Z.A., and Treur, J., Cognitive and Biological Agent Models for Emotion Reading. In: Jain, L., Gini, M., Faltings, B.B., Terano, T., Zhang, C., Cercone, N., Cao, L. (eds.), *Proceedings of the 8th IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT'08*. IEEE Computer Society Press, 2008, pp. 308-313.

[18] Navalpakkam, V., & Itti, L. A goal oriented attention guidance model. In: *Lecture Notes Computer Science* 2525, pp453-461. November 2002.

[19] Nothdurft, H. Salience from feature contrast: additivity across dimensions. Vision Research 40, 2000, 1183-1201.

[20] Parkurst, D., Law, K., and Niebur, E. Modeling the role of salience in the allocation of overt visual attention. Vision Research, 42, 2002, 107-123.

[21] Posner, M. E., Orienting of attention, Q. J. Exp. Psychol., 32, 1980, pp. 3-25.

[22] Sklar, A.E., & Sarter, N.B. (1999). Good Vibrations: tactile feedback in support of attention allocation and human-automation coordination in event-driven domains. *Human Factors 41(4)*, pp. 543-552.

[23] Theeuwes, J., Endogenous and exogenous control of visual selection, Perception, 23, 1994, pp. 429-440.

[24] Theeuwes, J. Abrupt luminance change pops out; abrupt color change does not, Perception & Psychophysics, 57(5), 1995, 637-644.

[25] Treisman, A. Features and objects: The 14th Bartlett memorial lecture. Q.J. Experimental Psychology A. 40, 201-237.

[26] Turatto, M., and Galfano, G., Color, form and luminance capture attention in visual search. Vision Research, 40, 2000, 1639-1643.

[27] Wickens, C.D., McCarley, J.S., Alexander, A.L., Thomas, L.C., Ambinder, M., & Zheng, S. Attention-Situation Awareness (A-SA) model of pilot error. In: Foyle, D.C. and Hooey, B.L. (eds.). *Human Performance Modeling in Aviation, 2008*, pp 213-239 Taylor & Francis Group, Florida.

[28] http://www.tobii.com.

[29] http://www.yoyogames.com/gamemaker

Appendix A: Matlab code attention model

```
% This matlab code calculates the attention values for each time step of a given data set:
% v_gaze2 (gaze data)
% v_target4, v_target5a (task environment data)
% v_increments (gaze to target data offset conversion data)
% a_parameters (model parameters)
% v_model4 (output data)

% constants initialization
xstep = 20; % x-pixel length in grid
ystep = 20; % y-pixel length in grid
c_gridmaxX = 10;
c_gridmaxY = 10;
c_timeinterval = 500; % number of milliseconds per time step
a_participantnumber = 1;
i_steps = 2000; % number of time steps
v_model1 = zeros(i_steps, c_gridmaxX, c_gridmaxY); % momentaneous
v_model2 = zeros(i_steps, c_gridmaxX, c_gridmaxY); % normalized (1)
v_model3 = zeros(i_steps, c_gridmaxX, c_gridmaxY); % temporal
v_model4 = zeros(i_steps, c_gridmaxX, c_gridmaxY); % normalized (2)

% if gazeweight = 0 then distance between de EPOG and the contact does not
% have any effect, if gazeweight = 'infinity' only the grid coordinates of
% de EPOG will have saliency
gazeweight = 1;

% if decayfactor = 0, then old information is not used
% if decayfactor = 1, then new information is not used
decayfactor = .8;

% initialization model
v_model4(1,:,:) = 1/(c_gridmaxX*c_gridmaxY);
v_model1(1,:,:) = v_model4(1,:,:);
v_model2(1,:,:) = v_model4(1,:,:);
v_model3(1,:,:) = v_model4(1,:,:);

% calculate attention values for each time step
for i = 2:i_steps % "2" because "1" is already initialized
 summodel = zeros(1,3);
 for x = 1:c_gridmaxX
  for y = 1:c_gridmaxY
   taskfactor = 0; % influence by contacts on grid (x,y)
   numberofcontactsonxy = 0;
   if v_increments(i,1,2) > 0 % if there are contacts at this timestep
    for k = 1:v_increments(i,1,2) % go through all contacts
     if v_target4(v_increments(i,1,3)+k-1,3)+1==x && v_target4(v_increments(i,1,3)+k-1,4)+1==y
      % calculate the task factor with the previously specified weights per participant
      % using a_parameters(for each participant, parameternumber)
       average = (a_parameters(a_participantnumber, 1)* ...
        v_target5a(v_increments(i,1,3)+k-1,2) + ...
        a_parameters(a_participantnumber, 2) * ...
        v_target5a(v_increments(i,1,3)+k-1, 3) + ...
        a_parameters(a_participantnumber, 3) * ...
        v_target5a(v_increments(i,1,3)+k-1, 4)) /...
        sum(a_parameters(a_participantnumber, 1:3));
      end
      taskfactor = max(taskfactor, average); % use only maximum task factor
      numberofcontactsonxy = numberofcontactsonxy + 1;
     end
    end
   end
   if numberofcontactsonxy == 0 % is no contacts than use default values
    taskfactor = a_parameters(a_participantnumber, 6);
    numberofcontactsonxy = a_parameters(a_participantnumber, 7);
   end
   if v_gaze2(i,2) >= 0 % if there is a gaze
```

```
    gazefactor = 1/(1 + gazeweight * sqrt( ( xstep*( x - v_gaze2(i,2) ) ).^2 + ...
    ( ystep*( y - v_gaze2(i,3) ) ).^2 ) / sqrt(c_primarymaxX.^2 + c_primarymaxY.^2));
   else
    gazefactor = 1/sqrt(c_primarymaxX.^2 + c_primarymaxY.^2);
   end
   v_model1(i,x,y) = taskfactor*gazefactor;
   summodel = summodel + v_model1(i,x,y);
  end
 end
 % normalize model (1)
 v_model2(i,:,:) = v_model1(i:,:)/summodel;
end

% merge old with new
for i = 2:i_steps
 summodel = zeros(1,3);

 % calculate real attention values
 for x = 1:c_gridmaxX
  for y = 1:c_gridmaxY
   DECAY = decayfactor.^(c_timeinterval/1000);
   OLD = v_model4(i-1,x,y);
   NEW = v_model2(i,x,y);
   v_model3(i,x,y) = OLD*DECAY + NEW*(1-DECAY);
   summodel = summodel + v_model3(i,x,y);
  end
 end

 % normalize (2)
 v_model4(i,:,:) = v_model3(i,:,:)/summodel;
end
```