Proceedings of the First
Artificial Intelligence and Interactive
Digital Entertainment Conference
(AIIDE-2005)

Particle-Based Communication Among Game Agents

Mike Klaas and **Tristram Southey** and **Warren Cheung**
Computer Science Dept.
University of British Columbia
{klaas,tristram,wcheung}@cs.ubc.ca

Abstract

One approach to creating realistic game AI is to create autonomous agents that can perform effectively with no more knowledge than a human player would have in their place. In a multi-agent setting, it is *also* necessary to devise a means for communicating among agents in collaborative game scenarios (such as a group of controlled agents that are searching for the player), since agents no longer have access to global knowledge. We present a method for communication using particle filters in the setting of game state estimation.

Particle filters are an efficient, nonparametric means of performing inference in complex environments. Their use in game AI is particularly compelling, as they provide an easy way to represent nonlinear, non-Gaussian inferences about the state space, while exhibiting computational thrift. We demonstrate that communication among a group of agents—using particle filters to reason about the state space—can be accomplished in a natural way by sharing particles among the agents' filters. We also show how a criterion for deciding when to communicate naturally falls out of this framework.

We apply this model in the setting of coordinated target detection, and find that agents of heterogenous types and complexities can nevertheless coordinate effectively.

Introduction

Many modern games strive to provide the illusion of a group of realistic and challenging opponents that only utilise in-game information and do not "cheat." This can be achieved by simulating autonomous agents that communicate and coordinate in a believable manner. We present a solution to the problem of coordinated state estimation in a multi-agent game setting through the use of particle sharing between multiple particle filters.

As an example, consider a group of pursuers—opposing guards attempting to locate a friendly espionage unit. Each guard searches independently, rather than sharing a "group mind," and reasons about the game state on their own. However, a guard can also "radio" specific data to the other guards to send and receive updates on the status of their search for the player. In this way, the guards avoid searching areas that has already been covered yand can determine the regions where the player is likely to be hiding.

Collaborative game state estimation is related to the general problem of multi-agent coordination. It poses many of the same challenging subproblems: determining *what* information is important to send to other agents, *when* to send this information, and *how* to integrate the data other agents send you into your beliefs about the state.

The application of particle filters to produce realistic state estimation in games was recently shown in (Bererton 2004). He argued the importance of isolating the knowledge available to game agents from the complete knowledge available to the game system. Particle filtering provids an efficient and flexible means of modelling the knowledge of a game agent searching for a target. We demonstrate that particle filtering as an inference algorithm naturally provides other compelling advantages for use in game AI, namely an automatic means of determining the importance of an observation (by estimating its likelihood), and a direct way of sharing game state inferences among agents by sharing particles.

This problem is highly related to the well-developed field of distributed inference and sensor fusion. However, many systems previously proposed have depended on a central element to coordinate inference, a requirement we do not accept for reasons of scalability and realism. Fully distributed systems have been proposed, however (Rosencrantz, Gordon, & Thrun 2003). In this protocol, an agent will request an update from another agent by sending them a subsampled version of its probability distribution. The receiving agent then compares the distribution against all of his observations, and replies with the observation most divergent to the received distribution. The authors do not, however, propose a protocol for when agents should query other agents. Also, agents must have identical observation models for this scheme to operate effectively, and be able to evaluate the probability of such observations, which may be impossible without local data. We propose that the inferences themselves be transmitted, rather than the observations. This saves computation, as the receiving agent does not have to re-evaluate observations, and allows agents' observation models to be arbitrary and unknown to the other agents.

Our particle-passing scheme is demonstrated using several examples, focusing on cooperative tracking. These examples include various observation models whose complexity leads to nonlinear multi-modal beliefs about the state space which can nonetheless be effectively shared.

Bayesian Filtering

Our goal is to develop game agents that have the ability to plan and act intelligently without access to priviledged knowledge. Instead, the agents have access to the same *observations* that a human player would have in their place. Thus, the agent is faced with the problem of estimating the *latent state* of the game given her observations. The latent state can be represented as a multi-dimensional vector comprised of anything relevant to the agent's planning; obvious examples include the location of the player's character(s) or the details of the map.

More formally, let $\mathbf{x}_t$ be the latent state of the game at time t, and $\mathbf{z}_t$ the observation vector. The task is to estimate the probability distribution over the latent state, i.e., the *marginal filtering distribution* $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, where $\mathbf{z}_{1:t} \triangleq \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$. In the sequential setting, a Markov assumption is typically made, so that the dynamics are completely specified by the transition prior $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and the observation model $p(\mathbf{z}_t|\mathbf{x}_t)$.

Bayesian filtering boils down to a two-step recursion. First, a $prediction$ is made of the current state given past knowledge, given by marginalizing over $\mathbf{x}_{t-1}$:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1}. \quad (1)$$

Equation (1) is called the *predictive density*. Next, we incorporate the observations at time t to produce the filtering distribution:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \propto p(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{1:t-1})\,. \quad (2)$$

Unfortunately, it is usually impossible to evaluate equations (1) and (2) analytically. This is particularly true in game settings, where complex observation models are common and the posterior distribution is often multi-modal. Fortunately, *particle filters* provide an efficient, general means of performing state estimation in these cases (Metropolis & Ulam 1949).

Particle filters estimate a probability distribution $p(\mathbf{x})$ with a set of samples $\left\{\mathbf{x}^{(i)}\right\}$ called *particles* with associated weights $\left\{w^{(i)}\right\}$. The approximation of the density is then:

$$p(d\mathbf{x}) = \sum_{i=1}^{N} w^{(i)} \delta_{\mathbf{x}^{(i)}}(d\mathbf{x})$$

where $\delta_{\mathbf{x}^{(i)}}(d\mathbf{x})$ is the delta Dirac function.[1] The sequential particle filtering algorithm at time t proceeds as follows: First, we sample N particles $\left\{\mathbf{x}_t^{(i)}\right\}$ from the predictive density. Next, we calculate the weight of each particle using the update equation. Renormalization of the weights produces a particle representation of the posterior. An optional resampling step may also be performed, to prevent degeneracy (Fearnhead 1998). Further details can be found in (Robert & Casella 1999; Doucet, de Freitas, & Gordon 2001).

[1] Defined as $\delta_{\mathbf{x}^{(i)}}(d\mathbf{x}) \triangleq \begin{cases} 1 & \mathbf{x}^{(i)} \in d\mathbf{x}, \\ 0 & \text{otherwise.} \end{cases}$

Distributed filtering

There is a wide body of distributed tracking literature. Many approaches focus on a centre that collects distributed information, but recent work has focused on truly distributed tracking, where each node is responsible for some of the inference work. This is more appropriate for the settings we consider, where we have multiple agents making independent decisions. Previous work has also concentrated on the symmetric case, where all agents involved share the same observation and transition models, which facilitates distributed inference considerably. For instance, this enables observations to be shared directly. We consider an assymetric case, where the group of agents are all trying to estimate the same quantity (such as the position of the evader), but have access to different observations, and have different observation models.

If every agent had a completely distinct model, then there would be no way to relate their inferences. Our point of departure is to assume that the predictive density (eq. (1)) is a mixture of the predictions from the various agents:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \sum_{m}^{n_a} \pi_m \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p_m(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\, d\mathbf{x}_{t-1}$$

Here $\{\pi_m\}$ are fixed mixture coefficients that determine the weight each agent has toward the final prediction, and $p_m(\cdot|\cdot)$ is the posterior distribution for agent m. We do not assume that $p_m(\cdot|\cdot)$ obeys any parametric model—we need only that it is approximable by a particle distribution (which admits virtually all distributions of interest).

After the prediction step, agent a computes her posterior distribution $p_a(\mathbf{x}_t|\mathbf{z}_{1:t})$ as before, using the update equation (2):

$$p_a(\mathbf{x}_t|\mathbf{z}_{1:t}) \propto p_a(\mathbf{z}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$$

We are inspired by the work of Vermaak *et al.* (2003) who use a mixture to model the *posterior* distribution to perform multi-target tracking efficiently. The technique cannot be directly applied to our situation where each agent has distinct observation models, leading to our use of a mixture-based *prediction* model.

Particle filter algorithm

In a particle filter approximation, we assume that each agent a has a set of particles $\left\{\mathbf{x}_{t-1,a}^{(i)}\right\}_{i=1:N}$ with associated weights $\left\{w_{t-1,a}^{(i)}\right\}_{i=1:N}$ that constitute a Dirac approximation of the agent's posterior in the previous time step, *i.e.*

$$p_a(d\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) = \sum_{i=1}^{N} w_{t-1,a}^{(i)} \delta_{\mathbf{x}_{t-1,a}^{(i)}}(d\mathbf{x}_{t-1})$$

which produces the following approximation to the predictive density:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \sum_{m}^{n_a} \pi_m \sum_{i}^{N} w_{t-1,m}^{(i)}\, p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1,m}^{(i)}\right). \quad (3)$$

Thus, if we use the prediction model as our proposal distribution, we obtain the following algorithm for agent a's particle filter:

1. sample $\mathbf{x}_{t,a}^{1:N} \sim \sum_m^{n_a} \pi_m \sum_{i=1}^{N} w_{t-1,m}^{(i)}\, p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1,m}^{(i)}\right)$
2. calculate weights $\widetilde{w}_{t,a}^{(i)} = p_a\left(\mathbf{z}_t \middle| \mathbf{x}_{t,a}^{(i)}\right)$
3. normalize $w_{t,a}^{(i)} = \widetilde{w}_{t,a}^{(i)} / \sum_j \widetilde{w}_{t,a}^{(j)}$.

Note that no resampling step is required as the predictive density is already a mixture. If there is only one agent, then the predictive density (3) simplifies to

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \sum_{i}^{N} w_{t-1}^{(i)}\, p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{(i)}\right).$$

Sampling from this mixture is equivalent to resampling the particles then sampling from the transition prior.

Sporadic communication

In the previous section, we assumed that the particles from all agents were available at each time step from which to draw the predictive particles. Of more interest is the case of sporadic communication. At time steps in which no message is being sent by agent a_m, we distribute her predictive responsibility over the remaining agents in proportion to their original weights. Let C_t be the set of agents that are sending a message at time t. The the mixture weights for the predictive density at time t are:

$$\widetilde{\pi}_{t,m} = \frac{\pi_m \mathbb{I}_{a_m \in C_t}}{\sum_i^{n_a} \pi_i \mathbb{I}_{a_i \in C_t}}.$$

Limiting the size of messages passed among agents is desirable both from a resource (maintaining the computation advantage of particle filters) and game perspective (it may be unrealistic in the game setting that an agent can communicate verbosely with other agents). The size of the message can be varied by limiting the number of particles to send. If this number is fewer than the size of the filter, the agent will subsample her own particles. Note that the agent's mixture component in this case is *unweighted*—if R particles are sent, each particle will have weight $1/R$.

Deciding when to communicate

One of the main problems an agent faces is deciding *when* to communicate her inferences to the other agents. We show that particle filters naturally provide an easy way of making this decision. We want to minimize communication of data that has been previously communicated while ensuring the pertinent information is conveyed immediately. To do this, we assume that anomalous data (data that poorly fits the current model of the world) is significant, and should be conveyed to the other agents. We accomplish this by evaluating the likelihood of the data at time t: $p(\mathbf{z}_t|\mathbf{z}_{1:t-1})$. When the likelihood falls too low, the agent updates its model to account for the current observations, and sends it to the other agents.

Let t' be the time of last communication. We wish to compute the likelihood of all observations obtained since t', namely

$$p(\mathbf{z}_{t'+1:t}|\mathbf{z}_{1:t'}) = \prod_{i=t'+1}^{t} p(\mathbf{z}_i|\mathbf{z}_{1:i-1}).$$

When this value drops below a threshold τ, communication is triggered. The value of τ depends on the probability model and on how often agent communication is desirable. This parameter can thus be used as a difficulty gauge by tuning the performance of the group of agents.

One of the many advantages of a particle filter is that $p(\mathbf{z}_t|\mathbf{z}_{1:t-1})$ is trivial to compute: it is simply the sum of the unnormalized weights $\widetilde{w}_t^{(i)}$. This gives us the following procedure for agent a at time t:

1. Save the unnormalize weights from step (2) in the particle filtering algorithm: $W_{t,a}^{\Sigma} = \sum_i \widetilde{w}_{t,a}^{(i)}$
2. Update likelihood for time t: $L_{t,a} = L_{t,a} \cdot W_{t,a}^{\Sigma}$
3. If $L_{t,a} < \tau$, trigger communication, and reset $L_{t,a} = 1$.

Discussion

The model we have adopted is known as a *mixture of experts*. In this scheme, each expert has a certain weight (π_m in the previous section), and the final belief is the weighted sum of all the experts. As we have presented the algorithm, there is a single set of mixture weights $\{\pi_m\}$, but there is no reason to limit all agents to use the same weights. In our examples, for instance, each agent puts more weight on her own beliefs than those of her fellow agents. If this system was used in a game in which agents did not have complete confidence in each other, the expert weights could be used as a way to tune the level of "trust" an agent has for other agents.

The choice of the mixture of experts model has a negative consequence, however: When agents are spatially separated, it is likely that the quality of beliefs of various agents will vary in different areas of the state space (agent a may have searched a room that agent b hasn't yet visited). By adopting a global mixture parameter, we limit a's ability to covey *negative knowledge* since b still believes that it is possible that the target is in that room (negative knowledge pertains to areas in which the state probability density is low). A possible solution to this problem is to segment the state space, and assign confidence weights separately in the various regions. In our small examples, however, we have found that negative knowledge can still be conveyed despite the use of global, fixed mixture weights.

A consequence of our communication criterion is that an observation may misfit the model due to noise rather than being an unusual feature. In extremely noisy environments, this scheme may cause the communication threshold to be crossed too often.

We also note that the communication particle filter preserves the linear cost in the number of particles, since each agent always samples the same number of particles from the mixture regardless of the size of the mixture. Particle filters can operate with more or fewer particles from turn to turn, depending on the computational resources available at the time.

Example 1: Jungle Search

In Figure 1, three agents (coloured circles) are searching for the target (black $\times$) within a "jungle" without success. The "jungle" contains regions of total and partial occlusion,

which reduce the probability of detecting the target. In this example, the three agents determine that the target is not within the high-visibility regions near them, and pass this information to each other.

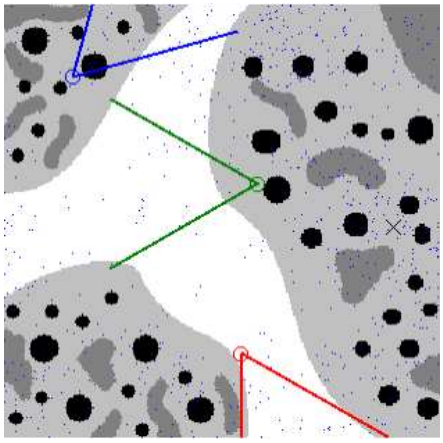

Figure 1: Agent B state estimate; Dirac distribution of 1000 particles. Example of information from two agents allowing a third agent to determine that the target is not only not in her own viewing cone but also not in the viewing cone of the others. The clear (white) regions have few particles, as messages from the green and red agents have low probability in those regions. In the obscured (grey) regions, none of the agents can be sure that the target is not present, so the particle density is higher.

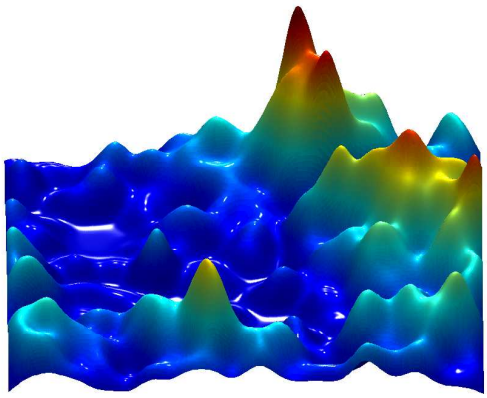

Figure 2: Smoothed probability density corresponding to the particle approximation in Figure 1.

Our probabilistic observation model allows for both partial and total occlusion. Each agent has a limited angle of view but is assumed to be able to see an unlimited distance in the absence of occlusion. Each pixel on the map has an associated occlusion value γ which is the probability of the line of sight of an agent being occluded if it passes through this pixel. The total occlusion value for a target P from an agent O is therefore equal to 1 if the target is outside the agent viewing angle, and equal to the product of all the occlusion values of the pixels on the line from the agent to the target, if the target is within the agent viewing angle. That is,

$$occ(O, P) = \prod_{i=0}^{p} (1 - \gamma_i)$$

where $\gamma_0..\gamma_p$ are the visibility probability for the pixels along a line drawn between the agent O and the target P. An example of this observation model for an agent looking through jungle foliage can be seen in Figure 3.

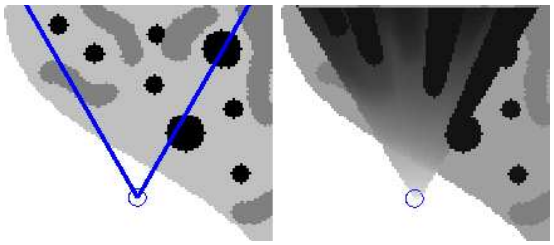

Figure 3: Observation model for an agent in the jungle. *left*: the original obscurity map. *right:* the darkness of each pixel within the agent's fi eld of view is proportional to the occlusion value of the pixel—the probability that an object at this location will *not* seen. Objects that are opaque (the black trees) "cast shadows" of regions of zero visibility.

We have three coloured agents: R, B and G, all using the observation model described above.

As agent B communicates with the other agents R and G, they are passing an approximation of their entire belief state, simultaneously transferring information about where the target is not located in addition to where the target may be hidden.

This example also shows that by using particle fi lters, we can easily accommodate observation models that include uncertainty in their observations. We can model a variety of environments, from obscurity from shadows and nighttime darkness to occlusion-fi lled terrains such as jungle and fog. The posterior belief of the state space for agent B is shown in Figure 2. This complex, multi-modal distribution would be impossible to handle using analytic Bayesian Filtering techniques (such as the Kalman Filter (Kalman 1960)).

Example 2: Range and Direction Sensors

This is an example of data integrated from two different observation models. We have two agents, R and D, attempting to detect a target T. R is able to noisily detect the distance to the target when making a successful observation. The observation model for agent D is only able to detect the direction in which the target is located when making a successful observation. Independently, the agents are cannot pinpoint the location of the target.

The agents both start off with evenly-distributed particles which represents a uniform belief in the target's location. Once they both successfully observe the target, they will initially infer the game state shown in the left of Figure 4.

Here, R has an obsevation of the range for the T but no direction, resulting in a ring of particles at the distance of the target. D has an observation of the direction of T, resulting in a cone of particles in the direction of the target. Neither sensor alone is able to accurately pinpoint the location of the target. However, as the two of them communicate, they update their models and are eventually fairly certain of the position of the target as seen in the right of Figure 4.

This example highlights the flexibility of the observation models that can be used. Agents with radically differing observation models are still able to communicate and coordinate with one another. Different agents could include human guards, cameras that detect a fixed field of view in a fixed position, laser tripwires that only detect the target passing through its line, touch pads that detect the target touching a region, motion sensors that can detect movement in a room, guard dogs that can smell a trail or heartbeat sensors that work through walls.

Example 3: A Camera and a Tripwire

In our third example, we look at a scene with a fixed security camera C and a laser scanner in a doorway L. C can reorient itself but has difficulty seeing long distances. The laser has a different observation model. It can only detect in the target passes directly over its length but it is highly accurate at doing so. At the outset (Figure 5), neither agent has any evidence about the targets location. The laser detects the target and communicates to the camera, telling it where to focus. The camera then turns to track the target.

Applications

We have focused on games where searchers spread throughout a map are passively guarding regions or actively searching for the player. The player wishes to accomplish his set of goals without having his exact position discovered by the searchers. Our model allows for realistic interaction between the player and the searchers. As the searchers do not communicate continuously, the player can disable a searcher that has discovered significant evidence of his position, preventing the searcher from communicating. The other searchers retain their own inferences and can continue to collaborate, while the state held by the disabled searcher is unavailable. Should the disabled searcher recover, he can rejoin the searchers and pass his inferences to the other searchers. The player could also feed false information to the searchers (by causing a noise, for instance), who would act on and communicate this misinformation. These induced errors in the model are eventually corrected by the searchers investigating and observing the true state, but would provide a temporary distraction.

In many real-time strategy games, a player must search a map for the location of one or more resources such as lumber or metal. By using our methods to direct search for these resources, a computer player would not have any undue advantage in searching resources over a human player. This would also allow artificial opponents to be able to find resources on an unknown or randomised map, and be unaffected by randomised starting positions. It would also easily handle resources that move, such as migrating buffalo, or perhaps automate this resource collection for a human player.

The player could also be in control of an army and be supported by game agents who would serve as leaders of his units. Limited communication allows the player to only update their model when we simulate communication from the player's leaders, which may be frequent in the case of infantry squad leaders in radio contact, or very rarely in the case of captains of submarines that operate mostly underwater or satellites which are often out of position.

Conclusion

Representing the inferred game state of multiple agents can be readily accomplished through the use of particle filters (Rosencrantz, Gordon, & Thrun 2003; Bererton 2004). Using a mixture model representation of the predictive density, agents can perform collaborative estimation of game state. Each agent can have an arbitrary level of confidence in the predictions made by other agents. Particle filters also provide a natural way of determining when communication should occur, by examining the likelihood of the observations, which is given straightforwardly in a particle filter by the sum of the unnormalized importance weights. Our model allows the agents to act autonomously, with heterogenous observation and prediction models, while also providing a framework for sharing their inferences.

This system was implemented in the context of coordinated target detection. We introduce an enhanced observation model which models partial occlusion using a cumulative per pixel occlusion. In this setting, our model successfully replicates intelligent cooperative behaviour of multiple agents that communicate in a realistic manner.

Acknowledgements

We thank Nando de Freitas and Jim Little for their invaluable advice.

References

Bererton, C. 2004. State estimation for game ai using particle filters. In *AAAI workshop on challenges in game AI*.

Doucet, A.; de Freitas, N.; and Gordon, N. J., eds. 2001. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Fearnhead, P. 1998. *Sequential Monte Carlo Methods in Filter Theory*. Ph.D. Dissertation, Department of Statistics, Oxford University, England.

Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Eng.* 82:35–45.

Metropolis, N., and Ulam, S. 1949. The Monte Carlo method. *JASA* 44(247):335–341.

Robert, C. P., and Casella, G. 1999. *Monte Carlo Statistical Methods*. New York: Springer-Verlag.

Rosencrantz, M.; Gordon, G.; and Thrun, S. 2003. Decentralized sensor fusion with distributed particle filters. In *UAI*.

Vermaak, J.; Doucet, A.; and Peŕez, P. 2003. Maintaining multi-modality through mixture tracking. In *ICCV*.

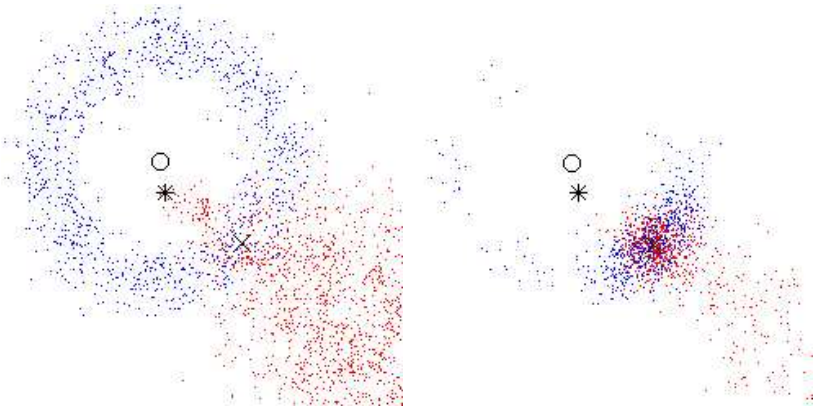

Figure 4: Example of two agents attempting to locate a target. The circle agent (blue particles) is only able to observe the range to the target and the asterisk agent (red particles) is only able to determine in which direction to the target. Apart, neither is able to precisely determine the targets location. The left fi gure represents the the particle fi lters of the respective agents after they detect the target. The right fi gure shows the particle fi lters of the agents after they share their inferences, successfully locating the target.

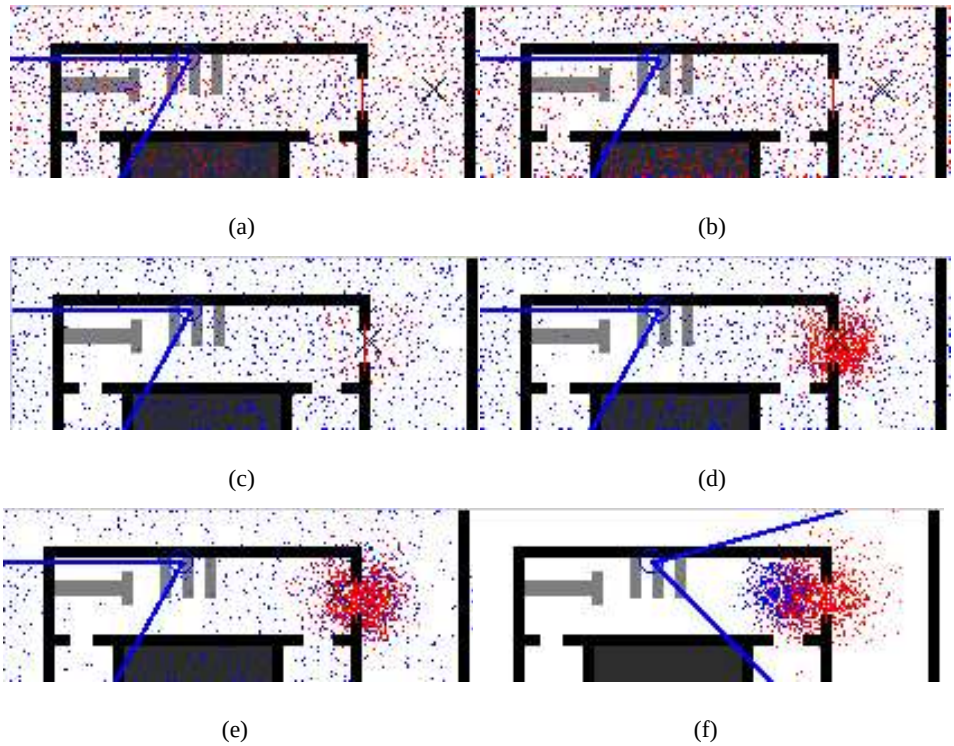

Figure 5: Example of two devices trying to locate the target. C is rotating security camera with a visual fi eld defi ned by the blue arc. S is a laser door sensor which can only detect the target if it passes over the red line. (a) C and S begin with an even distribution of particles. (b) C has eliminated all particles in its viewing angle. (c) S has sensed the target moving over it and recentered its particles on the target. (d) S creates new particles from the positive detection. (e) C receives information from S and adjusts its own particles. Each now has an accurate estimate of the target's position. f) C turns to orient on the target and detects it. S's estimate is now incorrect since it does not observe the target, but C can now remit the new position of the target.