4

Multi-UAV Cooperative Perception Techniques

Luis Merino[1], Fernando Caballero[2], Joaquín Ferruz[2], Johan Wiklund[3], Per-Erik Forssén[3], and Aníbal Ollero[2]

[1] Pablo de Olavide University, Crta. Utrera km. 1, 41013 Seville (Spain)
lmercab@upo.es
[2] Robotics, Vision and Control Group, University of Seville, Camino de los Descubrimientos s/n, 41092 Seville (Spain)
{caba,ferruz,aollero}@cartuja.us.es
[3] Linköping University, SE-581 83, Linköping (Sweden)
{jowi,perfo}@isy.liu.se

Summary. This Chapter is devoted to the cooperation of multiple UAVs for environment perception. First, probabilistic methods for multi-UAV cooperative perception are analyzed. Then, the problem of multi-UAV detection, localization and tracking is described, and local image processing techniques are presented. Then, the Chapter shows two approaches based on the Information Filter and on evidence grid representations.

4.1 Introduction

Applications such as natural and human made disasters scenarios, search and rescue, law enforcement, aerial mapping, traffic surveillance, inspection or cinematography require robust and flexible perception systems [31]. These perception systems use the sensors on-board the UAVs to perceive the environment, estimate the situation of an event and/or the own state of the robots.

In the case of the team of robots, one can do better than a robot perceiving alone. The information that each robot obtains about the environment can be *shared* to improve the perception, so that each robot obtains a better picture of the world than if it would be alone. Moreover, from the perception point of view, the robots can explicitly *cooperate*, developing actions to collect data. Thus, *cooperative robot perception* could be defined as the collaboration inside a fleet of robots for the estimation of the state of the environment, by sharing information or even by developing cooperative actions.

In this chapter we consider multiple heterogeneous UAVs in cooperative perception activities (Fig. 4.1 illustrates a possible scenario). The heterogeneity increases the complexity of the problem, but also provides several advantages for the application such us the possibility to exploit the complementarities of different UAV platforms with different mobility attributes and also different sensor and perception functionalities. It should be noted that many applications require several sensors that cannot be carried by only one UAV due to payload limitations.

A. Ollero and I. Maza (Eds.): Mult. Hetero. Unmanned Aerial Vehi., STAR 37, pp. 67–110, 2007.
springerlink.com

Fig. 4.1. Different moments during a fire detection mission using three UAVs

In these cases the cooperation between the UAVs, equipped with different sensors, should be established also at a perception level.

4.1.1 Main Issues

When considering multi-robot perception, there are several important issues that should be addressed.

Knowledge representation and information fusion. Robots will use their sensors to update their knowledge. This knowledge is usually arranged into a hierarchy from lower to higher abstraction levels, ranging, for instance, from the raw sensor readings, estimations of the own robot position and velocity or the position of the surrounding obstacles to the estimation of the shape or appearance aspects of an object or the identity of a particular object.

If the robots can communicate, then the information received from other robots can be fused with the local one to improve this knowledge. Information fusion is, then, a key issue in a cooperative perception system.

Information fusion requires to translate the received information to the same local representation. Therefore, the general fusion architecture will affect the local representation employed by each robot. Also, related to that is the problem of data association, that should be solved in order to determine if two robots are referring to the same part of the world.

Fusion rules should lead to an improved knowledge about the world, but care should be taken to avoid *rumor propagation* in the case of decentralized systems. This can occur within a fleet of robots, and can lead to overoptimistic estimations.

Cooperation. Robots use locally the perception estimates to react under a changing world and even to develop plans if they are endowed with decisional abilities. These actions can include information gathering tasks that improve the local knowledge of the world, like visiting unknown zones or moving to better points of view. Moreover, the robots can coordinate themselves and even cooperate in these tasks. In this case, it should be considered *metrics* about the gain of information that the actions of a particular robot of the fleet produce from the point of view of perception.

4.1.2 Approach and Outline

The framework that will be used for knowledge representation is probability theory. It is accepted that any system capable of uncertain reasoning is generally more robust that other one that does not [46]. So, the perception algorithms should be able to handle uncertain data, and to maintain information about this uncertainty. In the last decade, probabilistic algorithms have become one of the most prominent tools for uncertain reasoning among roboticists. In probabilistic perception, the current knowledge is represented by a probability distribution that assigns to each possible value of the state space a probability, the *belief state*.

Regarding the multi-robot perception architecture, the first decision is whether the information gathered by the fleet is combined in a purely centralized fashion (all the data collected in a central node that builds a world representation) or in a decentralized manner, in which each robot compounds its own model and communication occurs at a higher level. Also, the architecture considered could be something in between these two extremes. Of course, the solution depends on several issues, like the physical communication layer available, the local processing power of the robots, the tasks to be accomplished, the structure of the environment, the autonomy with which the individual robots should be endowed, etc.

In general, centralized solutions can provide optimal solutions, but they do not scale well with the number of robots. Therefore, the proposed multi-robot perception architecture is decentralized, in which each robot builds its own local (and possibly partial) representation of the world, which is described by the belief state. Then, the robots will share their beliefs, and include the beliefs received from other robots in order to improve their knowledge. No central node is required, although the information is transmitted to a central station for planning and replanning activities, operator visualization, etc.

The rest of the Chapter is organized as follows. The probabilistic framework for decentralized perception is summarized. This framework will be applied for cooperative detection, localization and tracking. The main sensors considered are video cameras. Then, the local processing techniques employed by the UAVs are described. A first approach, based on the information filter, is presented. Finally, a grid-based approach for fusion of heterogeneous sensors is described.

The perception system (see [25] for a more detailed description of the software system), together with the decisional architecture presented in Chap. 2, allows for cooperative detection, confirmation and localization of events of interest. Chapter 8 will present the application to forest fire detection. This is a very relevant application in many countries where forest fires have disastrous social, economic and environmental impact.

4.1.3 Related Work

An important part of the work on multi-robot systems has been focused on the development of architectures for robot coordination. Several architectures have been summarized in Chap. 2. In general, these architectures focus on task planning, task allocation, coordination, planning and control, conflict resolution,

etc. Although a group of robots can coordinate with no or little communication [24, 4], all in all the previous architectures require communication of actions, plans, tasks, which can be interpreted as a kind of information sharing. However, the knowledge sharing problem and its implications on robot coordination are not explicitly considered in most of those approaches.

Of course, there have been applications of robot teams for distributed and/or cooperative perception activities. For instance, in the task of map making [39, 38, 15, 5]. More recently, approaches including cooperative multi-robot Concurrent Mapping and Localization (CML) have been presented [11, 47].

Other applications include multi-robot surveillance. For instance, the objective of the CyberScout project [35] is the development of a network of robots for reconnaissance and surveillance operations. Multi-robot target tracking is considered in several multi-robot systems, like [33, 44]. More recently, Howard [19] has described results within the DARPA Software for Distributed Robotics initiative with an heterogeneous team of near 80 robots in indoor experiments. Mapping activities, SLAM and cooperative detection of intruders are applications considered. In the context of Robocup several cooperative perception results have also been presented, as for instance [43, 52, 36].

Many of the applications are ad-hoc algorithms for combining data from several sources. The different approaches differ in the way they represent the information, how the data is communicated and fused, and the network topology (centralized or distributed). References [28, 33] deal with issues of decentralized information fusion employing probabilistic techniques.

Most previous approaches are applied in structured environments. Regarding unstructured environments, less work can be identified. Moreover, the application of cooperative perception for teams of UAVs are more rare than in teams of ground robots.

In the BEAR project [50], pursuit-evasion games involving UAVs and Unmanned Ground Vehicles (UGVs) are considered. A probabilistic framework is employed for obstacle and evaders position estimation by the fleet of pursuers (UAVs), and to determine pursuit polices [51]. In this case, the data fusion and policy determination are carried out by a central node.

Closest to part of the work presented here is the work described in [45] developed in the framework of the ANSER project. In that project, an architecture for multi-vehicle data fusion is designed, and its application to multi-UAV SLAM using vision is presented. State estimation is performed distributelly using the information form of the Kalman filter. Each vehicle uses the information received to update its state and its local map. The map information (represented as the location of a discrete set of landmarks) estimated for each vehicle is propagated to the rest of the fleet. In this work, artificial landmarks of known size are used in order to obtain range and bearing measurements. Nevertheless, the main issues regarding decentralized data fusion are considered.

In the context of the same group, Reference [17] presents techniques for coordination of sensor platforms in order to maximize the information gain, which is important for cooperative perception activities. It describes results derived from

the information form of the Kalman filter or Information Filter. A study on the use of particle filters on the same context have been presented recently [32].

In [53], the authors present a multi-UAV map-building approach based on evidential reasoning. The objective is that a group of UAVs build a certainty grid about potential targets on an area. The paper shows results only in simulation, and presents an interesting comparison with a Bayesian approach to the same problem. The authors conclude that the Dempster-Shafer approach can yield better results in terms of timing when the sensors' accuracy is low. Also, the paper considers cooperative path planning methods based on the results from both, the Bayesian and evidential approaches. Nevertheless, the algorithm presented is purely centralized, and nothing is said about a decentralized version for the evidential approach.

In [7] the feasibility of the application of a team of small (low-altitude short endurance) UAVs to cooperatively monitor and track the propagation of large forest fires is explored. The paper provides simulations using a six degree of freedom dynamic model for the UAV and a numerical propagation model for the forest fire.

4.2 Probabilistic Algorithms for Cooperative Perception

As said before, the robots will employ probability theory to represent their current knowledge. The environment and the robots are characterized by what is called the *state*, which is represented at a given instant t as the vector $\mathbf{x}_t$. The objective of a *cooperative* perception system is to obtain an estimation of this state from the sensorial data on-board the different robots. $\mathbf{z}_t$ represents the set of measurements obtained at time t. In a probabilistic framework, the current knowledge about the state is given by what is called the *belief* on the state, defined as:

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}^t). \tag{4.1}$$

that is, the probability distribution on the state conditioned on all the information gathered up to time t, $\mathbf{z}^t$. From the probabilistic point of view, this conditional distribution, called the *posterior*, represents all the information the robot can compute from the sensor data collected.

The main tool for probabilistic state estimation is Bayesian inference [42, 46]. Bayesian inference allows to integrate measurements generated by the perception tools with prior knowledge about the state to obtain an updated estimation of the belief on the state. Under some assumptions, the Bayes filter allows for a recursive estimation of the state of the events. The equation for the Bayes recursion [42] is:

$$p(\mathbf{x}_t|\mathbf{z}^t) = \overbrace{\eta^{-1} p(\mathbf{z}_t|\mathbf{x}_t)}^{update} \underbrace{\int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}^{t-1}) \mathrm{d}\mathbf{x}_{t-1}}_{prediction} \tag{4.2}$$

$$\eta = p(\mathbf{z}_t|\mathbf{z}^{t-1}) = \int_{\mathbf{x}_t} p(\mathbf{z}_t|\mathbf{x}_t) \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}^{t-1}) \mathrm{d}\mathbf{x}_{t-1} \mathrm{d}\mathbf{x}_t \tag{4.3}$$

Equation (4.2) generally is decomposed in two steps, called *prediction* and *updating*. Prediction usually implies an increase in the amount of uncertainty on $\mathbf{x}$, while updating narrows this uncertainty as a consequence of the new measurement.

The important terms in the previous expressions are the conditional distributions $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ (the *transition* model) and $p(\mathbf{z}_t|\mathbf{x}_t)$ (the *measurement* model) and the conditional dependencies among the random variables.

Using the same assumptions, it is possible to derive the Bayes filter for the recursive estimation of the belief for the full state trajectory $p(\mathbf{x}^t|\mathbf{z}^t)$ [42]:

$$p(\mathbf{x}^t|\mathbf{z}^t) = \eta p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}^{t-1}|\mathbf{z}^{t-1}) \tag{4.4}$$

Another convenient expression for this equation, that unrolls the previous expression until considering the *prior* information $p(\mathbf{x}_0)$, is:

$$p(\mathbf{x}^t|\mathbf{z}^t) = \eta^{'} p(\mathbf{x}_0) \prod_{\tau=1}^{\tau=t} p(\mathbf{z}_\tau|\mathbf{x}_\tau)p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}) \tag{4.5}$$

4.2.1 Multi-robot Perception

In the case of a multi-robot fleet (for instance, a fleet of UAVs), the objective is to cooperatively estimate the state of the world (that is, the relevant information, represented by $\mathbf{x}_t$) from the measurements obtained by all the robots of the fleet.

In order to determine what information should be communicated and how this information is fused with the local knowledge of each robot, first it is analyzed the resultant knowledge in the case that all information were available at any point of the fleet. That could be considered an ideal `omniscient` situation, in which a central node gets all the available information at any time.

The measurements $\mathbf{z}_t$ are the collection of all the measurements gathered by all the sensors of the fleet of robots $\{\mathbf{z}_{j,t}, j = 1, \ldots, M_t\}$. The current measurement is given by the vector $\mathbf{z}_t^m = [\mathbf{z}_{1,t}^T, \ldots, \mathbf{z}_{M_t,t}^T]^T$. Then, the belief state for the central node is given by:

$$bel_m(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}^{m,t}) = \eta p(\mathbf{z}_t^m|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}^{m,t-1})\mathrm{d}\mathbf{x}_{t-1} \tag{4.6}$$

Given the assumption that the data gathered by the different robots at any time instant t are *conditionally independent* given the state $\mathbf{x}_t$, the previous equation becomes:

$$bel_m(\mathbf{x}_t) = \eta \prod_{j=1}^{M_t} p(\mathbf{z}_{j,t}|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}^{m,t-1})\mathrm{d}\mathbf{x}_{t-1} \tag{4.7}$$

For data fusion purposes, it will be very important to determine the likelihood function $p(\mathbf{z}_{j,t}|\mathbf{x}_t)$ for every source of measurements within the robots of the fleet. Likewise, in the multi-robot case, (4.5) becomes:

$$bel_m(\mathbf{x}^t) = p(\mathbf{x}^t|\mathbf{z}^{m,t}) = \eta^{'} p(\mathbf{x}_0) \prod_{\tau=1}^{\tau=t} \Big[\prod_{j=1}^{M(\tau)} p(\mathbf{z}_{j,\tau}|\mathbf{x}_\tau) \Big] p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}) \tag{4.8}$$

The computation of (4.7) and (4.8) are the ideal objective for the cooperative perceptions algorithms. It should be noted that nothing has been commented yet about what the state variables are, the particular distributions considered and so forth.

4.2.2 Semi-decentralized Belief Computation

In general, it is not possible to dispose of all the data in a central node without delays due to bandwidth limitations, bounded communication ranges, etc. On the other hand, if the robots have decisional capabilities, then they should maintain their own local belief states that will be used by those decisional layers for planning activities and so forth. Therefore, the idea would be to combine in some way these local estimations, by communicating high-level belief states instead of raw data.

In a fully decentralized approach, the robots *share* their beliefs with their neighbors. Then, the received information is locally fused in order to improve the local perception of the world. The main question is if there is a way of combining the belief states so that the final belief state is closer (ideally the same) to the global belief that could be computed in the centralized case, represented by (4.7) and (4.8).

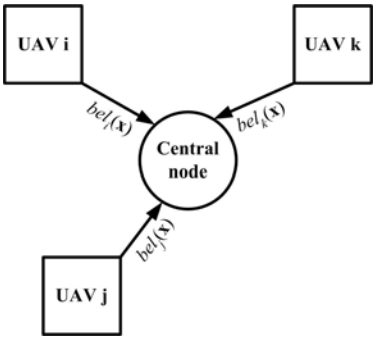

Fig. 4.2. Scheme that shows the situation in the semi-decentralized scheme

Before getting into the issues related to the fully decentralized computation, this section analyzes a first approach, depicted in Fig. 4.2. This approach will be named semi-decentralized. In it, each robot i maintains its local belief state $bel_i(\mathbf{x}_t)$:

$$bel_i(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_i^t) = \eta_i p(\mathbf{z}_{i,t}|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_i^{t-1}) \mathrm{d}\mathbf{x}_{t-1} \tag{4.9}$$

where $\eta_i^{-1} = p(\mathbf{z}_{i,t}|\mathbf{z}_i^{t-1})$. Then, the robot sends its belief state to a central node that combines all the local beliefs to obtain a global one.

The Case of Static State

If the state is static, then the belief state at time t using all robot data is given by:

$$bel_m(\mathbf{x}_t) = \eta_m p(\mathbf{x}) \prod_{\tau=1}^{\tau=t} p(\mathbf{z}_\tau^m|\mathbf{x}) \tag{4.10}$$

where the time index has been removed to indicate that the state is static, and then $bel(\mathbf{x}_t)$ means the belief state after all the data gathered up to time t has been integrated. Similarly, for any robot i:

$$bel_i(\mathbf{x}_t) = \eta_i p_i(\mathbf{x}) \prod_{\tau=1}^{\tau=t} p(\mathbf{z}_{i,\tau}|\mathbf{x}) \tag{4.11}$$

If M robots are considered, as $p(\mathbf{z}_t^m|\mathbf{x}) = \prod_{i=1}^{M} p(\mathbf{z}_{i,t}|\mathbf{x})$, then, if the prior beliefs $p(\mathbf{x})$ are the same:

$$bel_m(\mathbf{x}_t) = \eta p(\mathbf{x}) \prod_{i=1}^{M} \frac{bel_i(\mathbf{x}_t)}{p(\mathbf{x})} \tag{4.12}$$

This equation gives a basic formula to combine the robots beliefs in order to obtain the global one. It means that the central node directly combines all the beliefs received, after removing the common information that all robot share (the prior $p(\mathbf{x})$). Another convenient way of representing the previous relations are in recursive form.

$$bel_m(\mathbf{x}_t) = \eta_m p(\mathbf{z}_t^m|\mathbf{x}) bel_m(\mathbf{x}_{t-1}) \tag{4.13}$$

$$bel_i(\mathbf{x}_t) = \eta_i p(\mathbf{z}_{i,t}|\mathbf{x}) bel_i(\mathbf{x}_{t-1}) \tag{4.14}$$

so

$$bel_m(\mathbf{x}_t) = \eta bel_m(\mathbf{x}_{t-1}) \prod_{i=1}^{M} \frac{bel_i(\mathbf{x}_t)}{bel_i(\mathbf{x}_{t-1})} \tag{4.15}$$

Expressing the beliefs in logarithmic form:

$$bel_m(\mathbf{x}_t) = \log(\eta) + bel_m(\mathbf{x}_{t-1}) + \sum_{i=1}^{M} \left[bel_i(\mathbf{x}_t) - bel_i(\mathbf{x}_{t-1})\right] \tag{4.16}$$

Therefore, what the central node has to do is to sum (in logarithmic form) into a running total the increase in *evidence* given by every robot of the fleet. Even though no particular form or representation for the belief states is assumed yet, an interesting characteristic of this case is that, as the state is static, the message size for communicating the beliefs is fixed. Moreover, the number of messages sent by each robot does not depend on the number of robots [33]. Finally, one of the most important issues is that, in this case, the beliefs can be received *asynchronously* and with *arbitrary latency* (i.e., in the previous equations $t-1$

can be substituted by the previous instant in which the central node received information from each robot). Hence, in the case of a static state, each robot can accumulate evidence and transmit it at convenience, without additional memory costs.

Dynamic Environments

In order to reconstruct the ideal centralized belief state, as noted in [33], in the dynamic case the role that was played before by belief state at time t, $bel(\mathbf{x}_t)$, is played now by the belief state over the full state trajectory up to time t, $bel(\mathbf{x}^t)$.

The belief state over the full trajectory for robot i is:

$$bel_i(\mathbf{x}^t) = p(\mathbf{x}^t|\mathbf{z}_i^t) = \eta^{'} p(\mathbf{x}_0) \prod_{\tau=1}^{\tau=t} p(\mathbf{z}_{j,\tau}|\mathbf{x}_\tau) p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}) \tag{4.17}$$

Comparing this expression to (4.8) then:

$$bel_m(\mathbf{x}^t) = \eta p(\mathbf{x}_0^t) \prod_{i=1}^{M} \frac{bel_i(\mathbf{x}^t)}{p(\mathbf{x}_0^t)} \tag{4.18}$$

where $p(\mathbf{x}_0^t) = p(\mathbf{x}_0) \prod_{\tau=1}^{\tau=t} p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1})$. Therefore, if each robot sends its belief state over the state trajectory, the beliefs can be combined to obtain a global one that would be equal to the one obtained in a centralized version. Moreover, as in the static case, the belief states can be received asynchronously. Each robot can accumulate evidence, and send it whenever is possible to communicate with the central node. However, the problem in this case is that the state grows over time, and therefore the size of the message needed to communicate the corresponding beliefs. In general, this will make this scheme of communication unaffordable. Nevertheless, for the normal operation of the robots, only the state trajectory over a time interval is needed, so this belief trajectory can be bounded. In any case, depending on the representation of the belief states and the dimension of the state itself, the size of information needed to be transmitted can be prohibitively high.

4.2.3 Decentralized Cooperative Perception

In a fully decentralized scheme, there is no central node that fuses the beliefs obtained by the robots. In a decentralized fleet, no robot have knowledge of the global network topology, and there is no global communication facility, in the sense that cannot be ensured that the information transmitted by one robot will reach all the robots of the fleet [16]. At any time instant, each robot will be able to communicate and share its belief directly with a subset of the fleet.

There is a situation in which the previous general equations can be applied. If the belief network topology[1] is a tree as shown in Fig. 4.3, that is, if there

[1] The term network refers in this case should be interpreted at the data level. That is, the underlying physical or transportation layers may have a different topology.

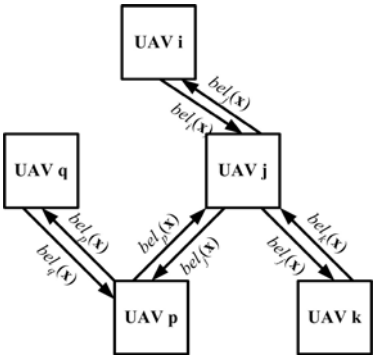

Fig. 4.3. A group of UAVs forming a tree-shaped network

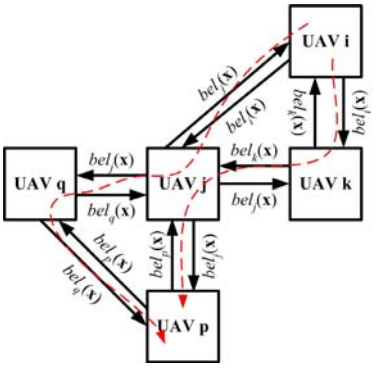

Fig. 4.4. Multiple paths can lead to overestimation, when incorporating several times the same information

is a unique path between any pair of providers and receivers, then the fusion equations (4.12) and (4.18) for the central node can be applied at every robot to fuse the beliefs received from its neighbors [49], and then update their own beliefs. The only aspect that must be considered is to avoid considering multiple times the same information through the direct link between two robots with direct communication (like UAV p and UAV q in Fig. 4.3). However, this can be solved locally, without further information about the rest of the network. Each robot only needs to know what information it transmitted to its neighbors in order to remove it when integrating received data.

However, the same scheme cannot be applied to general topologies without further considerations. Consider the scheme of connections of Fig. 4.4. In this case, there are multiple paths from the same UAV to others. The problem in this case is that the *same information* could be accounted *several times*. Thus, the belief combination rule has to consider the common information [16, 22, 45]. Otherwise, this could lead to overconfident estimates.

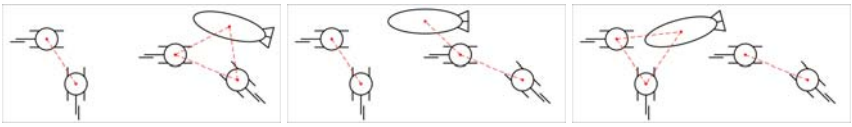

Fig. 4.5. In the general case, the topology of the network will change with time

Without additional assumptions on the form of the belief distributions, their representations and others, there is no general solution to the problem of considering the common information in networks of arbitrary topology [49], if only local information is considered. One solution would be to force a tree topology in the network. However, in a fleet of robots, the topology of the connections is dynamic and the tree structure of the network would have to be ensured along the time. Also, if a tree topology is forced, an interesting property is lost. One robot can act as a *data mule* between different subsets of robots of the fleet (see Fig. 4.5). When connected to one subset, the robot updates its belief from the information received from the subset. Then, when connected with the second subset, the robots of this second subset will incorporate information from the other subset indirectly through the belief state of the moving robot.

The previous sections have presented the general problem of information fusion for cooperative perception. The development of working algorithms, even in the case of centralized fusion, requires defining a particular representation of the belief states, which itself depends on the problem.

4.2.4 Developing Cooperative Perception Actions

The previous sections have showed the main issues in decentralized information fusion. The final important aspect of cooperative perception is the ability to develop actions to improve the knowledge about the environment. The objective is to determine which actions $\mathbf{u}^m$ the robots should carry out in order to increase some measurement about the goodness of the current information.

Determining robots actions considering uncertain worlds is usually tackled employing decision theoretic techniques. In the previous equations of the Bayes filter, the actions of the robots $\mathbf{u}^m$ can be considered explicitly.

$$bel_m(\mathbf{x}_t) = \eta \prod_{j=1}^{M_t} p(\mathbf{z}_{j,t}|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t^m) p(\mathbf{x}_{t-1}|\mathbf{z}^{m,t-1}, \mathbf{u}^{m,t-1}) \mathrm{d}\mathbf{x}_{t-1} \tag{4.19}$$

The objective is to select actions considering the current knowledge of the world (represented by the belief state) and the possible outcomes from actions carried by the robot. The computation of rational actions is considered including a payoff or reward function on the state $R(\mathbf{x}), \mathbb{R}^n \longrightarrow \mathbb{R}$. This payoff value can consider costs and revenues for being in a particular state. The objective is to find an optimal *policy*, that is, a function $\pi(\mathbf{x}) \rightarrow \mathbf{u}$, that determines what action $\mathbf{u}$ should be carried out if the state is $\mathbf{x}$. The policy should be optimal

in the sense that it should maximize the expected cumulative payoff (also called expected utility):

$$\pi^* = \arg\max_{\pi} E\big[\sum_{t=0}^{T} \gamma^t R(\mathbf{x}_t)|\pi\big] \tag{4.20}$$

that this, the expected sum of the rewards from time 0 to time T (T is called the planning horizon and $\gamma \in [0,1]$ is the discount factor).

If the environment is fully observable (that is, if we are certain about the state any time we get new information), but the effect of the actions carried out is non-deterministic, the problem is called a *Markov Decision Process* (MDP) [34]. If the more general case in which the environment is not observable, the problem is called *Partially Observable Markov Decision Processes* (POMDPs)[40].

Choosing a convenient payoff function, POMDPs can be used for developing perception actions; in this case, the payoff function usually considers higher values for narrower belief states, meaning that more informative belief states are preferred. However, the POMDP framework is usually infeasible in this case, as the number of unknown is high. The problem is even more important in the case of multiple robots, in which the problem to be solved is to determine an optimal policy that computes the actions of the robot of the fleet $\mathbf{u}^m$ given the global belief state $bel_m(\mathbf{x})$.

Another option for developing cooperative perception actions is to define a measurement of the information gain obtained when executing a certain information gathering task. Several different measurements of the information gain can be used. For instance, for unimodal distributions, the covariance could give a measure of the amount of information gained when executing an action (the bigger the covariance, the more uncertain about the actual value of the state). Another and more general measure about the information is the entropy of a probability distribution. The entropy $H(p)$ of a probability distribution $p(x)$ is defined as the expected value of the information $-\log[p(x)]$. That is, the entropy of the belief state is given by:

$$H(t) = E[-\log bel(\mathbf{x}_t)] = -\int bel(\mathbf{x}_t) \log bel(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t \tag{4.21}$$

The information gain is defined as the variation in the entropy after carrying an action $\mathbf{u}_{t+1}$. When executing this action, a new belief state $bel(\mathbf{x}_{t+1})$ will be obtained from the measurement $\mathbf{z}_{t+1}$ received, with a new entropy $H(\mathbf{u}_{t+1}, \mathbf{z}_{t+1})$. However, only the action $\mathbf{u}_{t+1}$ can be selected. Therefore, what can be computed is the expected entropy considering the potential measurements $H(\mathbf{u}_{t+1}) = E_{\mathbf{z}_{t+1}}[H(\mathbf{u}_{t+1}, \mathbf{z}_{t+1})]$. Therefore, the information gain associated to action $\mathbf{u}_{t+1}$ is defined as follows:

$$I(\mathbf{u}_{t+1}) = H(t) - E_{\mathbf{z}_{t+1}}[H(\mathbf{u}_{t+1}, \mathbf{z}_{t+1})] \tag{4.22}$$

This metric can be used to establish preferences among actions. A policy that at the same time maximizes the information gain and minimices action costs can be used for developing cooperative actions for perception purposes.

The decisional architecture for multi-robot coordination presented in Chap. 2 can consider coordinated information gathering tasks, as cooperative detection and confirmation, and cooperative monitoring. Nevertheless, the previous considerations are not yet included and the actions are selected following off-line learned models of performance (like detection rates and precision in localization tasks).

4.3 Vision-Based Object Detection, Localization and Tracking with Multiple UAVs

4.3.1 Initial Assumptions

In order to test the decentralized perception scheme presented above, the application to the detection, localization and tracking of events of interest by a fleet of UAVs is considered. The objective of the fleet is to detect potential targets, estimate their positions and also discard false alarms.

The important information for this scenario is the location of the UAVs and the location (and perhaps other information) of the events of interest. Then, the state to be estimated in this case, in its most simple version, is comprised by the information associated to a set of N events and by the position of the M UAVs.

It is assumed that the UAVs carry on-board Global Positioning System (GPS) receivers, Inertial Measurement Units (IMUs) and other sensors, different from cameras, for navigation. These sensors are used to localize the UAVs in a common global frame, and thus the localization problem is not considered here. However, as it will be seen, the conditional dependence of the measurements obtained by the cameras on the robot pose estimates (denoted by $\mathbf{q}_t$) must be taken into account.

The second assumption is that the actions of the robots will not affect the state of the objects to be tracked. This assumption usually holds when the objects to be tracked are not aware of the robots patrolling, for instance, if the objects of interest are fire alarms, or are not *evading* targets, but this would not hold for evading targets or opponent robots [51].

4.3.2 Event State Definition

In the general case of event detection and localization, the state $\mathbf{x}$ to be tracked obviously include the position of the object $\mathbf{p}_t$. If a moving object is considered, the velocity $\dot{\mathbf{p}}_t$ is also included into the state to be estimated. This will be called the kinematic part of the state.

Further information will be also needed. An important objective in some missions is to confirm that an object belongs to a certain class within a set $\mathbf{\Gamma}$ (for instance, in the case of fire alarms detection, this set will include as classes fire alarms and false alarms). Therefore, the state will include information regarding the classification of the object. Also, in certain applications, some appearance information could be needed to characterize an event, which also can help in the

Fig. 4.6. An infrared and a visual image of a fire alarm. The objective is to fuse all the data from the different vehicles of the fleet to obtain a global belief on the parameters of a given alarm.

task of data association between different UAVs with different cameras. This kind of information usually will be static, and will be represented by $\boldsymbol{\theta}$.

The complete state to be estimated is composed by the status of all the events. The number of events (N_t) can vary with the time. The state at time t is then represented by vector $\mathbf{x}_t = [\mathbf{x}_{1,t}^T, \ldots, \mathbf{x}_{N_t,t}^T]^T$. Each potential alarm i is defined by:

$$\mathbf{x}_{i,t} = \begin{pmatrix} \mathbf{p}_{i,t} \\ \dot{\mathbf{p}}_{i,t} \\ \boldsymbol{\theta}_i \end{pmatrix}. \tag{4.23}$$

4.3.3 The Likelihood Function for Vision

The key point in the Bayesian framework adopted is to determine the likelihood function $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{q}_t)$ for the case of the cameras. Notice that the (uncertain) pose of the UAV is included in the model, and to determine the final likelihood this dependence will be marginalized out.

In the limit, $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{q}_t)$ would be a probabilistic model of image formation, that is, the probability that the pixels of the image have certain illuminance or colour values given the status of the environment and the position of the sensor. In most cases it is not needed such a complex model. Instead of the raw pixels, some features related to the application are extracted from the images. Nevertheless, it is required to formulate probabilistically all these steps in order to incorporate the measurements in the estimation process.

Feature Extraction: Segmentation Algorithms

In the application considered (object detection and localization), the images captured by the cameras on-board the UAVs should be analyzed looking for the objects of interest. The image processing functions should be able to segment the objects of interest on the image plane, and differentiate them from the background. Moreover, the algorithms will obtain a set of features related to the

identity of the object $\boldsymbol{\theta}$. In order to include this into the probabilistic framework, it is needed to relate the visual features to the state through:

$$p(\mathbf{z}_t|\mathbf{x}_t) = p(\mathbf{z}_t|\mathbf{p}, \dot{\mathbf{p}}, \boldsymbol{\theta}), \boldsymbol{\theta} \in \boldsymbol{\Gamma} \tag{4.24}$$

These features will depend on the application considered. In general, determining the likelihood function will consist on a learning phase over the sensorial space; this learning phase should provide a map between features and classes.

In the particular application of fire detection, the space state for the identity part of the state $\boldsymbol{\theta}$ consists only on two potential values, $\boldsymbol{\Gamma} = \{\text{fire, no fire}\}$. Chapter 8 will present the techniques employed for forest fire detection using cameras (of different modalities, both infrared and visual). These algorithms provide directly a binary decision over the image plane, so that the measurements $\mathbf{z}_t$ are a set of blobs over the image plane classified of fire. As it will be seen, it is important also to employ the negative information encoded in the regions of the image not classified as fire.

There is always the chance for *false positives* and *misdetections*. False positives occur when the algorithm detects objects but there are no objects of the considered class in the field of the camera. Misdetections happen if no response is given when an object is present. These two facts have to be used to determine the likelihood function associated to the processed images.

A simple model will be used. It consists of characterizing the segmentation algorithms by two values:

- The probability of detection (P_{D}), defined as the likelihood that, being an object of the given class within the field of view, the object is effectively detected $p(\mathbf{z}_t|\mathbf{p}, \dot{\mathbf{p}}, \boldsymbol{\theta} = \text{fire})$.
- The probability of false positive (P_{F}), defined as the likelihood that the algorithm generates a response when no actual object is within the field of view of the camera $p(\mathbf{z}_t|\mathbf{p}, \dot{\mathbf{p}}, \boldsymbol{\theta} = \text{no fire})$.

Projective Geometry

In order to complete the main aspects of the likelihood function, it is needed to relate objects on the image plane with the position of objects in the 3D world (the location part of the state $\mathbf{p}_t$). Cameras project points in the space into points on the image plane. Cameras are usually modeled using the tools of projective geometry [18]. The projection is modeled by the *pin-hole* projection model. Following this model, each point in the space, $\mathbf{p}$ and its corresponding image pixel $\mathbf{m}$ on the image plane of the camera are related by (4.25), where $\mathbf{p}$ and $\mathbf{m}$ are in homogeneous coordinates:

$$s\mathbf{m}_t = \mathbf{A}\left(\mathbf{R}_t \; -\mathbf{t}_t\right)\mathbf{p}_t \tag{4.25}$$

where $\mathbf{A}$ is the upper triangular internal calibration matrix of camera. $\mathbf{R}_t$ and $\mathbf{t}_t$ are the rotation and translation that refer the camera coordinate system and a global reference frame, and are part of the estimated state of the UAV $\mathbf{q}_t$. Several

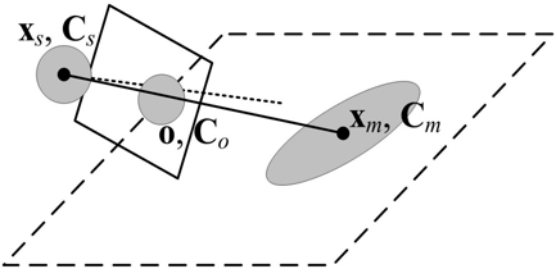

Fig. 4.7. Propagation of uncertainties through the geolocation procedure

methods can be used for camera calibration. Here, the method presented by [55] is used.

Equation (4.25) implies a non-linear relation between the state and the measurements. Moreover, if the pose is uncertain, it has to be considered when obtaining the corresponding likelihood.

Putting together the geometric model of the camera with the probabilistic characterization of the segmentation algorithms allow to establish a likelihood function for the image-base measurements obtained by the UAVs. $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{q}_t)$ would be the probability of getting the features extracted (that is, the objects segmented on the image plane) given the state of the events $\mathbf{x}_t$ and the position and orientation of the camera $\mathbf{q}_t$.

Prior Position Information from Cameras

As it will be seen, sometimes it is needed to determine the position of an object of interest from its position on the image plane. For instance, when an object is detected by the first time, one should obtain an initial estimation of its position $\mathbf{p}$.

Unfortunately, relation (4.25) is not invertible. Therefore, although $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{t}$ may be known, from the position on the image plane of an object $\mathbf{m}$, it is not possible to recover its 3D position $\mathbf{p}$. If nothing more is known, cameras provide only bearing information, and the full 3D position is not observable.

Several images from different points of view of the same object can be used to estimate the range of a point, and thus its 3D position, by triangulation, as in stereo vision [3, 9]. Also, if a Digital Elevation Map (DEM) of the scene is available, and the pose of the camera is known in the same coordinate frame than the DEM, then it is possible to estimate the position of an object on the ground by ray tracing techniques, provided that the camera is calibrated ($\mathbf{A}$ is known). Through this Chapter, it is assumed that such a map exists. In some applications, this kind of map could be provided by an UAV commanded to generate a DEM of the scenario [20]. Using the geolocation procedure, the UAVs will provide as measurements direct estimations on the position of potential alarms.

4.4 Local UAV Image Processing Techniques

The different UAVs will process locally their cameras in order to detect and track events. Interesting events could be mobile objects on a zone, fire alarms, and others. This section focuses on feature extraction and tracking and other set of basic functionalities local to each UAV, while Chapter 8 will show the fire segmentation techniques. These local functionalities are needed for cooperative detection and monitoring activities. The algorithms refer to vision, which currently is the most used exteroceptive sensor on-board UAVs.

4.4.1 Image Motion Estimation

Many algorithms and applications require as input an estimation on the image motion (for instance, mobile object detection or motion compensation, as it will be seen). Dense motion fields (or optical flow) has been used for monitoring activities using UAVs [8]. However, the computation of dense motion fields usually requires small displacement between consecutive images, which usually is not the case when an UAV is moving. Here, a method based on matching point-like features allows to obtain a sparse image motion field under less constrained motion. The image matching method presented is related to the described in [12], although significant improvements have since been made, as summarized in [30].

In [12] corner points were selected using the criteria described in [48]; each point was the centre of a fixed-size window which is used as a template in order to build matching window sequences over the stream of video images. Window selection provides for initial startup of window sequences as well as candidates (called direct candidates) for correlation-based matching tries with the last known template window of a sequence. The selection of local maxima of the corner detector function assured stable features, so window candidates in any given image were usually near the right matching position for some window sequence.

The correlation-based matching process with direct candidates within a search zone allowed to generate a matching pair data base, which described possibly multiple and incompatible associations between tracked sequences and candidates. A disambiguation process selected the right window to window matching pairs by using two different constraints: least residual correlation error and similarity between clusters of features.

The new approach uses the same feature selection procedure, but its matching strategy is significantly different. First, the approach ceases to focus in individual features. Now clusters are not only built for validation purposes; they are persistent structures which are expected to remain stable for a number of frames, and are searched for as a whole. Second, the disambiguation algorithm changes from a relaxation procedure to a more efficient predictive approach, similar to the one used in [3] for contour matching. Rather than generating an exhaustive data base of potential matching pairs, only selected hypothesis are considered. Each hypothesis, with the help of the persistent cluster data base, allows to define reduced search zones for sequences known to belong to the same cluster as the hypothesis, if a model for motion and deformation of clusters is known.

For this approach to be feasible, the following points should be defined

- Cluster model.
- Cluster building and validation.
- Cluster based prediction.

Cluster Model

The similarity of shape between regions of different images is verified by searching for clusters of windows whose members keep the same relative position, after a scale factor is applied. For a cluster of window sequences $\Gamma = \{\Phi_1, \Phi_2, \ldots, \Phi_n\}$, the shape similarity constraint is given by the following expressions:

$$\left| \frac{\|w_{\Phi_k}, w_{\Phi_l}\|}{\|v_{\Phi_k}, v_{\Phi_l}\|} - \frac{\|w_{\Phi_p}, w_{\Phi_q}\|}{\|v_{\Phi_p}, v_{\Phi_q}\|} \right| \leq k_p \ , \ \forall \Phi_k, \Phi_l, \Phi_p, \Phi_q \in \mathit{\Gamma} \tag{4.26}$$

In (4.26), k_p is a tolerance factor, w_i are candidate windows in the next image and v_i are template windows from the preceding image. The constraint is equivalent to verify that the euclidean distances between windows in both images are related by a similar scale factor; thus, the ideal cluster would be obtained when euclidean transformation and scaling can account for the changes in window distribution. Furthermore, an additional constraint over the maximum difference of rotation angle between pair of windows is used:

$$|\alpha_{\Phi_k, \Phi_l} - \alpha_{\Phi_p, \Phi_q}| \leq \gamma_p \ , \ \forall \Phi_k, \Phi_l, \Phi_p, \Phi_q \in \mathit{\Gamma} \tag{4.27}$$

where α_{rs} is the rotation angle of the vector that links windows from sequences r and s, if the matching hypothesis is accepted, and γ_p is a tolerance factor. Although the cluster model is adequately simple and seems to fit the current applications, more realistic local models such as affine or full homography could be easily integrated in the same scheme.

Cluster-Based Prediction for Hypotesis Propagation

It is easy to verify that two hypothesized matching pairs allow to predict the position of the other members of the cluster, if their motion can be modeled approximately by euclidean motion plus scaling, as the constraints (4.26) and (4.27) imply. Using this model, the generation of *candidate clusters* for a previously known cluster can start from a primary hypothesis, namely the matching window for one of its window sequences. This assumption allows to restrict the search zone for other sequences of the cluster, which are used to generate at least one secondary hypothesis. Given both hypothesis, the full structure of the cluster can be predicted with the small uncertainty imposed by the tolerance parameters k_p and γ_p, and one or several candidate clusters can be added to a data base.

For a specific cluster, assuming that a primary hypothesis is available, the cluster candidate generation process involves three steps:

- Secondary hypothesis generation.
- Cluster-based prediction.
- Indirect candidate generation.

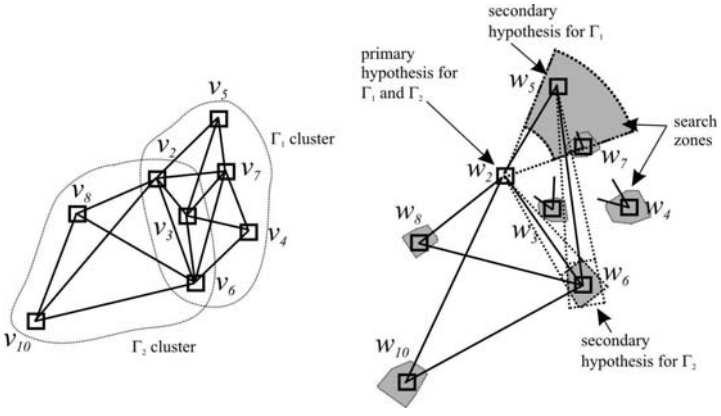

Fig. 4.8. Generation of cluster candidates

Secondary hypothesis are generated by single-window prediction. A search area can be defined by considering maximum limits of frame-to-frame cluster scaling and rotation, based on the knowledge of the environment; thus, the search of matching pairs can be restricted to the direct candidates contained in the area. Any window sequence linked to the cluster can be used as secondary hypothesis this way.

Several primary/secondary pairs can be obtained. For each available pair of primary/secondary hypothesis, a full cluster can be predicted; in this case, the tolerance parameters k_p and γ_p are used to further constraint the search zone of each component. Matching windows for each sequence can be found by exploring the set of direct or previously generated indirect candidates located within the computed limits.

If there remains some window sequences without candidate after the direct candidate expansion, new indirect candidates are generated by prediction of a suitable starting point for the error minimization algorithm.

The creation of any given candidate cluster can trigger the creation of others for neighbour clusters, provided that there is some overlap among them; in Fig. 4.8, for example, the creation of a candidate for cluster 1 can be used immediately to propagate hypothesis and find a candidate for cluster 2. Direct searching of matching windows is thus kept to a minimum. Indeed, it makes sense to consider a higher likelihood for candidates which already take part in a sufficiently large cluster candidate.

As Fig. 4.9 shows, successful candidate generation can be propagated through neighbor clusters. Two clusters can be said to be connected if they can be linked by an arbitrary long chain of overlapping clusters; the set of clusters can be partitioned in subsets of connected elements, or groups. The computational efficiency of the process is increased by maximizing propagation, so that candidates are reused whenever possible and direct search and error-minimization is avoided. In addition, window candidates which are already supported by a cluster candidate

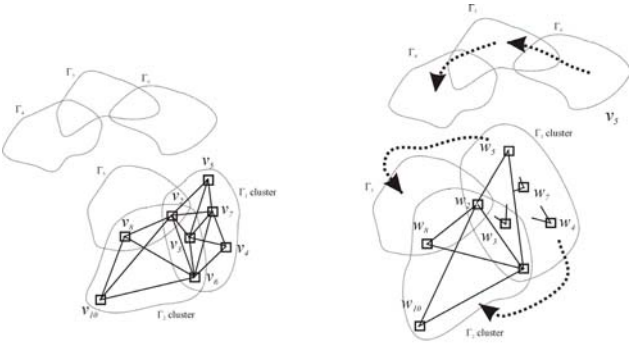

Fig. 4.9. Hypothesis propagation through cluster groups

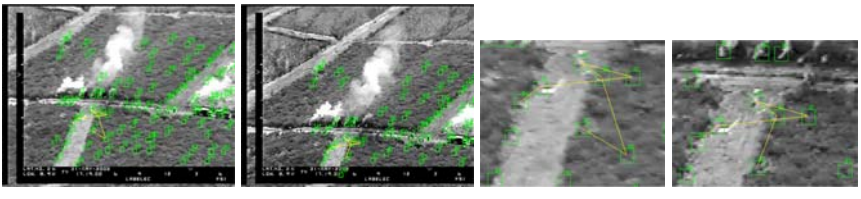

Fig. 4.10. Example of cluster matching

should be preferred to unsupported candidates. At the final stage of the method, the best cluster candidates are used to generate clusters in the last image, and determine the matching windows for each sequence. The practical result of the approach is to drastically reduce the number of matching tries, which are by far the main component of processing time when a great number of features have to be tracked, and large search zones are needed to account for high speed image plane motion. This is the case in non-stabilized aerial images, specially if only relatively low frame rate video streams are available. Figure 4.10 shows an example with two frames of an aerial sequence.

4.4.2 Blob Features

Although point-like features are suitable for image motion estimation, for certain applications, such as matching disparate views (for instance, views taken by *different* UAVs), features with more invariance properties are needed.

Homogeneity features are called *blobs* in scale-space theory [23]. In contrast to segmentation, blob detection does not attempt to segment out exact shapes of objects, but to extract robust and repeatable features discarding exact shapes and thin connections between patches (see Fig. 4.11). Blob features are also related to *maximally stable extremal regions* (MSER) [29]. MSER features are regions grown around an intensity extrema (maxima or minima) and are used to generate affine invariant frames, which are then used for view based object recognition [29].

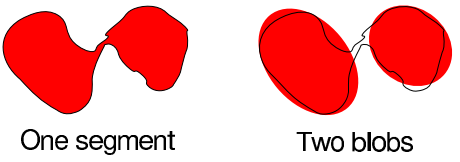

Fig. 4.11. Difference between segmentation and blob detection

Blob features can be extracted using a clustering pyramid built using robust estimation in local image regions [14, 13]. Each extracted blob is represented by its average colour $\mathbf{p}_k$, area a_k, centroid $\mathbf{m}_k$, and inertia matrix $\mathbf{I}_k$, i.e. each blob is a 4-tuple

$$\mathcal{B}_k = \langle \mathbf{p}_k, a_k, \mathbf{m}_k, \mathbf{I}_k \rangle \ . \tag{4.28}$$

Since an inertia matrix is symmetric, it has 3 degrees of freedom, and we have a total of $3+1+2+3=9$ degrees of freedom for each blob. Figure 4.12 gives a brief overview of the blob detection algorithm.

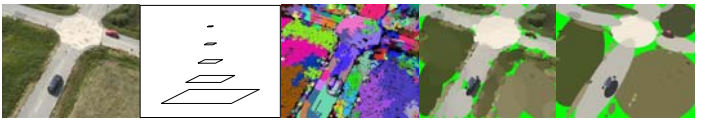

Fig. 4.12. Steps in blob detection algorithm

Starting from an image, the algorithm constructs a clustering pyramid, where a pixel $\mathbf{p}^*$ at a coarser scale is computed as the robust average of 12 pixels $\mathbf{p}_k$ at the lower scale

$$\arg\min_{\mathbf{p}^*} \sum_k r_k \rho(||\mathbf{p}_k - \mathbf{p}^*||) \, . \tag{4.29}$$

Regions where the support of the robust average is below a threshold $c_{\min}$ have their confidence flag r^* set to zero. The algorithm then creates a label image using the pyramid, by traversing it top-down, assigning new labels to points which have their confidence flag set, but do not contribute to any robust mean on the level above. The labelling produces a set of compact regions, which are then merged in the final step by agglomerative clustering. Regions with similar colour, and which fulfil the condition

$$M_{ij} > m_{\text{thr}} \sqrt{\min(a_i, a_j)} \tag{4.30}$$

are merged. Here M_{ij} is the count of pixels along the common border of blobs $\mathcal{B}_i$ and $\mathcal{B}_j$, and m_{thr} is a threshold. For more information about the feature

estimation, please refer to [14], and the implementation [1], both of these are available on-line. These blob features have been applied for view matching between different vehicles for multi-UAV localization purposes [26].

4.4.3 Image Stabilization

Monitoring activities are easier if the visualized scene is static; also, the processing algorithms are simpler because the reference positions are fixed along the digital process.

Considering UAVs with hovering capabilities, unavoidable control errors, turbulence and vibrations produce changes in the camera position which lead to image motion. Therefore, for certain applications it is necessary to solve the camera motion problem. Currently, electro-mechanic systems can solve this problem, but they are heavy, expensive and usually have a residual vibration.

The proposed technique to cancel the camera motion is based on the apparent motion computation between two consecutive image frames. The results presented in this section are obtained by using the robust point-like feature tracking method presented in Sect. 4.4.1, which provides a sparse image motion field. The same method can be used with different features. If there are enough of these features and they are equally distributed over the image, it will be possible to extract the scene apparent motion.

A homographic model will be used to represent image motion between consecutive frames. This model describes the transformations in the image plane when the scene is planar or, even being 3D scene, the camera undergoes a pure rotation (which can model the motion of a camera affected by vibrations).

The algorithm described here assumes that the percentage of objects with independent motion in the scene is low, and can be treated as outliers (objects with independent motion can mask the fundamental scene movement generated by vibrations or camera motion). The algorithm should be able to detect those outliers.

Finally, a homographic model is fitted to the computed sparse motion field between two consecutive images. Then, the inverse model is applied to all pixel positions in the current image to compensate the motion, this process is called *image warping*.

Homography Computation

A homography is any linear and invertible application from the projective space $\mathbf{P^2}$ into $\mathbf{P^2}$ [18]. These applications transform lines into lines. It is represented by a 3×3 invertible matrix defined up to scale factor, so it is composed by eight independent parameters.

If a set of points in the scene lies in a plane, and they are imaged from two viewpoints, then the corresponding points in images i and j are related by a plane-to-plane projectivity or planar homography, $\mathbf{H}$:

$$s\tilde{\mathbf{m}}_i = \mathbf{H}\tilde{\mathbf{m}}_j \tag{4.31}$$

where $\tilde{\mathbf{m}}_k = [u_k, v_k, 1]^t$ is the vector of homogenous image coordinates for a point in image k, $\mathbf{H}$ is a homography matrix and s is a scale factor. The same equation holds if the image to image camera motion is a pure rotation. Even though the hypothesis of planar surface or pure rotation may seem too restrictive, they have proved to be frequently valid for aerial images ([30, 6] show some examples). An approximate planar surface model usually holds if the UAV flies at a sufficiently high altitude, while an approximate pure rotation model is a good approximation for the motion induced by a hovering helicopter. Thus, the computation of $\mathbf{H}$ will allow under such circumstances to compensate for camera motion.

Since $\mathbf{H}$ has only eight degrees of freedom, we only need four matches to determine $\mathbf{H}$ linearly. In practice, more than four correspondences are available, and the overdetermination is used to improve accuracy. For a robust recovery of $\mathbf{H}$, it is necessary to reject outlier data.

The homography computation is summarized in Fig. 4.13. It can be seen that the computation is divided in two basic steps: *oulier rejection* and *robust estimation*. The first step tries to detect the outliers in order to increase the accuracy of the computed homography. In the proposed applications, outliers will not always be wrong matching pairs; image zones where the homography model does not hold (moving objects, buildings or structures which break the planar hypothesis) will also be regarded as outliers, although they may offer potentially useful information. The overall design of the outlier rejection procedure used in this work is based on LMedS [54].

In the second step, *robust estimation*, the homography is computed by using an iterative method (M-Estimator) that allows to automatically weight the set of data in function of the residue in each iteration. It was selected a simple weighting function, concretely a Fair function, because the set of data were well filtered by the outlier rejection stage. The Fair M-Estimator will guarantee a good convergence to the correct solution in few iterations.

Image Warping

Given the homography that relates two images, a model with information about the motion of each pixel is known. This model can be used to compensate the existing motion in order to avoid camera vibrations or movements.

Thus, if $\mathbf{H}$ is the homography that represents the motion produced from image I to image I', the new position of the pixels of I after the compensation will be:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad \begin{array}{rl} k &= x * h_{31} + y * h_{32} + h_{33} \\ x' &= (x * h_{11} + y * h_{12} + h_{13})/k \\ y' &= (x * h_{21} + y * h_{22} + h_{23})/k \end{array} \tag{4.32}$$

In general, the transformed position $\mathbf{m}' = [x', y']^t$, corresponding to pixel $\mathbf{m} = [x, y]^t$, will not be an integer due to the algebraic operations.

It is necessary to define a method to assign an integer position to the transformed pixels in order to warp correctly the images. This problem can be seen in Fig. 4.14, where the point marks the position where the pixel has been placed.

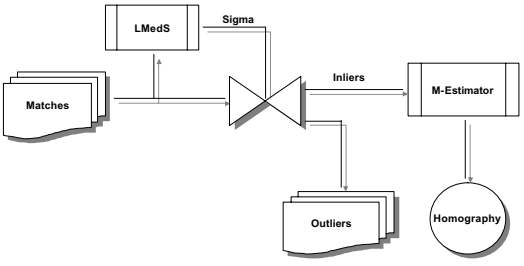

Fig. 4.13. Homography computation diagram

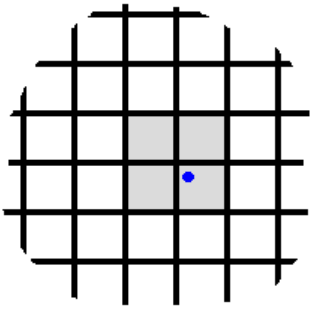

Fig. 4.14. Warping problem representation

Fig. 4.15. The sequences show a moving car and other UAV of the fleet detected using a motion detection algorithm based on the image stabilization procedure

The integer nature of the pixels force to select one of the four point neighbors (shadow pixels in the figure).

A method based on the *pixel similarity* is used. The technique uses the euclidean distance between the RGB value of the actual pixel and the values of the four neighbors in the previous image (shadow pixels in Fig. 4.14). The position of the pixel will be the one corresponding to the neighbor with the shortest distance. Thus, for each pixel, the method tries to minimize the RGB differences respect to the previous image. This helps to increase the alignment between sequenced images even correcting little errors in the homography computation.

The image stabilization procedure described above can be used for moving objects detection using monocular sequences of images. Given two consecutive images, and using the stabilization procedure, the two images can be warped to the same image coordinate frame. There, moving regions can be detected analyzing image differences. The moving regions will be due to independently moving objects or fixed objects with parallax respect to the plane used as reference for the warping procedure. Figure 4.15 shows some results.

4.5 Information Filter-Based Multi-UAV Detection and Localization Using Vision

The first approach presented in this Chapter employs Gaussian distributions to represent the belief state. They can be employed to represent unimodal beliefs, with a value with maximum probability (the mean) and uncertainty represented by the covariance of the distribution. They are used elsewhere to represent beliefs about positions of objects, etc.

Therefore, the prior knowledge at some time will be described by a Gaussian distribution $p(\mathbf{x}_0) \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. If the state at time t is a linear function of the state at time $t-1$ plus Gaussian noise $\boldsymbol{\nu}_t \sim N(\mathbf{0}, \mathbf{R}_t)$, and the measurements are a linear function of the state plus Gaussian noise $\boldsymbol{\varepsilon}_t \sim N(\mathbf{0}, \mathbf{S}_t)$:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \boldsymbol{\nu}_t, \tag{4.33}$$

$$\mathbf{z}_t = \mathbf{M}_t \mathbf{x}_t + \varepsilon_t, \tag{4.34}$$

then the Bayes filter (4.2) reduces to the well-known Kalman filter.

In a multi-robot application, the Information form of the Kalman filter or Information Filter (IF) [45, 46] has some interesting properties. Information filters are derived employing the canonical form of Gaussian distributions. In the canonical form, the parameters of a Gaussian distribution are the information vector $\boldsymbol{\xi}$ and the information matrix $\boldsymbol{\Omega}$. If $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance matrix of a Gaussian distribution, its information matrix is $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ and its information vector $\boldsymbol{\xi} = \boldsymbol{\Sigma}^{-1}\mu$. Algorithm 4.1 shows the Information Filter for updating the belief state $bel(\mathbf{x}_t) = \mathcal{N}(\boldsymbol{\Omega}_t^{-1}\boldsymbol{\xi}_t, \boldsymbol{\Omega}_t^{-1})$.

This representation has its advantages and drawbacks. One important advantage is that complete uncertainty is easily represented by $\boldsymbol{\Omega} = \mathbf{0}$.

A second interesting property is that the corresponding information matrix for the full trajectory is block-diagonal, and the space needed to store it is then linear with the trajectory length. Algorithm 4.2 shows the filter for the full belief trajectory. The functions **Augment_Matrix** and **Augment_Vector** augment the state space. However, as line 1 shows, the information matrix for the trajectory $\boldsymbol{\Omega}^t$ is block diagonal at any time (see Fig. 4.16).

Algorithm 4.1. $(\boldsymbol{\xi}_t, \boldsymbol{\Omega}_t) \leftarrow$ Information Filter$(\boldsymbol{\xi}_{t-1}, \boldsymbol{\Omega}_{t-1}, \mathbf{z}_t)$

1: $\boldsymbol{\Psi}_t = \mathbf{A}_t^{-T} \boldsymbol{\Omega}_{t-1} \mathbf{A}_t^{-1}$
2: $\bar{\boldsymbol{\Omega}}_t = \boldsymbol{\Psi}_t - \boldsymbol{\Psi}_t (\mathbf{R}_t^{-1} + \boldsymbol{\Psi}_t)^{-1} \boldsymbol{\Psi}_t$
3: $\bar{\boldsymbol{\xi}}_t = \bar{\boldsymbol{\Omega}}_t \mathbf{A}_t \boldsymbol{\Omega}_{t-1}^{-1} \boldsymbol{\xi}_{t-1}$
4: $p(\mathbf{x}_t | \mathbf{z}^{t-1}) \sim \mathcal{N}(\bar{\boldsymbol{\Omega}}_t^{-1} \bar{\boldsymbol{\xi}}_t, \bar{\boldsymbol{\Omega}}_t^{-1})$
5: $\boldsymbol{\Omega}_t = \bar{\boldsymbol{\Omega}}_t + \mathbf{M}_t^T \mathbf{S}_t^{-1} \mathbf{M}_t$
6: $\boldsymbol{\xi}_t = \bar{\boldsymbol{\xi}}_t + \mathbf{M}_t^T \mathbf{S}_t^{-1} \mathbf{z}_t$
7: $p(\mathbf{x}_t | \mathbf{z}^t) \sim \mathcal{N}(\boldsymbol{\Omega}_t^{-1} \boldsymbol{\xi}_t, \boldsymbol{\Omega}_t^{-1})$

Algorithm 4.2. $(\boldsymbol{\xi}^t, \boldsymbol{\Omega}^t) \leftarrow$Information Filter Trajectory$(\boldsymbol{\xi}^{t-1}, \boldsymbol{\Omega}^{t-1}, \mathbf{z}_t)$

1: $\bar{\boldsymbol{\Omega}}^t = \textbf{Augment\_Matrix}(\ \boldsymbol{\Omega}^{t-1}) + \begin{pmatrix} \begin{pmatrix} \mathbf{I} \\ -\mathbf{A}_t^T \end{pmatrix} \mathbf{R}_t^{-1} \begin{pmatrix} \mathbf{I} & -\mathbf{A}_t \end{pmatrix} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$

2: $\bar{\boldsymbol{\xi}}^t = \textbf{Augment\_Vector}\ (\boldsymbol{\xi}^{t-1})$

3: $\boldsymbol{\Omega}^t = \bar{\boldsymbol{\Omega}}^t + \begin{pmatrix} \mathbf{M}_t^T \mathbf{S}_t^{-1} \mathbf{M}_t & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$

4: $\boldsymbol{\xi}^t = \bar{\boldsymbol{\xi}}_t + \begin{pmatrix} \mathbf{M}_t^T \mathbf{S}_t^{-1} \mathbf{z}_t \\ \mathbf{0} \end{pmatrix}$

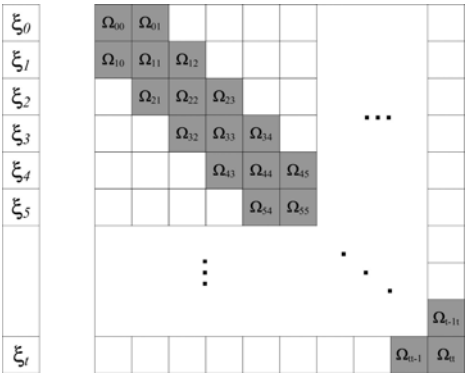

Fig. 4.16. Structure of the information matrix for the full trajectory. The information matrix is a block tridiagonal symmetric matrix.

4.5.1 Multi-robot Perception Employing Information Filters

It is straightforward to determine the Information Filter (IF) for the centralized multi-robot perception case. Recalling (4.7), the likelihood is given by $\prod_{j=1}^{M} p(\mathbf{z}_{j,t}|\mathbf{x}_t)$. If the assumptions of the IF hold, and as the local likelihood functions $p(\mathbf{z}_{j,t}|\mathbf{x}_t)$ are Gaussian, then this updating step consists of a sum of all the information contributions from all the robots of the fleet:

$$\boldsymbol{\Omega}_t = \bar{\boldsymbol{\Omega}}_t + \sum_{j=1}^{M} \mathbf{M}_{j,t}^T \mathbf{S}_{j,t}^{-1} \mathbf{M}_{j,t} \tag{4.35}$$

$$\boldsymbol{\xi}_t = \bar{\boldsymbol{\xi}}_t + \sum_{j=1}^{M} \mathbf{M}_{j,t}^T \mathbf{S}_{j,t}^{-1} \mathbf{z}_{j,t} \tag{4.36}$$

The centralized solution requires that the central node should receive all the information gathered by the UAVs. Even having a preprocessing stage carried on board the different UAVs, so that the central node does not receive the raw data, but only estimated features, the system does not scale well with the number of

robots. Moreover, in this case the central node should be within communication range at any time. However, the IF leads to a decentralized implementation in which each UAV maintains a local belief employing only local information, and then shares this belief with the local neighbors.

In this case, each UAV will run locally Algorithm 4.1 (with $\mathbf{A}_t = \mathbf{I}$ and $\mathbf{R}_t = \mathbf{0}$ if the state is static) or 4.2. And, for updating the status of its alarms, it will use the information obtained from its sensors. Of course, the estimated belief state will be different from the centralized one due to the missed terms in the updating steps 5 and 6 (the information collected by the rest of the robots).

When UAV i flies within communication range with other UAV j, they can share their beliefs. UAV i can apply the relation (4.15) to update its belief state. Denoting by $bel_{i,m}(\mathbf{x}_t)$ the resultant belief state of UAV i after incorporating the belief from UAV j, and employing the logarithmic form of the combination rule (4.16):

$$\begin{aligned} bel_{i,m}(\mathbf{x}_t) &= \log(C) + bel_i(\mathbf{x}_t) + bel_j(\mathbf{x}_t) - bel_j(\mathbf{x}_{t-1}) = \\ = \log(C) - \frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_t^i\mathbf{x} + \mathbf{x}^T\boldsymbol{\xi}_t^i - \frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_t^j\mathbf{x} + \mathbf{x}^T\boldsymbol{\xi}_t^j - \frac{1}{2}\mathbf{x}^T\boldsymbol{\Omega}_{t-1}^j\mathbf{x} + \mathbf{x}^T\boldsymbol{\xi}_{t-1}^j &= \\ = \log(C) - \frac{1}{2}\left[\mathbf{x}^T(\boldsymbol{\Omega}_t^i + \boldsymbol{\Omega}_t^j - \boldsymbol{\Omega}_{t-1}^j)\mathbf{x}\right] + \mathbf{x}^T(\boldsymbol{\xi}_t^i + \boldsymbol{\xi}_t^j - \boldsymbol{\xi}_{t-1}^j) & \end{aligned} \tag{4.37}$$

Comparing this with the logarithmic of the Gaussian distribution it can be seen that the resultant information vector and matrix are updated with the increase in evidence $\boldsymbol{\Omega}_t^{i,m} = \boldsymbol{\Omega}_t^i + (\boldsymbol{\Omega}_t^j - \boldsymbol{\Omega}_{t-1}^j)$ and $\boldsymbol{\xi}_t^{i,m} = \boldsymbol{\xi}_t^i + (\boldsymbol{\xi}_t^j - \boldsymbol{\xi}_{t-1}^j)$, where $\boldsymbol{\Omega}_{t-1}^j, \boldsymbol{\xi}_{t-1}^j$ in this case denotes the last received belief from UAV j (whenever it was received).

If there are more than one UAV within communication range, then, if $C(i)$ is the set of neighbors of UAV i:

$$\boldsymbol{\Omega}_t^{i,m} = \boldsymbol{\Omega}_t^i + \sum_{j \in C(i)} (\boldsymbol{\Omega}_t^j - \boldsymbol{\Omega}_{t-1}^j) \tag{4.38}$$

$$\boldsymbol{\xi}_t^{i,m} = \boldsymbol{\xi}_t^i + \sum_{j \in C(i)} (\boldsymbol{\xi}_t^j - \boldsymbol{\xi}_{t-1}^j) \tag{4.39}$$

If the state is static the amount of information needed to store the belief state (and to communicate it) is constant along time. Moreover, one UAV does not have to communicate continuously its belief, as it can accumulate evidence and transmit it later (without increasing the storage). Therefore, from the point of view of bandwidth requirements, it could be adjusted depending on the network conditions.

If the state is dynamic, the UAVs update and transmit their trajectory estimations. This estimation grows with time, although only linearly. Nevertheless, the amount of information to be transmitted should be bounded. Each UAV only stores a trajectory interval (the last 20 seconds). The rest of the trajectory is marginalized out, which maintains the sparse structure of the information matrix.

Eliminating Common Information

The previous equations, as commented in Sect. 4.2.3, assume that there are no loops in the network of UAVs, in the sense that the belief information shared

between two robots follows an unique path. If it is not the case, prior to combining the beliefs, the common information should be removed.

There are several options that can be considered for avoiding an overconfident estimation due to accounting common information several times. The first one is to ensure a tree topology in the belief network, as seen in Sect. 4.2.3. However, this option imposes a strong constraint on the potential communication links among the UAVs.

Another option is that each UAV only sends its last local information update at every time instant. This way, no information is duplicated. The problem in this case is that the use of an UAV as *data mule* is lost: one UAV that collects the evidence from a group of local neighbors will communicate it to other robots that could be initially disconnected from the firsts. Moreover, if the connection between two UAVs is lost, it will lose information that would have been available in the case that the robot had sent the complete belief.

The last option is to employ a conservative fusion rule, which ensures that the robot does not become overconfident. For the case of the information filter, there is an analytic solution for this, given by the *Covariance Intersection* (CI) algorithm [22]. The CI algorithm is a way of combining information from random variables whose cross-correlations are unknown.

Therefore, the conservative rule to combine the belief of UAV i with that received from the set $C(i)$ of local neighbors is given by:

$$\boldsymbol{\Omega}_t^{i,m} = \omega \boldsymbol{\Omega}_t^i + (1-\omega) \sum_{j \in C(i)} \boldsymbol{\Omega}_t^j \tag{4.40}$$

$$\boldsymbol{\xi}_t^{i,m} = \omega \boldsymbol{\xi}_t^i + (1-\omega) \sum_{j \in C(i)} \boldsymbol{\xi}_t^j \tag{4.41}$$

for $\omega \in [0\,1]$. It can be seen that the estimation is consistent in the sense that $\boldsymbol{\Sigma}^{i,m} - \hat{\boldsymbol{\Sigma}}^i \geq \mathbf{0}$ (where $\boldsymbol{\Sigma}^{i,m} = \boldsymbol{\Omega}^{i,m-1}$ is the estimated covariance matrix and $\hat{\boldsymbol{\Sigma}}^i$ is the actual covariance matrix) for any (unknown) cross-correlation matrix $\boldsymbol{\Sigma}^{ij}$, and for any ω. The value of ω can be selected following some criteria, as maximizing the obtained determinant of $\boldsymbol{\Omega}^{i,m}$ (minimizing the entropy of the final distribution). Another option is to use it as a weight that shows the UAV confidence in its own estimation and the neighbor's ones.

4.5.2 Decentralized Information Filter for Object Detection and Localization

Recalling what was presented in Sect. 4.3, the objective is to estimate the state of the potential alarms present in an scenario. In the general case that the objects move, the state will be the position and velocity of all the alarms, and its nature.

$$\mathbf{x}_t = [\mathbf{p}_{1,t}^T, \dot{\mathbf{p}}_{1,t}^T, \boldsymbol{\theta}_1, \ldots, \mathbf{p}_{N_t,t}^T, \dot{\mathbf{p}}_{N_t,t}^T, \boldsymbol{\theta}_{N_t}]^T \tag{4.42}$$

Each UAV will update its knowledge locally employing the Information Filter. The local estimations will be shared with the other UAVs of the fleet, which will

combine them by using the relations given by (4.40) and (4.41) in order to avoid overconfident estimates in the fully decentralized case.

Motion Model

In general, the motion of the alarms will not depend on the others. That is, at time t, $\mathbf{A}_t = diag\{\mathbf{A}_{1,t}, \ldots, \mathbf{A}_{N_t,t}\}$. Also $\mathbf{R}_t = diag\{\mathbf{R}_{1,t}, \ldots, \mathbf{R}_{N_t,t}\}$.

As a generic motion model of the targets, a discrete version of the continuous white noise acceleration model or second-order kinematic model is used [42]. In this model, the velocity is assumed to be affected by an acceleration modeled as a white noise of zero mean and with power spectral density $\boldsymbol{\delta}$. The discretized version of this linear motion model for an object i, is characterized by:

$$\mathbf{A}_{i,t} = \begin{pmatrix} \mathbf{I} & \Delta t\mathbf{I} \\ 0 & \mathbf{I} \end{pmatrix} \tag{4.43}$$

and

$$\mathbf{R}_{i,t} = \begin{pmatrix} \frac{1}{3}\Delta t^3\mathbf{I} & \frac{1}{2}\Delta t^2\mathbf{I} \\ \frac{1}{2}\Delta t^2\mathbf{I} & \Delta t\mathbf{I} \end{pmatrix} \boldsymbol{\delta} \tag{4.44}$$

Measurement Model and Likelihood Function

If the alarms are located on the ground, and if the geolocation procedure described above can be applied, each UAV can determine directly the 3D position of the objects segmented on the image plane. If a measurement j is associated to a particular alarm i, then:

$$\mathbf{M}_{ji,t} = \begin{pmatrix} \mathbf{I}_i & \mathbf{0} \end{pmatrix} \tag{4.45}$$

The errors $\boldsymbol{\varepsilon}_{j,t}$ in the estimated position arise due to the errors on the position and orientation of the sensor ($\mathbf{q}_t$) and the terrain model. Moreover, the geolocation procedure is non-linear. In order to propagate the uncertainties in $\mathbf{q}_t$ and obtain an estimation of the covariances of the error in (4.34) the Unscented Transform (UT) is used. Therefore, the UT is used to determine the mean $\mathbf{z}_{j,t}$ and covariance matrix $\mathbf{S}_{j,t}$ on the estimated position for any alarm detected on the image plane.

Prior Belief

The only issue to be described is how alarms are initialized. When an UAV segments an alarm, the geolocation procedure directly provides the initial values for $\boldsymbol{\xi}_i$ and $\boldsymbol{\Omega}_i$:

$$\boldsymbol{\Omega}_i = \begin{pmatrix} \mathbf{S}_{j,t}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{4.46}$$

and

$$\boldsymbol{\xi}_i = \begin{pmatrix} \mathbf{S}_{j,t}^{-1}\mathbf{z}_{j,t} \\ \mathbf{0} \end{pmatrix} \tag{4.47}$$

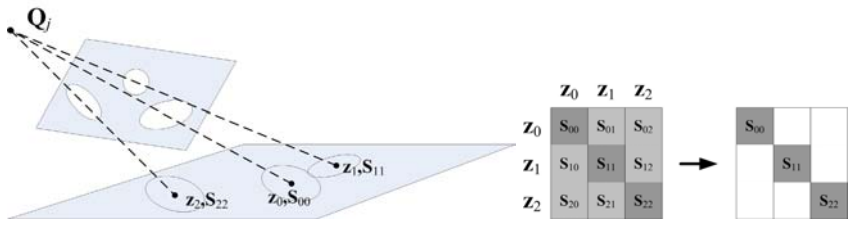

Fig. 4.17. The errors in $\mathbf{q}_t$ induce correlated errors in the estimated position of the alarms $\mathbf{z}_{i,t}$ (left). If the alarms are to be evolved independently, the dependences should be marginalized out (right).

Local Information Filter with Perfect Data Association

The previous equations are then used locally by each UAV to update their local beliefs about the current alarms. A further assumption allows to simplify the algorithm. As the motion model and motion noise $\boldsymbol{\nu}_t$ covariances are block diagonal, if the information matrix is block diagonal $\boldsymbol{\Omega}_{t-1} = diag\{\boldsymbol{\Omega}_{1,t-1}, \ldots, \boldsymbol{\Omega}_{N_t,t-1}\}$, each alarm i can be predicted separately leading to a (local) parallelized computation.

If the set of measurements can be associated with the current set of alarms, so that any measurement j is associated to an alarm i (that is, only one block $\mathbf{M}_{ji,t}$ for each row of matrix $\mathbf{M}_t$ is non zero) and if the measurements are independent (and thus, $\mathbf{Q}_t$ is block diagonal), the global Information Filter for all the alarms can be divided into N Information Filters, a separated one for each alarm.

However, the position measurements about the different alarms obtained by one UAV are not independent. When obtaining the position of the alarms, the errors on the UAV position $\mathbf{q}_t$ induce errors on the estimated position, errors that are correlated for all the alarms detected (see Fig. 4.17). Therefore, the dependencies among measurements should be explicitly marginalized out if one wants to keep the alarms updated independently. The marginalization is straightforward for the case of Gaussian measurements. It should be noted that getting rid of measurements dependencies reduce the amount of storage required and the computational burden, but it comes at a cost. Some information is lost: the relation between alarms. This information would allow to propagate information about one alarm to the other related (as in the SLAM problem).

Data Association

Previous sections have presented the filter under the assumption of perfect data association. That is, each measurement j is associated to one particular alarm i, or when an UAV receives information from other, it knows to which alarm it belongs.

The data association problem tries to determine what measurements correspond to what alarms, or what alarm corresponds to what alarm when combining beliefs received from other UAVs. The first one is usually called scan-to-track association, while the later is called track-to-track association. This general problem

is known to be NP-hard in the general case. As the main objective is to show cooperative characteristics, the approach followed is fairly simple, although more complex ones could be used.

For the scan-to-track association a gated nearest neighbor technique [10] is employed. For the track-to-track association, each alarm is marked with a label, but the labels are not synchronized, so the UAVs must determine the conversion. This is also accomplished by using a nearest neighbor technique, so that two alarms p and q of UAVs i and j are associated if the Mahalanobis distance is below a threshold:

$$d_{kp}^{ij2} = [\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_p^j]^T \boldsymbol{\Sigma}_{kp}^{ij-1} [\boldsymbol{\mu}_k^i - \boldsymbol{\mu}_p^j] \leq d_{\text{th}}^2 \tag{4.48}$$

In the case of combining belief trajectories, it is needed to recover the mean $\boldsymbol{\mu}^i$ for every alarm by solving the system $\boldsymbol{\Omega}^i \boldsymbol{\mu}^i = \boldsymbol{\xi}^i$. Also, it is needed to obtain the inverse of $\boldsymbol{\Sigma}^{ij}$. The information matrix $\boldsymbol{\Omega}^i$ for the case of the full trajectory (and, thus, matrix $\boldsymbol{\Sigma}^{ij}$) can be of the order of hundreds of rows/columns (for instance, for a 20 seconds trajectory and if each block row corresponds to one second, the matrix is 120×120). However, the information matrix is very structured, which allows for efficient algorithms for matrix inversion. In [2], the authors show how for a symmetric tridiagonal block matrix there exist algorithms that are nearly two orders of magnitude more efficient than direct inversion.

In the case of track-to-track association, once the same identifier has been associated to a local alarm the track is definitely associated, so that the data association problem becomes straightforward. However, this comes at cost that sometimes this could lead to wrong assignments. More complex data association techniques can be used. An approach based on the information representation of the posterior over all potential track-to-track associations is described in [37].

4.5.3 Results

This section presents some results obtained in simulation. The same algorithms have been applied for real experiments of fire detection and monitoring, which are presented in Chap. 8.

Experiment 1

In the first experiments, three vehicles are considered, and two static alarms. Figure 4.18 shows the simulated trajectories and the position of the alarms. One of the vehicles flies at a higher altitude, while the others are given a certain area to patrol. The vehicle flying at higher altitude acts as an information relay for the other two UAVs.

Figures 4.19 and 4.20 show the estimated positions and estimated variances for the two objects and 3 UAVs. The UAVs employ the Covariance Intersection to avoid overconfident estimates. It can be seen as all converge to the same solution and to the correct position of the alarms.

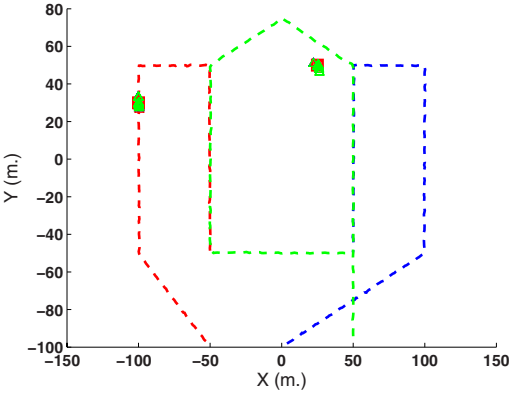

Fig. 4.18. Map showing the trajectories of the vehicles and the position of the events for Experiment 1

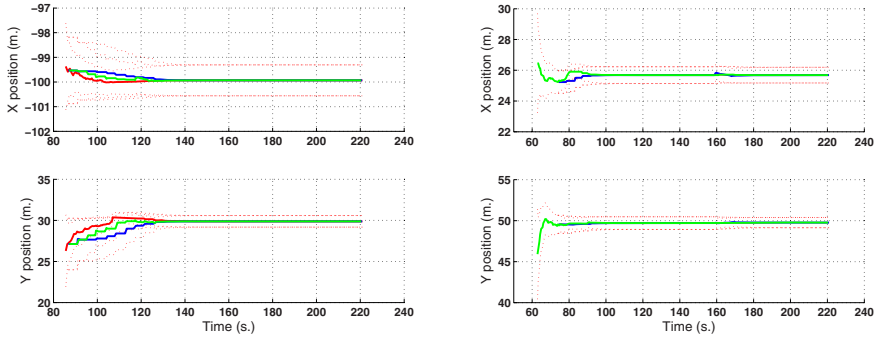

Fig. 4.19. Estimated position of both alarms for the three vehicles in Experiment 1

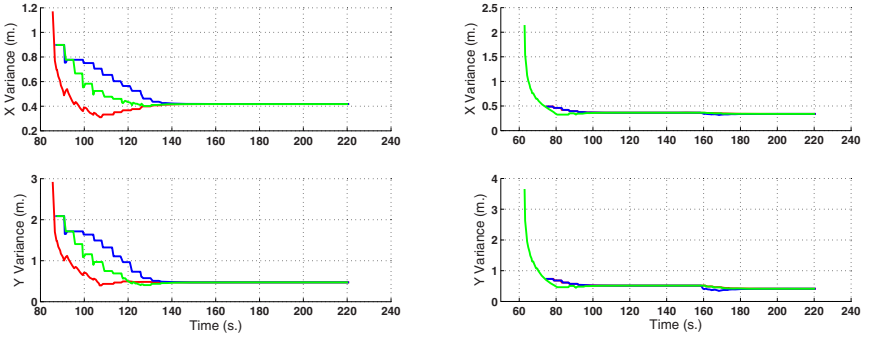

Fig. 4.20. Estimated variances of the errors for the three vehicles in Experiment 1

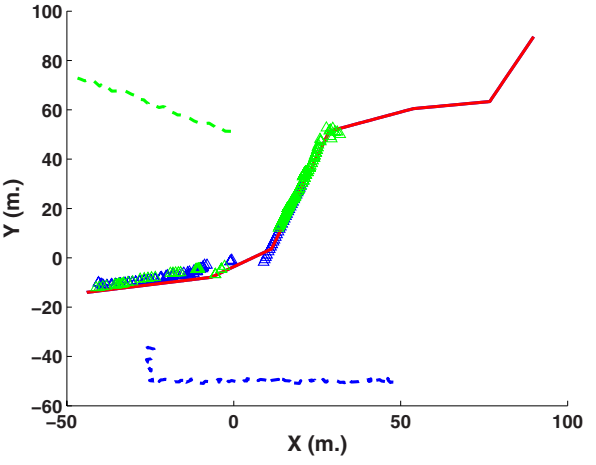

Fig. 4.21. Map showing the trajectory of the event (solid), the trajectories of the two vehicles (dashed) and the estimated position of the object (triangles) in Experiment 2

Experiment 2

The second experiment consists of the detection and tracking of a moving object. Two UAVs are commanded to fly to a certain point and hover looking to the object. Figure 4.21 shows the estimated position of a moving object by two UAVs using cameras.

Figure 4.22 shows a detail of the estimated position by the UAV located around position $(0, 50)$ against the actual one. It can be seen that first the object is within its field of view. One it abandons it, the uncertainty grows until

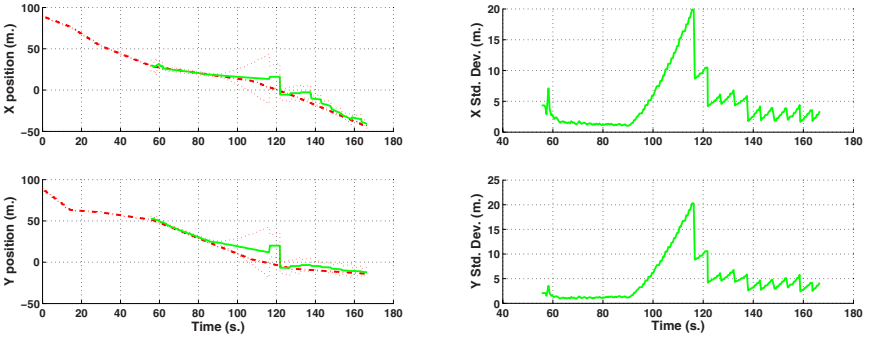

Fig. 4.22. Left: Estimated position of the object (solid) and actual one (dashed). Right: estimated variances for one of the UAVs. At time 115 it receives information from the other UAV, incorporating it into its own belief.

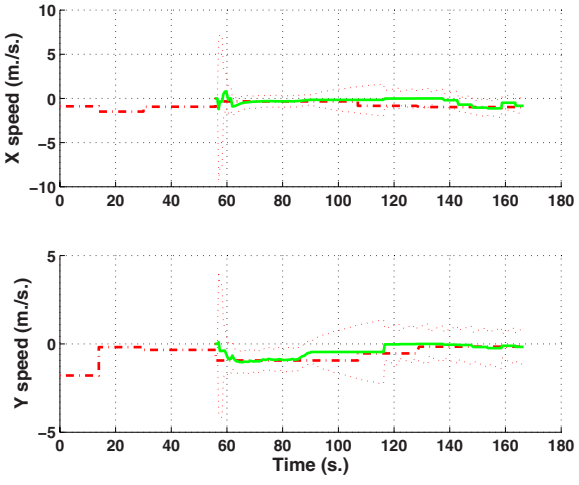

Fig. 4.23. Estimated velocity of the object (solid) and actual one (dashed)

it receives information from the other UAV. Also, the variances in the X and Y coordinates are shown. Figure 4.23 shows the estimated velocity of the object for the same UAV.

4.6 Grid-Based Multi-UAV Detection and Localization Using Vision and Other Sensors

The Information Filter (and its dual, the Kalman Filter), requires some restrictions that in some cases do not hold. For instance, they are restricted to Gaussian distributions, and thus they cannot handle multi-modal distributions, which arise when several hypotheses should be maintained at the same time. Also, the problem of data association has to be taken into account [42].

Furthermore, the previous method is suitable for sensors that can provide *contacts*. That is, the raw data that each sensor provides can be segmented into information that can be associated to a particular alarm. However, there are sensors that do not provide directly the position of objects of interest (indeed, cameras only provide bearing measurements, and only under the assumption of a known DEM the position of the objects on the ground can be determined, even for a perfectly localized camera). For instance, in Chapter 8 a fire detection application is presented which also uses fire detectors as sensors. Fire detectors are cheap sensors that provide information about the presence or absence of fire within their fields of view, but no direct information about the actual localization or size of the fire.

Grid-based approaches [27, 41, 46] can overcome the previous mentioned problems. The approach is to divide the scenario to be explored into cells, in what is called a certainty or evidence grid. To each cell k, a discrete (binary) random

Algorithm 4.3. $\{l_{k,t}\} \leftarrow$ Binary_LogOdds_Filter$(\{l_{k,t-1}\}, \mathbf{z}_t)$

1: **for** $k = 1$ to L **do**
2: $\quad l_{k,t} = l_{k,t-1} + \log p(\mathbf{z}_{i,t}|x_k) - \log p(\mathbf{z}_{i,t}|\bar{x}_k)$
3: **end for**

variable x_k is attached, representing the presence or absence of the object. Also, each cell of the grid has a 3D position associated, $\mathbf{p}_k$.

The variable can have two values, true or false. The probability that there is an object at cell k is represented by $p(x_k = 1)$ or $p(x_k)$, and we denote by $\bar{x}_k$ the fact that there is no object at cell k. Then, by definition, $p(x_k = 0) = p(\bar{x}_k) = 1 - p(x_k)$. The objective of the robot team is to update the probability of the different cells.

4.6.1 Local Filter

For this application, it is assumed that the status of each cell does not change with time. The state $\mathbf{x}_t$ is comprised by the status of all the cells of the grid $x_{k,t}$ at time t. The joint posterior $p(\mathbf{x}_t|\mathbf{z}^t)$ has to take into account all the possible combinations of values for all the cells. The full posterior for a grid with L cells should consider 2^L different states [46]. Maintaining this posterior is computationally unaffordable. Instead, this posterior will be approximated by the products of its marginals over each cell.

$$p(\mathbf{x}_t|\mathbf{z}^t) = \prod_k p(x_k|\mathbf{z}^t) \tag{4.49}$$

In this case, the Bayes filter takes a particular simple form considering log-odds:

$$\frac{bel(x_{k,t})}{1 - bel(x_{k,t})} = \frac{p(\mathbf{z}_t|x_k)}{p(\mathbf{z}_t|\bar{x}_k)} \frac{bel(x_{k,t-1})}{1 - bel(x_{k,t-1})} = \; = \prod_{\tau=0}^{t} \frac{p(\mathbf{z}_\tau|x_k)}{p(\mathbf{z}_\tau|\bar{x}_k)} \frac{bel(x_{k,0})}{1 - bel(x_{k,0})} \tag{4.50}$$

Calling $l_{k,t} = \log \frac{bel(x_{k,t})}{1-bel(x_{k,t})}$, then the local algorithm for updating all the cells of the grid is given by Algorithm 4.3.

4.6.2 Distributed Filter in the Multi-robot Case

If a centralized node receives all the data provided by the robots, as $p(\mathbf{z}_t^m|x_{k,t}) = \prod_j p(\mathbf{z}_{j,t}|x_{k,t})$ the central filter is the same as Algorithm 4.3, but line 2 is substituted by

$$l_{k,t} = l_{k,t-1} + \sum_j [\log p(\mathbf{z}_{j,t}|x_k) - \log p(\mathbf{z}_{j,t}|\bar{x}_k)] \tag{4.51}$$

The filter is easily decentralized. Each robot i computes part of the running total of (4.51) considering only local data, and sends to the other robots its own belief state in logarithmic form. Each robot incorporates the information received from others by using (4.37) adapted to the binary static case. The final filter is given by Algorithm 4.4.

Algorithm 4.4. Decentralized_Grid_Multi_Robot(i)

for all k **do**
  $l^i_{k,0} = 0$
end for
while true **do**
  for all k **do**
    if New data $\mathbf{z}_{i,t}$ **then**
      $l^i_{k,t} \leftarrow l^i_{k,t-1} + \log p(\mathbf{z}_{i,t}|x_k) - \log p(\mathbf{z}_{i,t}|\bar{x}_k)$
    end if
    if New belief from UAV j **then**
      $l^i_{k,t} \leftarrow l^i_{k,t} + l^j_{k,t} - l^j_{k,t-\Delta t}$
    end if
  end for
end while

4.6.3 Grid-Based Detection and Localization of Events

The main issue in the filter described is to determine the likelihood function $p(\mathbf{z}_t|x_k)$. It indicates the probability of having data $\mathbf{z}_t$ considering that there is an object of the class considered in cell k at time t. The data $\mathbf{z}_t$ consist of all the data gathered by the vehicles of the fleet at time t, and in the particular case considered, these data are images and fire sensor readings gathered by the different vehicles.

The likelihood function should take into account the position of the sensors respect to the map and the geometric characteristics of the distinct sensors (for instance the pin-hole model of the cameras). The latter are obtained through calibration, while the former will be provided by the UAVs. As commented above, the uncertainty on the pose of each UAV j, $\mathbf{q}_{j,t}$, should be taken into account for a correct definition of the likelihood function for the measurements gathered by this UAV $\mathbf{z}_{j,t}$:

$$p(\mathbf{z}_{j,t}|x_k) = \int p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t}) p(\mathbf{q}_{j,t}) \mathrm{d}\mathbf{q}_{j,t} \tag{4.52}$$

An equivalent equation to (4.52) is used to compute $p(\mathbf{z}_{j,t}|\bar{x}_k)$ (which is required in the filter). Equation (4.52) implies a further simplification. The measurements depend conditionally not only in one cell, but at least in all the cells of the grid within their field of view $S(j)$ of the sensor j. Therefore,

$$p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t}) = \sum_{i \in S(j)} \sum_{x_i} p(\mathbf{z}_{j,t}|x_k, x_i, \mathbf{q}_{j,t}) p(x_i) \tag{4.53}$$

However, when computing (4.52) this dependence will not be considered. The rest of the section describes the likelihood functions for several sensors.

Measurement Model for the Cameras

As presented in Sect. 4.3.3, cameras are an important source of information in the fleet of UAVs. Cameras of different modalities can be considered, but the first

assumption is that in a preprocessing stage, objects of interest are segmented from the background, so that the measurements provided by the cameras are binary images that classify the pixels as belonging to the class or not (as fire or no fire for instance). This segmentation algorithms are defined by the probabilities $P_{\mathrm{D},j}$ and $P_{\mathrm{F},j}$ for sensor j.

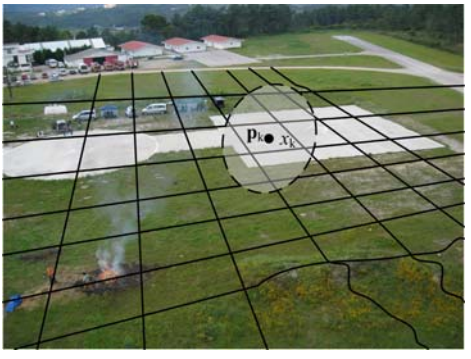

Fig. 4.24. Due to the uncertainties in sensor position and the cell resolution, one cell corresponds to a zone on the image plane (represented by the ellipse)

In order to determine the likelihood function (4.52), it should be considered the following. Each cell k has a position associated, $\mathbf{p}_k$. For a given value of the position of the sensor $\mathbf{q}_{j,t}$, the center of the cell will correspond to a pixel $\mathbf{m}_{k,j}$ on the image plane of camera j (if it is within the field of view of that camera). The pixel position is given by (4.25).

If pixel $\mathbf{m}_{k,j}$ corresponds to a region segmented as fire, then the likelihood is defined by $p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t})$:

$$\begin{aligned} p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t}) &= P_{\mathrm{D},j} \\ p(\mathbf{z}_{j,t}|\bar{x}_k, \mathbf{q}_{j,t}) &= P_{\mathrm{F},j} \end{aligned} \tag{4.54}$$

while if the pixel is classified as background, then the term is given by:

$$\begin{aligned} p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t}) &= 1 - P_{\mathrm{D},j} \\ p(\mathbf{z}_{j,t}|\bar{x}_k, \mathbf{q}_{j,t}) &= 1 - P_{\mathrm{F},j} \end{aligned} \tag{4.55}$$

However, the position of the sensor is not known accurately, and thus, the position of the corresponding pixel is also uncertain. So in order to compute the likelihood $p(\mathbf{z}_{j,t}|x_k)$, (4.52) should be integrated for possible values of the pose of the camera $\mathbf{q}_{j,t}$. This could be done by sampling values of $\mathbf{q}_{j,t}$. However, this is computationally expensive, and indeed this should be done for all the cells of the grid that are within the field of view of the camera.

The pixel position is related to the sensor and cell position through the non-linear pin-hole model of (4.25), so that $\mathbf{m}_{kj} = \mathbf{f}(\mathbf{q}_{j,t}, \mathbf{p}_k)$. Instead of directly

solving (4.52), the uncertainties in $\mathbf{q}_{j,t}$ are propagated into uncertainties on the pixel position $\mathbf{m}_{kj}$ corresponding to cell k using the Unscented Transform [21]. Moreover, in the procedure, the uncertainties in the position $\mathbf{p}_k$ due to the limited resolution of the grid are also taken into account. As a result, each cell k corresponds to a Gaussian distribution on the pixel position $p(\mathbf{m}_{kj})$. And then, equation (4.52) becomes:

$$p(\mathbf{z}_{j,t}|x_k) = \int p(\mathbf{z}_{j,t}|x_k, \mathbf{q}_{j,t}) p(\mathbf{q}_{j,t}) \mathrm{d}\mathbf{q}_{j,t} = \sum_{\mathbf{m}} p(\mathbf{z}_{j,t}|\mathbf{m}_{kj,t}) p(\mathbf{m}_{kj,t}) \tag{4.56}$$

where the sum is done over a region on the image plane determined by the second order moments of the distribution $p(\mathbf{m}_{jk,t})$. The same procedure is used to compute the likelihood function for the hypotheses $\bar{x}_k$.

To complete the model, the probabilities $P_{\mathrm{D},j}$ and $P_{\mathrm{F},j}$ are modified depending mainly on the relative position of the cell k, $\mathbf{p}_k$, respect to the position and orientation of the sensor $\mathbf{q}_j$. Thus:

$$\begin{aligned} P_{\mathrm{D},j}(\mathbf{p}_k, \mathbf{q}_j) &= P_{\mathrm{D},j} - w_{\mathrm{D},j}(d_{kj}^2) \\ P_{\mathrm{F},j}(\mathbf{p}_k, \mathbf{q}_j) &= P_{\mathrm{F},j} - w_{\mathrm{F},j}(d_{kj}^2) \end{aligned} \tag{4.57}$$

where $w_{\mathrm{D},j}$ and $w_{\mathrm{F},j}$ are functions that decrease the values of $P_{\mathrm{D},j}$ and $P_{\mathrm{F},j}$ with the distance between cell k and sensor j, d_{kj}. These are the actual values employed by the likelihood function.

Measurement Model for the Fire Sensor

The fire sensor considered is a fire detector, whose main component is a photodiode set-up to limit its sensibility to the band of $[185, 260]$ nm, normally associated to fires. The output of the sensor is a scalar value, proportional to the radiation received. Being a magnitude sensor, it is not possible to determine if a measure is due to a big fire far away or a nearby small fire. Using a threshold, this value is used to indicate if a fire is present or not within the field of view of the sensor.

Thus, the functioning of the sensor can be also characterized by probabilities of detection P_{D} and false positive generation P_{F}. These probabilities depend on the threshold selected. A higher threshold implies a lower P_{F} at a cost of worse detection capabilities.

If the sensor detects something, then for the updating of the cells the following values hold:

$$\begin{aligned} p(\mathbf{z}_{j,t}|x_k) &= P_{\mathrm{D},j}(\mathbf{p}_k, \mathbf{q}_j) = P_{\mathrm{D},j} - w_{\mathrm{D},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj}) \\ p(\mathbf{z}_{j,t}|\bar{x}_k) &= P_{\mathrm{F},j}(\mathbf{p}_k, \mathbf{q}_j) = P_{\mathrm{F},j} - w_{\mathrm{F},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj}) \end{aligned} \tag{4.58}$$

and, in case that the sensor does not detect anything:

$$\begin{aligned} p(\mathbf{z}_{j,t}|x_k) &= 1 - P_{\mathrm{D},j}(\mathbf{p}_k, \mathbf{q}_j) = 1 - [P_{\mathrm{D},j} - w_{\mathrm{D},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj})] \\ p(\mathbf{z}_{j,t}|\bar{x}_k) &= 1 - P_{\mathrm{F},j}(\mathbf{p}_k, \mathbf{q}_j) = 1 - [P_{\mathrm{F},j} - w_{\mathrm{F},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj})] \end{aligned} \tag{4.59}$$

In fact, this model could be used for any other presence sensor that provides binary decisions about the presence or absence of an object of a certain kind within its field of view.

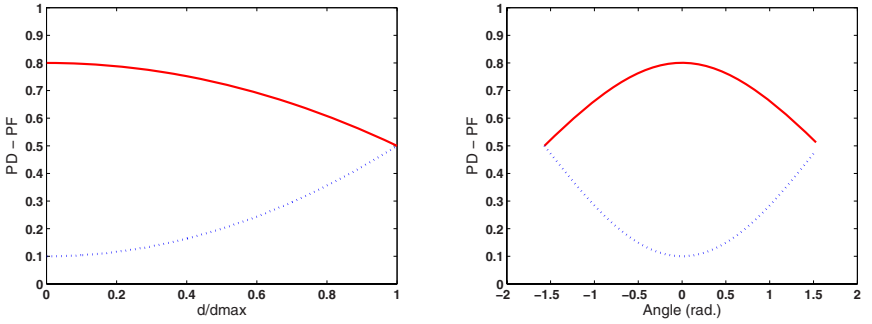

Fig. 4.25. Plots of equations $P_{\mathrm{D},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj})$ and $P_{\mathrm{F},j}(d_{kj}^2, \alpha_{kj}, \theta_{kj})$. Left, distance component; right, angular component. Solid, P_{D}, dotted, P_{F}.

Obtaining Measures from the Grid

Using the equations described above, the status of the grid is recursively estimated using the data the vehicles are providing. From a Bayesian point of view, the grid represents all the information about the possible alarms at time t. However, in some applications, more specific measures are required. For instance, if a fleet is looking for fire alarms, a control center would expect the position of the potential fire alarm detected, in order to plan a new mission, sending new vehicles to confirm the alarm. Also, we will use this value to compare it with the position of the fire recorded with GPS for validation purposes.

This can be accomplished in various ways. In this case, the set of cells of the grid with probabilities over a given threshold is obtained every T seconds. An alarm is raised for each set R of connected cells over this threshold. The position of the alarm is computed as the weighted geometric mean of the positions of the cells.

$$\mu_R = \frac{\sum_{k \subset R} \mathbf{p}_k p(x_k|\mathbf{z}^t)}{\sum_{k \subset R} p(x_k|\mathbf{z}^t)} \tag{4.60}$$

Also, it can be obtained an estimation of the uncertainty on the computed position from the second order moments of the region R.

Experimental Results

Figure 4.26 shows the evolution of the grid of one UAV in several phases of an actual fire experiment carried out in the framework of the COMETS project, and that are described in Chap. 8. The first figure shows the status of the grid after one of the UAVs has flown over a place with no fire, using a fire detector. The second figure shows how the sensor produces two big high probability blobs on the grid, one due to a false alarm and other due to the actual alarm. Afterwards, another UAV takes off and uses its IR camera over the zone of the possible alarms. This UAV receives the estimated grid from the other one. The third grid shows how after several images and fire sensor data are integrated, the high

Fig. 4.26. The status of the gird at three moments during the mission. The filled square represents the actual position of the fire.

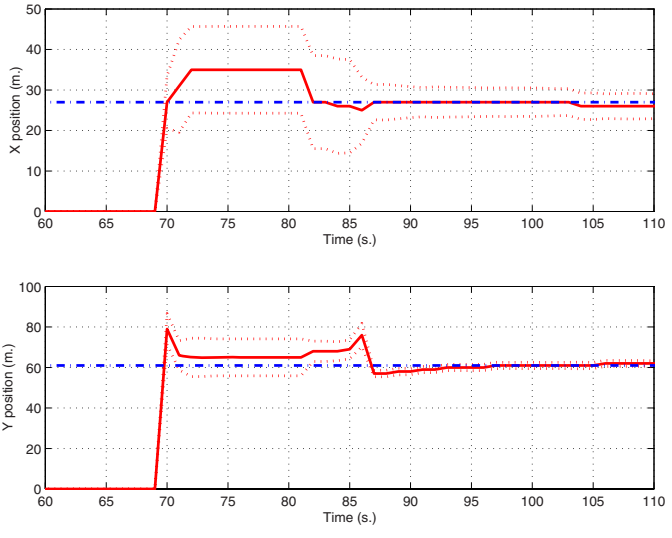

Fig. 4.27. Estimated mean position of one of the high probability regions. Dotted: estimated variances. Dash-dotted: actual fire position.

probability region is constrained to a smaller region, which includes the actual position of the fire.

Figure 4.27 shows the evolution of the position of the high probability regions computed using (4.60) compared to the actual fire position. It also shows the estimation on the uncertainty on the computed position.

4.7 Conclusions

UAV environment perception is a main issue in aerial robotics. UAV perception techniques include motion estimation from images, stabilization in sequences of images taken with on-board cameras subject to vibrations and UAV turbulences, automatic detection, classification and geo-localization of objects in the images, and UAV vision-based localization, which can be very useful in case of GPS

unavailability or failures. Different methods for the application of these techniques have been presented in this Chapter.

The Chapter has also shown that a probabilistic approach is suitable for cooperative detection, localization and tracking. Particularly, an information filter for multi-UAV cooperative perception has been presented showing good results in detection and tracking. Moreover, a grid-based method for detection and tracking using vision and other sensors has been also presented demonstrating its capability for fire detection and tracking.

The presented methods can be applied integrated in the decisional architecture presented in Chap. 2. Chapter 8 of this book will present experiments on forest fire detection and monitoring.

References

1. Blobdetect software. `http://www.isy.liu.se/~perfo/software/`.
2. A. Asif and J.M.F. Moura. Block matrices with L-block-banded inverse: inversion algorithms. *IEEE Transactions on Signal Processing*, 53(2):630–642, Feb. 2005.
3. N. Ayache and P. T. Sander. *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*. The MIT Press, Cambridge, MA, USA, 1991.
4. T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.
5. W. Burgard, M. Moors, C. Stachniss, and F.E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, June 2005.
6. F. Caballero, L. Merino, J. Ferruz, and A. Ollero. A visual odometer without 3d reconstruction for aerial vehicles. applications to building inspection. In *Proceedings of the International Conference on Robotics and Automation*, pages 4684–4689. IEEE, April 2005.
7. D.W. Casbeer, D.B. Kingston, R.W. Bear, T.W. McLain, and S.M. Li. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of System Science*, pages 1–18, January 2005.
8. G. Farnebäck and K. Nordberg. Motion detection in the WITAS project. In *Proceedings SSAB02 Symposium on Image Analysis*, pages 99–102, Lund, March 2002. SSAB.
9. O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. The MIT Press, Cambridge, MA, USA, 1993.
10. H. J. S. Feder, J. J. Leonard, and C. M. Smith. Adaptive Mobile Robot Navigation and Mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.
11. J.W. Fenwick, P.M. Newman, and J.J. Leonard. Cooperative concurrent mapping and localization. In *Proceedings of the International Conference on Robotics and Automation*, pages 1810–1817, 2002.
12. J. Ferruz and A. Ollero. Real-time feature matching in image sequences for non-structured environments. applications to vehicle guidance. *Journal of Intelligent and Robotic Systems*, 28:85–123, 2000.
13. P.-E. Forssén and A. Moe. View matching with blob features. In *2nd Canadian Conference on Robot Vision*, pages 228–235, Victoria, BC, Canada, May 2005. IEEE Computer Society.
14. Per-Erik Forssén. *Low and Medium Level Vision using Channel Representations*. PhD thesis, Linköping University, 2004. Thesis No. 858.

15. Robert Grabowski, L.E. Navarro-Serment, C.J.J. Paredis, and P.K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.
16. S. Grime and H. F. Durrant-Whyte. Data fusion in decentralized sensor networks. *Control Engineering Practice*, 2(5):849–863, Oct. 1994.
17. B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. In *Proceedings of the International Conference on Robotics and Automation*, pages 1521–1526, September 2003.
18. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.
19. A. Howard, L.E. Parker, and G.S. Sukhatme. Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection. *The International Journal of Robotics Research*, 25(5-6):431–447, 2006.
20. E. Hygounenc, I-K. Jung, P. Soueres, and S. Lacroix. The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping. *The International Journal of Robotics Research*, 23(4-5):473–511, 2004.
21. S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, 1997.
22. S.J. Julier and J.K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, volume 4, pages 2369–2373, Jun. 1997.
23. Tony Lindeberg. *Scale-space Theory in Computer Vision*. Kluwer Academic Publishers, 1994. ISBN 0792394186.
24. M.J. Matarič. Minimizing complexity in controlling a mobile robot population. In *Proceedings of the International Conference on Robotics and Automation*, pages 830–835, 1992.
25. L. Merino, F. Caballero, J. R. Martinez, J. Ferruz, and A. Ollero. A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. *Journal of Field Robotics*, 23(3):165–184, 2006.
26. L. Merino, J. Wiklund, F. Caballero, A. Moe, J.R. Martínez de Dios, P.-E. Forssén, K. Nordberg, and A. Ollero. Vision-based multi-UAV position estimation. *IEEE Robotics and Automation Magazine*, 13(3):53–62, 2006.
27. H. Moravec. Certainty grids for sensor fusion in mobile robots. *Sensor Devices and Systems for Robotics*, pages 243–276, 1989. Also CMU Robotics Institute 1987 Annual Research Review, 1988, pp. 33-48. Also in AI Magazine v9(2), Summer 1988, pp 61-77.
28. E. Nettleton, P.W. Gibbens, and H. Durrant-Whyte. Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem. In *Sensor fusion: Architectures, algorithms, and applications IV*, pages 428–437, 2000.
29. Stepán Obdrzálek and Jirí Matas. Object recognition using local affine frames on distinguished regions. In *13th BMVC*, pages 113–122, September 2002.
30. A. Ollero, J. Ferruz, F. Caballero, S. Hurtado, and L. Merino. Motion compensation and object detection for autonomous helicopter visual navigation in the comets system. In *Proceedings of the International Conference on Robotics and Automation, ICRA*, pages 19–24. IEEE, 2004.
31. A. Ollero and L. Merino. Control and perception techniques for aerial robotics. *Annual Reviews in Control*, (28):167–178, 2004. Elsevier (Francia).

32. L.-L. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte. Decentralised data fusion with particles. Australasian Conference on Robotics and Automation, 2005.
33. M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proc. Conf. Uncertainty in Artificial Intelligence*, 2003.
34. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.
35. M. Saptharishi, C.S. Oliver, C.P. Diehl, K. Bhat, J. Dolan, A. Trebi-Ollennu, and P. Khosla. Distributed surveillance and reconnaissance using multiple autonomous ATVs: CyberScout. *IEEE Transactions on Robotics and Automation*, 18(5):826 – 836, Oct. 2002.
36. T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig. Cooperative probabilistic state estimation for vision-based autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 18:670–684, October 2002.
37. B. Schumitsch, S. Thrun, G. Bradski, and K. Olukotun. The information-form data association filter. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, 2005. MIT Press.
38. R. Simmons, D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000.
39. K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 254–259, 1993.
40. R.D. Smallwood and E.J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.
41. P. Stepan, M. Kulich, and L. Preucil. Robust data fusion with occupancy grid. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 35(1):106–115, February 2005.
42. L. D. Stone, T. L. Corwin, and C. A. Barlow. *Bayesian Multiple Target Tracking*. Artech House, Inc., Norwood, MA, USA, 1999.
43. A. Stroupe, M.C. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the International Conference on Robotics and Automation*, May 2001.
44. A.W. Stroupe, R. Ravichandran, and T. Balch. Value-based action selection for exploration and dynamic target observation with robot teams. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 4190–4197, 2004.
45. S. Sukkarieh, E. Nettleton, J.-H. Kim, M. Ridley, A. Goktogan, and H. Durrant-Whyte. The ANSER Project: Data Fusion Across Multiple Uninhabited Air Vehicles. *The International Journal of Robotics Research*, 22(7-8):505–539, 2003.
46. S. Thrun, W Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.
47. S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.
48. C. Tomasi. *Shape and motion from image streams: a factorization method*. PhD thesis, Carnegie Mellon University, 1991.
49. S. Utete and H.F. Durrant-Whyte. Reliability in decentralised data fusion networks. In *Proc. of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 215–221, Oct. 1994.

50. R. Vidal, S. Sastry, J. Kim, O. Shakernia, and D. Shim. The berkeley aerial robot project (bear). In *Proceeding of the International Conference on Intelligent Robots and Systems, IROS*, pages 1–10. IEEE/RSJ, 2002.
51. R. Vidal, O. Shakernia, H. J. Kim, D.H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, Oct. 2002.
52. T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel. CS Freiburg: coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, 18(5):685–699, October 2002.
53. Y. Yanli, A.A. Minai, and M.M Polycarpou. Evidential map-building approaches for multi-UAV cooperative search. In *Proc. of the American Control Conference*, pages 116–121, Jun. 2005.
54. Z. Zhang. Parameters estimation techniques. a tutorial with application to conic fitting. Technical report, INRIA, France, October 1995.
55. Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.