Multi-robot Surveillance Through a Distributed Sensor Network

Andrea Pennisi, Fabio Previtali, Cristiano Gennari, Domenico D. Bloisi, Luca Iocchi, Francesco Ficarola, Andrea Vitaletti and Daniele Nardi

Abstract Automatic surveillance of public areas, such as airports, train stations, and shopping malls, requires the capacity of detecting and recognizing possible abnormal situations in populated environments. In this book chapter, an architecture for intelligent surveillance in indoor public spaces, based on an integration of *interactive* and *non-interactive* heterogeneous sensors, is described. As a difference with respect to traditional, passive and pure vision-based systems, the proposed approach relies on a distributed sensor network combining RFID tags, multiple mobile robots, and fixed RGBD cameras. The presence and the position of people in the scene is detected by suitably combining data coming from the sensor nodes, including those mounted on board of the mobile robots that are in charge of patrolling the environment. The robots can adapt their behavior according to the current situation, on the basis of a

A. Pennisi (✉) · F. Previtali · C. Gennari · D. D. Bloisi · L. Iocchi · F. Ficarola · A. Vitaletti · D. Nardi
Department of Computer, Control, and Management Engineering, Sapienza University of Rome, via Ariosto 25, 00185 Rome, Italy
e-mail: Pennisi@dis.uniroma1.it

F. Previtali
e-mail: Previtali@dis.uniroma1.it

C. Gennari
e-mail: Gennari@dis.uniroma1.it

D. D. Bloisi
e-mail: Bloisi@dis.uniroma1.it

L. Iocchi
e-mail: Iocchi@dis.uniroma1.it

F. Ficarola
e-mail: Ficarola@dis.uniroma1.it

A. Vitaletti
e-mail: Vitaletti@dis.uniroma1.it

D. Nardi
e-mail: Nardi@dis.uniroma1.it

A. Koubâa and J.R. Martínez-de Dios (eds.), *Cooperative Robots and Sensor Networks 2015*, Studies in Computational Intelligence 604,
DOI 10.1007/978-3-319-18299-5_4

Prey-Predator scheme, and can coordinate their actions to fulfill the required tasks. Experimental results have been carried out both on real and on simulated data to show the effectiveness of the proposed approach.

Keywords Mobile robots · Wireless sensor networks · Multi-robot systems · Multi-robot surveillance

1 Introduction

A critical infrastructure (CI) is a system which is essential for the maintenance of vital societal functions. Public areas, such as airports, train stations, shopping malls, and offices, are examples of CIs that can be a target for terrorist attacks, criminal activities or malicious behaviors. Usually, CIs are monitored by passive cameras with the aim of detecting, tracking, and recognizing objects of interest to understand and prevent possible threats.

However, traditional vision-based systems can result ineffective when dealing with realistic scenarios, since their passive sensors can fail in identifying and tracking an object of interest in a large environment, due to partial and total occlusions, changes in illumination conditions, and difficulties in re-identifying objects in different non-overlapping views. Moreover, a network of fixed passive sensors can be subject to malicious physical attacks [11].

1.1 Contributions of the Book Chapter

In this chapter, the problem of monitoring a populated indoor environment is faced by combining data coming from multiple heterogeneous fixed and mobile sensors. The term "populated" is used through the book chapter to denote an environment with presence of people. In our description, we do not take into account crowded or densely-populated environments. We describe the development of an architecture designed for the surveillance of a large scenario, where authorized personnel wear Radio Frequency Identification (RFID) tags and the environment is monitored by fixed RGBD cameras with RFID receivers and it is patrolled by multiple mobile robots, equipped with laser range finders and RFID receivers (see Fig. 1). Laser scans, RFID tag data, and RGBD images gathered by the distributed sensors are merged to obtain information about the position and the identity of people in the scene. Moreover, the robots coordinate their actions through a dynamic task assignment to fully cover the operational environment.

The architecture is conceived to work in a fully distributed fashion and to automatically raise alarms (possibly communicated to a central operational station) when abnormal conditions are detected. To this end, the developed architecture integrates different technologies. Although all the above technologies have been already

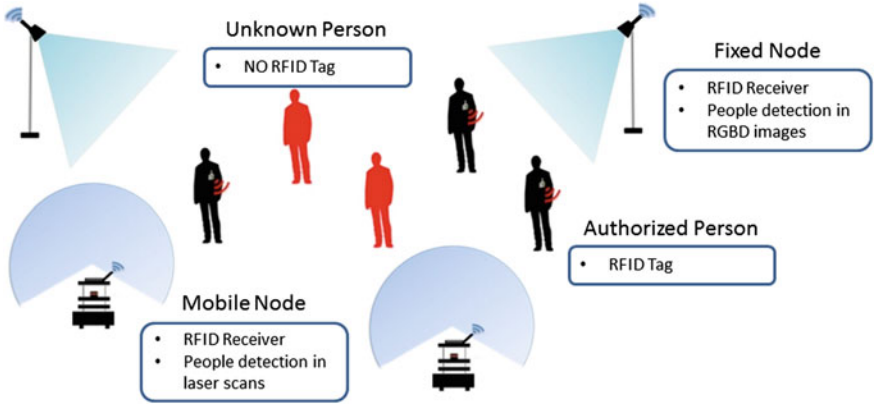

Fig. 1 The proposed architecture, combining mobile robots, fixed RGBD cameras, and RFID tags and receivers to monitor a populated environment

developed in previous works, their integration was not previously considered, in particular for surveillance applications. Moreover, an experimental analysis, carried out also in a real environment, shows the effectiveness of the implemented system.

The remainder of the chapter is organized as follows. Related work is analyzed in Sect. 2, while the definition of the problem is given in Sect. 3. The components of the architecture are described in Sect. 4 and the process of fusing the information coming from the different sensors is described in Sect. 5. The multi-robot coordination and the task assignment processes are detailed in Sect. 6. Results on both a real and a simulated environment are discussed in Sect. 7. Conclusions and future directions are drawn in Sect. 8.

2 Related Work

There exists a large literature about the problem of people detection in indoor environments by using fixed cameras. However, since a variety of factors, including illumination conditions, occlusions, and blind spots, limit the capacity of pure vision-based systems, it is possible to consider a combination of multiple heterogeneous sensors to achieve better results.

Approaches integrating multiple sensors can be divided into two main categories: (1) *interactive methods*, where each person has an active role during the detection process (e.g., by dressing an RFID tag) and (2) *non-interactive methods*, where the role of the person is passive and the analysis is carried out by the detection system only (e.g., a camera). In the rest of this section, some examples of interactive and non-interactive methods are described.

2.1 Interactive Methods

One of the first experiments about collecting information from a group of people in a physical real context is described by Hui et al. [8]: 54 individuals attending to a conference, dressed with an Intel iMote device consisting of a micro-controller unit (MCU), a Bluetooth radio and a flash memory, are considered. However, the choice of using Bluetooth does not allow for a fine-grained recording of social interactions, mainly because of the missing possibility of analyzing face-to-face interactions.

Multiple projects focusing on collecting data from social interactions are developed by the *SocioPatterns* collaboration. Partners participating in this collaboration have been the first to record fine-grained contacts by using active RFID sensors. This kind of devices allows to record face-to-face interactions within a range of 1.5 m. For example, Becchetti et al. [2] describe an experiment in which data coming from wireless active RFID tags worn by 120 volunteers moving and interacting in an indoor area are collected. The tags periodically broadcasts information about contacts with similar tags (i.e., whenever the person wearing the tag came close to another member of the volunteer group). Assuming that the subjects wear the tags on their chest and using very low radio power levels, contacts between tags are detected only when participants actually face one another, since the body effectively acts as a shield for the sensing signals. Thus, it is reasonable to assume that the experiment can detect an ongoing social contact (e.g., a conversation). SocioPatterns has made several installations in different social contexts, including conferences [1], hospitals [9], primary schools [16], and a science gallery [4], making some data sets publicly available on its website.[1]

Experiments similar to the SocioPatterns' ones have been conducted deployed by Chin et al. [5], consisting in monitoring people wearing active RFID badges during a conference. The goal is to build a system that can find and connect people to each other. A remarkable result of the experiment is that, for social selection, more proximity interactions lead to an increased probability for a person to add another as a social connection.

While the above approaches target the analysis of social human behaviors, in this book chapter we investigate the use of data acquired from interactive tags for surveillance applications. Indeed, we aim at integrating the SocioPatterns sensing platform together with other sensing technologies, including laser range finders and RGBD cameras, to overcome the problems related to traditional automatic surveillance. It is worth noticing that a scenario in which (1) authorized personnel wear RFID tags and (2) other authorized actors (e.g., visitors, travelers, spectators) may have an RFID tag as well (e.g., included in a ticket or a passport or a boarding pass) is a quite plausible one. Airports, embassies, and theaters are examples of scenarios where interactive methods can be used.

[1] http://www.sociopatterns.org/datasets.

2.2 Non-interactive Methods

Approaches in this category are based on passive sensors. Since the literature on vision-based systems is huge, we limit our description to existing approaches using technologies other than vision for addressing automatic surveillance. In the field of laser-based systems, Cui et al. [6] introduce a feature extraction method based on accumulated distribution of successive laser frames. A pattern of rhythmic swing legs is used to extract each leg of a person and a region coherency property is exploited to generate an efficient measurement likelihood model. A Kalman and a Rao-Blackwellized Monte Carlo data association (RBMC-DAF) filters are combined to track people. However, this approach is not effective for people moving quickly or partially occluded.

Xavier et al. [17] describe a feature detection system for real-time identification of lines, circles, and legs from laser data. Lines are detected by using a recursive line fitting method, while leg detection is carried out by taking into account geometrical constrains. This approach cannot handle scan data of a dynamic scene including moving people or not well separated structures.

A solution involving human-robot interaction is presented by Shao et al. [14]. Visual and laser range information are combined: Legs are extracted from laser scans and, at the same time, faces are detected from the images of a camera. A mobile robot uses the detection procedure (that returns the direction and the distance of surrounding people) to approach and to start interacting with humans. However, the swinging frequency is too low for people detection and tracking.

In the above cited papers, the main limitation concerns the problem of detecting multiple people. In most cases, the approaches can deal with well separated objects, but cannot be easily extended when multiple people are grouped together. In this book chapter, we propose an approach that can be used in a populated environment and that is suitable for monitoring groups of people. The method combines interactive and non-interactive heterogeneous sensors in order to overcome the problems of traditional vision-based systems.

Information coming from range finders, RFID receivers, and RGBD cameras are merged to obtain the position and the identity of people in the scene. Moreover, the actions of the robots are coordinated according to a dynamic task assignment algorithm, in order to a have a dynamic monitoring range.

3 Problem Definition

The problem of monitoring a populated environment can be modeled as a *Prey-Predator* game. Indeed, considering the sensor nodes as predators and the objects to be monitored as preys, it is possible to formalize the surveillance task as follows: *A predator tries to catch preys and a prey runs away from predators*.

The game consists of preys and predators living in the same environment. It is usually defined as a game where both predators and preys have a score and any individual can gain or lose points over time. A metric distance is assigned to each prey and to each predator as the game score. The goal for each prey is to maximize its distance from the predators, while each predator aims at minimizing its distance from the preys. In our setting, the preys are the people moving in the monitored environment, while the predators are the sensor nodes that are used for detecting the presence and for estimating the position of a person. A sensor node is made of an RFID reader and other additional sensors, like an RGBD camera or a laser range finder. Moreover, some sensor nodes are mounted on mobile robots that navigate in the environment. For such a reason and for the presence of blind areas also, the portion of the environment that is currently observable can vary over time.

The monitoring task consists of identifying every person that does not wear an RFID tag by assigning her/him an *identity number* (ID). The goal of the monitoring task is achieved whenever a sensor node can detect the presence and the position of a person, determining if such a person is wearing or not an RFID tag. Formulating the surveillance task as a Prey-Predator game provides the following advantages: (1) In the case of a person leaving the monitored area and then re-entering later, the re-identification problem is not an issue, since if a person was labeled with an ID i before exiting the scene, when she/he re-enters the scene the system can use another ID j ($i \neq j$) and continue its process of determining if j is wearing or not an RFID tag; (2) The same performance metric defined for the Prey-Predator game can be used for evaluating our approach, providing quantitative results (see the experiments reported in Sect. 7).

4 Sensor Nodes

The proposed monitoring approach uses a combination of multiple heterogeneous fixed and mobile sensor nodes. Authorized people wear RFID tags of the type shown in Fig. 2a. Mobile nodes are robots equipped with a laser range finder and an RFID receiver (Fig. 2b), while fixed nodes are made of RGBD cameras to grab visual 3D information and RFID receivers that are mounted near the camera (Fig. 2c). The communication between mobile and fixed sensors is achieved by using TCP/IP over a wireless network. This is a feasible solution, since the size of exchanged messages among nodes is quite small (up to 1 kb) and the possibility to either lost a message or receive a delayed message is negligible.

4.1 RFID Tags and Receivers

The two main entities of our sensing platform, designed and developed by the SocioPatterns research collaboration, are the OpenBeacon active RFID tags (Fig. 2a)

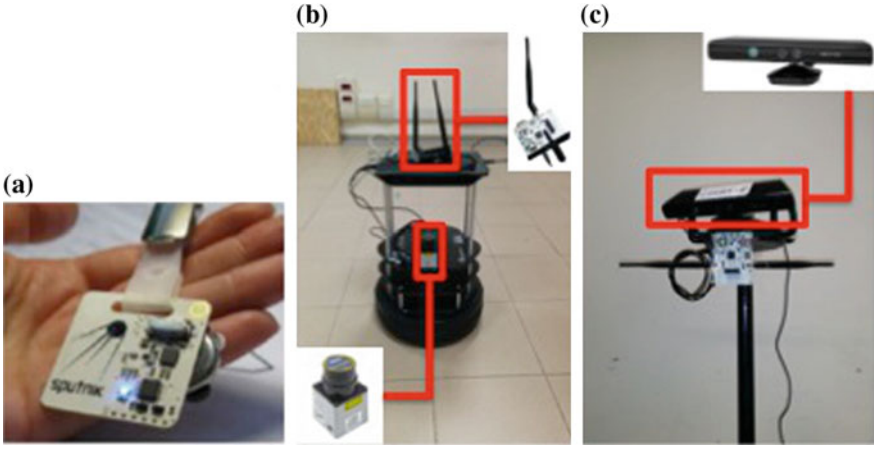

Fig. 2 **a** RFID tag. **b** Turtlebot robot equipped with a laser range finder and an RFID receiver. **c** Fixed RGBD camera with RFID receiver

and the OpenBeacon Ethernet reader (top right in Fig. 2b). The tags are electronic wireless badges equipped with a PIC16 micro-controller (MCU) and an ultra low power radio frequency transceiver. The MCU has a total SRAM of 256 bytes and can work up to 8 MHz of frequency, while the transceiver has very low energy consumptions: 11.3 mAh in transmission at 0 dBm of output power and 12.3 mAh in reception at 2 Mbps of air data rate. They are powered by batteries ensuring a lifetime of more than two weeks and are programmed to periodically broadcast beacons of 32 bytes at four different levels of signal strength: 0, −6, −12, −18 dBm. Every beacon contains the tag identifier, the information about the current signal strength, and other fields useful for debugging. Similarly, the RFID reader has a transceiver as well and an omni-directional covering range of 10 m.

The whole sensing platform is designed to allow the RFID receivers to collect the data sent by each tag via the wireless channel. In our scenario, a receiver is mounted on each robot and it is used to read the signal strength and the ID of a tag, in order to establish if a person detected in the environment is actually wearing a tag. All data collected by RFID readers are forwarded to a central logging server,[2] that stores all messages in log-files. Each record contains information about the tag whom sent the packet, including its ID, the signal strength, the sequence number and the IP of the reader that collected the corresponding message.

[2]OpenBeacon Logger. https://github.com/francesco-ficarola/OpenBeaconLogger.

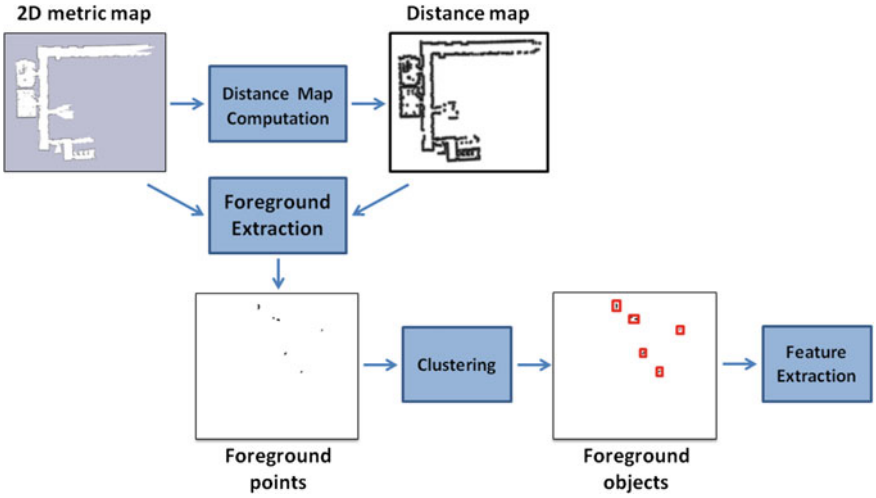

Fig. 3 People detection using the laser range finder

4.2 Laser Range Finders

The mobile sensor node is composed of a Turtlebot[3] equipped with a range finder and an RFID receiver (Fig. 2b). Multiple robots are involved in the task of patrolling the environment. Each robot has a 2D metric map of the environment, that is built off-line using the ROS gmapping tool.[4] Furthermore, each robot can be considered always well-localized on the 2D metric map by using the ROS implementation of the AMCL localization method.[5] Person detection is carried out by means of a *distance map*, indicating the probability that a given point in the current laser scan belongs to the metric map. By comparing the distance map with the metric map it is possible to extract the foreground objects, i.e., sets of points in the distance map that are far enough from the metric map points. From each foreground object the following features are extracted: the number of its points, their standard deviation, a bounding box, and the radius of the minimum enclosing circle (see Fig. 3). Then, the features are sent as input to an Ada-Boost based person classifier, trained with about 1800 scans.

People tracking relies on the particle filter algorithm called *PTracker*, that is described in Sect. 5. Data association is used to determine the relationship between observations and tracks, and multiple hypotheses are maintained when observations may be associated to more than one track. Finally, each track is combined with the signal detected by the RFID receiver mounted on each robot, in order to verify if a person is wearing the RFID tag.

[3] http://www.turtlebot.com/.

[4] http://wiki.ros.org/gmapping.

[5] http://wiki.ros.org/amcl.

4.3 RGBD Cameras

The Microsoft Kinect (version 1.0) has been used as RGBD camera. Kinect sensor supplies an RGB image with a resolution of 640×480 and a frame rate of 30 frames per second. 3D information are received in the form of a 11-bit depth image. Both color and depth information are used for computing an accurate foreground detection. RGB and depth data are stored for each captured frame.

A statistical approach, called Independent Multimodal Background Subtraction (IMBS) [3], is used to create the background model, that is updated every 15 seconds for dealing with illumination changes. The obtained foreground mask is used as starting point for a 3D clustering step.

Let $\mathfrak{B}$ denote the set of 3D points that corresponds to the 2D points belonging to the blobs in the foreground mask and $\mathfrak{C}$ denote all the 3D points of the point cloud generated from the depth data provided by the Kinect ($\mathfrak{B} \subset \mathfrak{C}$). In order to improve the detection results, all the 3D points $\in \{\mathfrak{C} \setminus \mathfrak{B}\}$ having a distance < 0.01 m from the points in $\mathfrak{B}$ are recursively added to $\mathfrak{B}$ itself. Then, to filter out possible false positives, all the blobs with a maximum height < 1.2 m are discarded, while the others are considered as valid observations. Finally, the 3D positions of the valid blobs are computed by estimating their Euclidean distance from the cameras. Indeed, since the positions of the cameras monitoring the environment are known, people can be localized on the 2D metric map by averaging the 3D points belonging to the their blobs and calculating the distance of the average point from the camera. The above described steps are summarized in Fig. 4.

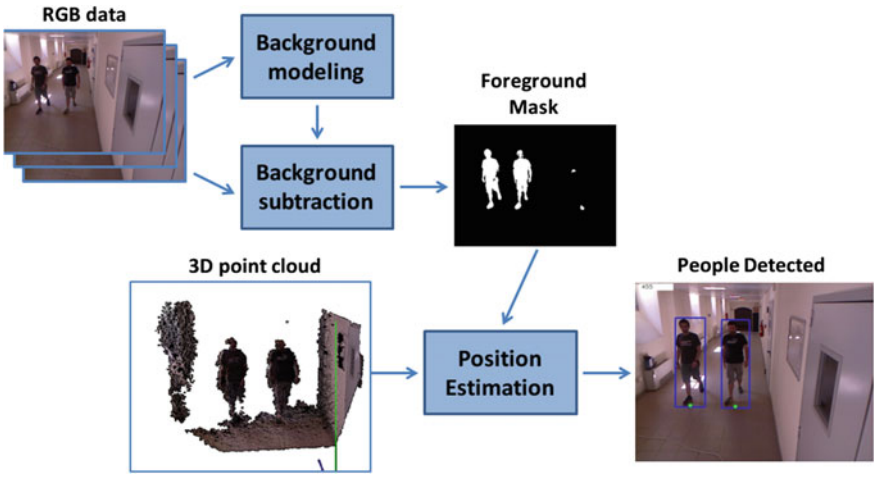

Fig. 4 People detection using a RGBD camera

5 Data Fusion

Information coming from fixed and mobile sensor nodes needs to be merged. The data fusion process is made of two phases: (1) Obtaining tracks by fusing visual and laser data and (2) Merging the tracks with RFID receiver information.

In the first phase, a multi-object particle filter approach, called PTracker, has been used in order to fuse and track the observations extracted from RGBD and laser data. A particle filter-based tracker maintains a probability distribution over the state of the object being tracked, keeping information about position, scale, color, direction and velocity of the object. Particle filters represent this distribution as a set of weighted samples (particles). Each particle represents a possible instantiation of the state of the object and it is a guess representing one possible position of the object being tracked. The set of particles contains more weight at locations where the object being tracked is more likely to be. This weighted distribution is propagated through time the Bayesian filtering equations, and the trajectory of the tracked object is determined by taking the particle with the highest weight or the weighted mean of the particle set at each time step. A detailed description of the data fusion method is available at http://www.dis.uniroma1.it/~previtali/downloads/DataFusion.pdf. The output of this phase is a set $S_t = \{o_t^1, \ldots, o_t^n\}$ containing all the observations o_t^i at time t, $1 \leq i \leq n$, where n is the total number of the observations.

In the second phase, S_t is merged with the information coming from the RFID receivers at time t, the set $U_t = \{id_t^1, \ldots, id_t^k\}$, where id_t^k is a triple $\langle t,\ p,\ r \rangle$, with t being the identification number of the tag, p the pose of the receiver that detects the tag t, and r the detection range of the receiver. An observation $o_t^i \in S_t$ is associated to a triple $(id_j^t = \langle t_j, p_j, r_j \rangle) \in U_t$ if all the particles of o_i^t (computed by PTracker) are included in the circular range having radius r_j and center p_j. After the merging phase, three different outputs can be generated:

1. $z_h^t = \langle o_i^t, id_j^t \rangle$ where o_i^t has been merged with id_j^t;
2. $z_h^t = \langle o_i^t, ? \rangle$ where no RFID data have been associated with the track o_i^t;
3. $z_h^t = \langle ?, id_j^t \rangle$ where no track information can be assigned to the detected RFID data id_j^t.

6 Multi-robot Surveillance

In our architecture, multiple robots are responsible for patrolling the environment, meaning that a team of different robotic agents have to be coordinated and controlled. This can be done by adopting different coordination strategies and control approaches, in order to plan the robots' behavior. We propose a novel approach based on a distributed coordination and a hybrid control, that makes use of a variant of the multi-robot Petri Net Plans [18].

6.1 Distributed Coordination

Coordination strategies for a team of robots can be divided into two categories: (1) Centralized methods, where an agent sends commands to other robots, and (2) Distributed approaches, where each agent decides its task and it shares information with the team.

In centralized solutions, the whole planning task is assigned to a single agent (central unit) that is responsible to calculate the next task for each robot. This must be done in a very short time, due to real-time constraints. Furthermore, the central unit represents a weak point, since in case it crashes, this will affect the functionality of the entire system.

Adopting a distributed approach, it is possible to deal with the above issues. The idea is to make each agent acting independently from each other, by using only local knowledge about the environment. An agent can collaborate with its neighbors in order to divide the task into sub-problems or to work together to achieve the defined goals.

Even if a distributed coordination increases the computation costs, due to the need of managing the necessary coordination messages, it allows to split the costs among all the team. Indeed, each robot performs a smaller amount of computation with respect to the whole computation load required to a single central node. The communication between agents in a distributed processing is greatly reduced as well, since it is not necessary to exchange information about perception, thus reducing the transmissions to a simple exchange of lightweight coordination messages. In addition, the reaction to external events is faster, since events are managed locally, without the need of waiting for instructions from a central global coordinator. Finally, a distributed execution is more robust to failures. Indeed, if an agent becomes unreachable, it can be replaced by another one without affecting the capability of the system.

6.2 Petri Nets Plans

We propose a distributed architecture with hybrid control and centralized planning. Using a centralized plan means that a plan is generated from a supervisor (an agent or a user) and it is sent in a distributed fashion to all the robots. Petri Nets Plans (PNP) [18] is a framework based on Petri Nets, conceived for designing, writing, executing, and debugging plans. The use of PNP allows a clear distinction between action specification and their implementation as well as a formal specification of plans, which permit to implement reasoning and to verify procedures.

Ziparo et al. [18] have extended the PNP framework to multi-robot systems. The extension uses a shared plan, that provides each agent with a plan created in a centralized manner, and with the model to run it in a distributed fashion. This approach allows to execute a set of PNPs, created by a shared multi-robot plan, for a single-robot, without the need of a central coordinator. The correctness of the distributed

execution with respect to the multi-robot PNP is enforced by using the communication primitives *send*(ID), *receive*(ID) and *sync*(ID,ID′), where ID and ID′ are unique identifiers for the state of execution of single-robot plans. The primitives are modeled as single-robot ordinary non-instantaneous actions and represent communication acts and they are used to define three operators for coordinating the plans for each single robot:

- **Hard Synchronization.** It synchronizes in time two single-robot plans and it allows for information sharing among them, through the communication of ID_s and ID_r which encode the state of execution for the plan of agent s and agent r, respectively.
- **Soft Synchronization.** It defines a precedence relation among two actions of two different robots.
- **Multi-robot Interrupt.** It allows for relating interrupts between the actions of two robots s and r. Since each robot has a receiving thread, when a sensing action launches an interrupt on s, it send a message to r, which starts an interrupt as well.

6.3 Hybrid Control

A robot decides the type of action to perform as the result of the perceived information from the environment. It contains two procedures, a reactive and a deliberative one, with an interface that is responsible for connecting them. The reactive procedure is used for handling situations that require an immediate reaction of the robot, such as avoiding obstacles or sending an alarm in case of intruders. The deliberative procedure is used for long-term decisions, like planning a trajectory. This two-level architecture has the advantage of increasing the reactivity of the system, having a dynamic and very responsive control. It is worth noting that, designing the interface between the two layers is not trivial.

The PNP formalism is used to manage both the deliberative and reactive control. In [18] the global plan is converted into individual single-robot plans. In such a way, a manual or an automatic rewriting of the single plans is needed. However, the individual plans must be distributed to each robot, making the operations of changing and debugging a plan rather complex.

We propose a different approach. Each agent is equipped with an execution model that can interpret and execute the original PNP. Then, the plan interpreter manages the various actions by implementing those addressed to it and by handling the actions which pertain to different agents, through communication primitives. The main difference with [18] lies in the PNP executor. Indeed, in [18] each received command is interpreted by using operators and primitives both defined in PNP, then the appropriate commands are sent to the robot in order to complete the required tasks (see Fig. 5a).

Our approach includes, instead, two new software modules: (1) the Coordinator and (2) the Robot ID Recognizer. The Coordinator is not a specific robot, but a

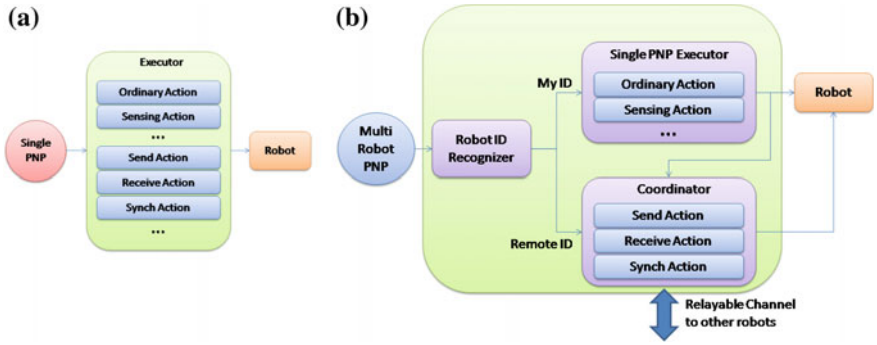

Fig. 5 **a** Architecture for the PNP Executor proposed in [18]. **b** Our modified architecture for the PNP Executor

software procedure running on all the robots in the team. Both modules are designed to work in a distributed asynchronous fashion. We assume that all the information are sent and received asynchronously between the robots and that some robots can be unable to receive all the available information due to lost packets.

The PNP executor is the same as the one in the single-robot case, which uses only the PNP primitives and operators. In the PNP plan, by following the rules of labeling, an action is preceded by the ID of the robot that has to perform it. First of all, the action is passed to the Robot ID Recognizer, which controls whether the action has to be executed by the local agent or by one of the remote robots. If it is the case for the local agent, the action is passed to the single PNP executor. Instead, if the action is for a remote robot, it is passed to the local coordinator module which transforms the action into a communication primitive, in order to synchronize the evolution of the local plan with the execution state of the remote robots. This can be done since the coordinator informs the other robots about the ending of the local action (see Fig. 5b), thus allowing to synchronize the plans of all the agents. A copy of the multi-robot plan is stored by all robots, and all the PNP executors are synchronized. In such a way, if no communication errors occur, each agent exactly knows the execution state of all the remote agents.

6.4 Implementation

For the low-level control, we use the Robotic Operative System (ROS),[6] which is a flexible framework for robotic infrastructures equipped with a collection of tools, libraries, and conventions aiming at simplifying the creation and management of robotic platforms. The tools are arranged in nodes, that can be integrated with other nodes to compose a complex architecture.

[6] http://www.ros.org.

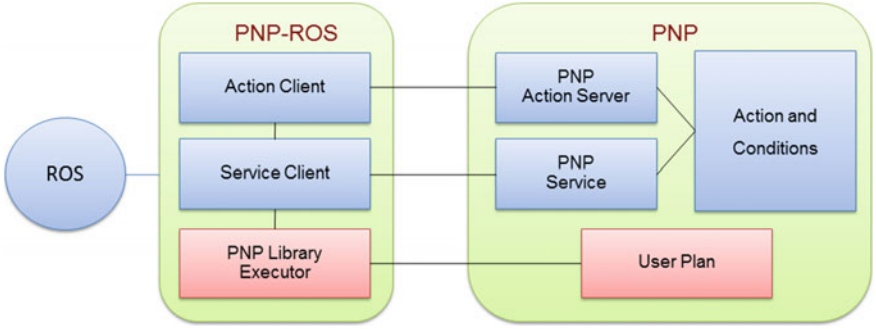

Fig. 6 Scheme of the PNP-ROS bridge

In order to interact with ROS, a node has been built for interfacing the external libraries. The scheme of the bridge from PNP to ROS is reported in Fig. 6. The Action Client asks the Action Server to execute a given action, while the Service Client asks the PNP Service to evaluate the firing conditions. The PNP Service is user-defined and it maintains the state of the system giving a response (i.e., a value *true* or *false*) about a condition. When PNP-ROS sends a request for an action, the Robot ID Recognizer (see Fig. 5b) pre-processes the request by checking if the action has to be sent to the local agent (and therefore it will be performed), or if the action has to be sent to a remote robot. In the last case:

- If the request is for a local action, the Action Server launches a new communication thread involving the other ROS modules in order to accomplish the task. When the action is finished, the boolean state variable is set to *true* and the PNP Service can respond with a positive value. This means that the PNP library Executor knows that the action is terminated and it can send an "ActionFinished" message to all the remote agents and proceed with another plan.
- If the address of the action is a remote agent *id*, the Action Server launches a new communication thread with a primitive *receive(id)*. This is a blocking function and thus the thread stops its execution waiting for a message. When the remote agent finishes its action, it sends an "ActionFinished" message and the thread can resume.

Using the above described protocol, each agent can directly use the multi-robot original plan, while maintaining the information about the state of execution of the plan for the other agents.

6.5 Example of PNP Execution

Figure 7 shows an example of a simple PNP execution. Two sensor nodes are monitoring the environment: Node0 is a fixed camera with an RFID receiver, while Robot1 is a mobile robot equipped with a laser range finder and an RFID receiver. Both nodes

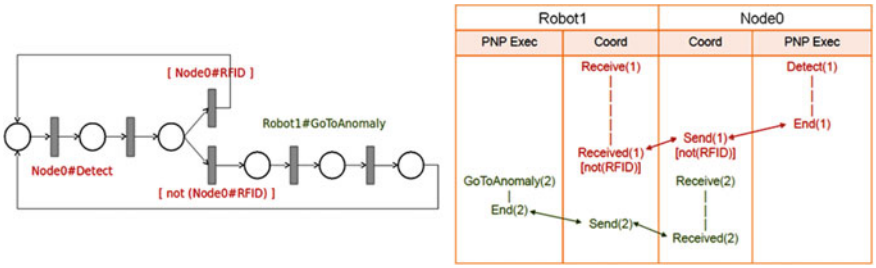

Fig. 7 Example of PNP execution

receive a request for the action "Node0#Detect" meaning that Node0 is required to check if a person is in its field of view. Robot1 understands that the action is for a remote agent, thus it sends the request to the Coordinator, that transforms the action in a primitive *receive*. When Node0 detects the presence of a person, it completes its action and sends a message, containing the information about the presence of an authorized person or not, to Robot1. In the example shown in Fig. 7, the person does not wear an RFID tag, so Robot1 is notified that it has to check the possible abnormal situation (action "Robot1#GoToAnomaly").

6.6 Dynamic Task Assignment

The robots in the team must work together on the current task, coordinating their actions and efficiently sharing the workload to maximize the overall task performance. This is a complex goal, since the robots operate in a dynamic environment and the perceptions can be noisy.

In order to deal with the coordination problem in a real scenario, we adopted a solution based on a greedy algorithm [12, 15] and the Prey-Predator game formalization (see Sect. 3). A Dynamic Task Assignment (DTA) process is responsible for assigning a prey to a predator. Such an assignment is unique, meaning that a predator cannot chase two or more preys. A predator creates a new *bid* each time it sees a prey (Fig. 8a). A bid describes its estimates of the expected information gain and costs of traveling to various locations for catching the prey. Bids, that are the same for both mobile and fixed sensors, are asynchronously sent to all the predators (Fig. 8b) and the DTA algorithm makes the assignment on the basis of the current bids (Fig. 8c). The tracking performance of a predator increases at the decrease of its distance from a prey, thanks to the higher quality of the received sensor data. During the chasing, a predator could change the prey to chase: To handle this situation, the DTA algorithm assigns the prey no longer chased to another predator.

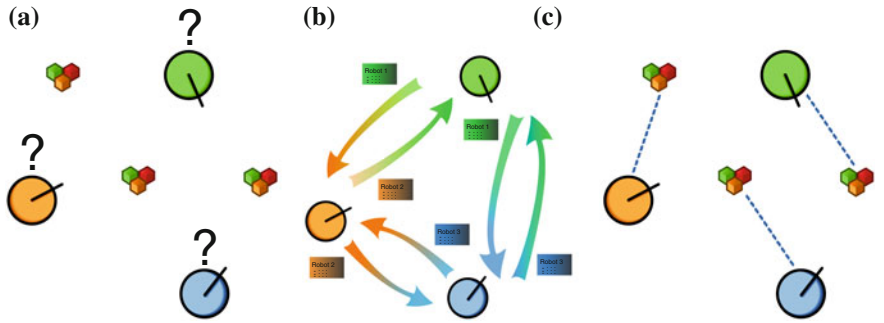

Fig. 8 Dynamic task assignment (DTA). **a** Predators do not yet know which prey chase. **b** Predators exchange their bids. **c** The DTA algorithm assigns at each predator the best prey to chase

7 Experimental Evaluation

Experimental results has been computed both in a real scenario and by using a simulator. The experiments carried out in the real scenario have been used to generate the error models for the sensors in the network nodes (a model for the RGBD camera and one for the laser range finder). The models are very useful for obtaining realistic results in the simulated environment, that are then used to quantitatively evaluate the effectiveness of the proposed architecture.

7.1 Experiments with Real Data

Experiments with real data have been performed with two purposes: (1) To generate the error models for the sensor nodes, and (2) To demonstrate the overall feasibility of the developed system.

The first set of experiments are thus focused on determining the error models of the sensors used for people detection: an RGBD camera (a Kinect sensor) and a laser range finder (a Hokuyo UTM-30LX). The setup is given by two fixed sensor nodes each including one of the two sensors and a person standing at a variable distance d from the sensor nodes (see Fig. 9). Four runs for each considered distance (ranging from 1 to 4 m) have been considered. The obtained results are reported in Table 1. As expected, the accuracy of the laser-based method is higher than the one of the RGBD-based technique, and the errors increase with the distance, for both the laser and the RGBD camera. The results allow to determine a suitable error model for the sensors involved in the architecture.

Moreover, when considering a mobile sensor (i.e., an RGBD camera or a laser mounted on a robot), the error in the self-localization routine carried out by the robot must be taken into account, since it can influence the detection accuracy. To this end, we performed a set of preliminary tests on different Turtlebot robots in

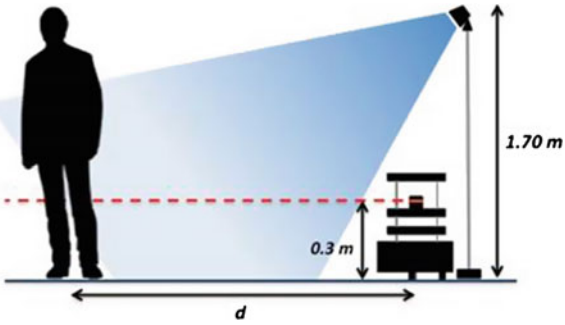

Fig. 9 Laser and visual data are merged using the floor as a common reference frame

Table 1 Results in the real scenario

Sensor type	Real distance (m)	Detected distance (m)	Error (m)
Kinect	1	1.441	0.441
Laser	1	1.029	0.029
Kinect	2	2.404	0.404
Laser	2	2.040	0.040
Kinect	3	3.464	0.464
Laser	3	3.068	0.068
Kinect	4	4.533	0.533
Laser	4	4.066	0.066

order to calculate their localization error. We used the well-known approach by Fox et al. described in [7], obtaining a localization error in the range between 8 cm and 16 cm. The computed error models (sensors + localization) are used as input for the simulated experiments (described below) to obtain realistic observations during the simulations.

The second set of experiments with real data has been performed to show the effectiveness of the entire approach. Here we do not collect quantitative measures, but just demonstrate the whole architecture running. Some videos showing the experiments are available in [13] and some snapshots are reported in Fig. 10. The behavior of the system is the following. Whenever the fixed sensor node detects a person, a Turtlebot equipped with a laser range finder and an RFID reader is sent in that location, in order to verify the status of the person (i.e., if she/he is wearing or not the RFID tag) and to report the anomaly if it is the case (see Fig. 11). Otherwise, if the system does not detect anomalies, sends a message to the robot which continues patrolling the environment.

Finally, we measured the computational speed of the entire system processing in terms of frames per second (FPS) on live data coming from the sensors, using an Intel Core i5-3210M 2.50 GHz (2 cores), 4 GB RAM and a virtual machine with a

Fig. 10 Real experiment: **a** fixed node composed by a Kinect and a RFID receiver, **b** two RFID tags, **c** a Turtlebot robot equipped with a laser range finder

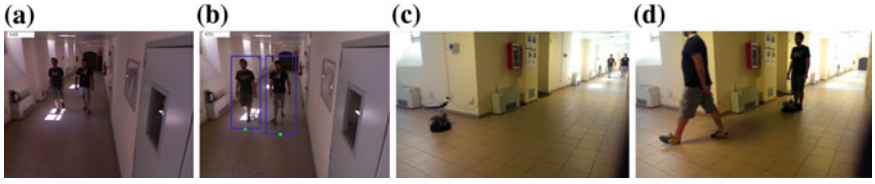

Fig. 11 Experiment with real data. **a** A person is wearing the RFID tag while the other one are not, **b** the fixed sensor detects the two people, identifies only one RFID tag and, **c** sends the coordinates to the mobile robot, **d** the robot stops in front of the person that is not wearing the RFID tag

Table 2 Computational speed for the RGBD detection module running with a single camera

Frame size	FPS (2 cores)	FPS (virtual machine, 2 cores)
320 × 240	25	20
640 × 480	23	18

simulated processor 2.00 GHz (2 cores), 4 GB RAM. The results are shown in Table 2, demonstrating that the proposed approach is suitable for real-time applications with commercial CPUs and even in the case of using a virtual machine.

7.2 Experiments in a Simulated Environment

The goal of the experimental evaluation on simulated data is to quantitatively evaluate the performance of our method. We run all the experiments by using the simulator Stage. In Stage, both the sensor nodes and the people are represented as robotic agents. The estimation of the position of the simulated people (i.e., the implementation of

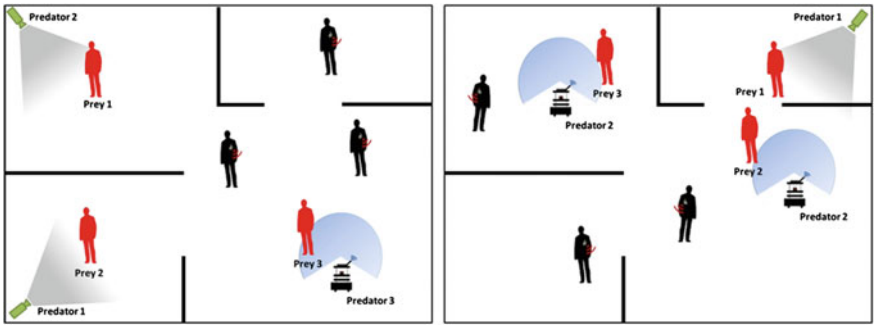

Fig. 12 The simulated environment in stage

the virtual sensors) is obtained by generating observations with the addition of an error calculated accordingly to the error model of the real sensors calculated in the experiments discussed above. Moreover, to have a realistic simulation, we adopted a realistic model of the people, the same model of the robots as well as the same field-of-view of the sensors as in the real scenario.

Figure 12 shows two screen-shots from an experiment in which three sensor nodes (i.e., predators) are chasing moving people without an RFID tag (i.e., preys). People with tags are no more chased once detected. The experiment has been carried out by launching multiple runs, changing every time the initial positions and the type of the sensors (fixed or mobile) and the starting positions of the people with and without tags. The average error has been calculated by using Eq. 1, while the standard deviation by using Eq. 2:

$$avg = \sum_{t=1}^{n} \sum_{i=1}^{k} \frac{1}{k} \frac{\sum_{j=1}^{m} \| e_{i,j}^{(t)} - g_j^{(t)} \|}{m} \tag{1}$$

$$std.\,dev. = \sum_{t=1}^{n} \sum_{i=1}^{k} \frac{1}{k} \frac{\sum_{j=1}^{m} \| e_{i,j}^{(t)} - avg_i^{(t)} \|}{m} \tag{2}$$

where $e_{i,j}^{(t)}$ is the jth estimation performed by the robot i at time t, $g_j^{(t)}$ is the ground-truth position provided by the simulator of the object j at time t, m is the number of estimations performed by the robot i at time t, n is the duration in seconds of the experiment and k is the number of robots (i.e., predators) involved in the experiment. The results obtained during the simulations are reported in Table 3: The low value of the standard deviation demonstrates a remarkable reliability of the proposed approach.

We also quantitatively measured the performance of the tracking module. To this end, we used the well- known CLEAR MOT [10] metrics MOTA and MOTP. MOTA (Multiple Object Tracking Accuracy) measures the accuracy and MOTP (Multiple

Table 3 Results in the simulated environment

Run #	Prey-predator distance (avg. ± std. dev.) (m)	Run #	Prey-predator distance (avg. ± std. dev.) (m)
1	0.81 ± 0.13	6	0.64 ± 0.17
2	1.22 ± 0.21	7	0.79 ± 0.31
3	0.83 ± 0.15	8	1.18 ± 0.35
4	1.43 ± 0.08	9	1.03 ± 0.22
5	1.38 ± 0.13	10	1.39 ± 0.28

Object Tracking Precision) calculates the precision of the tracking algorithm. MOTA results give a measure of how good the tracking algorithm can keep connect the object identities over time, while MOTP results are useful for evaluating the difference between the bounding box provided by the tracking algorithm and the minimum bounding box containing the tracked person. MOTA is defined as:

$$MOTA = 1 - \frac{\sum_{t=1}^{N_{frames}} (c_m(m_t) + c_f(fp_t) + cs(ID\text{-}SWITCHES_t))}{\sum_{t=1}^{N_{frames}} N_G^{(t)}} \tag{3}$$

where, after computing the mapping for frame t, m_t is the number of misses, fp_t is the number of false positives, $ID\text{-}SWITCHES_t$ is the number of ID mismatches in frame t considering the mapping in frame $(t-1)$, and $N_G^{(t)}$ is the number of objects present in frame t. The values for the weighting functions have been set to $c_m = c_f = 1$ and $c_s = log_{10}$.

To obtain the precision score, we calculated the spatio-temporal overlap between the reference tracks and the output tracks of our method. MOTP was defined as:

$$MOTP = \frac{\sum_{i=1}^{N_{mapped}} \sum_{t=1}^{N_{frames}^{(t)}} \left[\frac{|G_i^{(t)} \cap D_i^{(t)}|}{|G_i^{(t)} \cup D_i^{(t)}|} \right]}{\sum_{t=1}^{N_{frames}} N_{mapped}^{(t)}} \tag{4}$$

where N_{mapped} refers to the mapped system output objects over an entire reference track taking into account splits and merges, and N_{mapped}^{t} is the number of mapped objects in the tth frame.

Table 4 reports MOTA and MOTP values for all the experiments that have been carried out. The results show that the integration of data coming from heterogeneous sensor nodes composed of active RFID tags, RGBD cameras, and mobile laser range finders can be used to deal with the problem of monitoring a populated environment. A more accurate experimental analysis for measuring false positive/false negative rates in different situations and integration with other techniques (e.g., vision) would further improve the assessment of the quality of the system.

Table 4 Results of the tracking method in the simulated environment

Experiment	MOTA	MOTP	Experiment	MOTA	MOTP
1	0.95	0.85	6	0.91	0.87
2	0.97	0.90	7	0.95	0.89
3	0.96	0.88	8	0.95	0.91
4	0.92	0.91	9	0.97	0.93
5	0.99	0.95	10	0.98	0.92

8 Conclusions

Integrating multiple technologies for surveillance applications is an important and necessary step towards the developing and deploying of effective systems. In this book chapter we describe an architecture and several techniques used for integrating heterogeneous fixed and mobile sensor nodes in order to determine the presence and the position of people in an indoor environment. Different technologies (RFID tags, laser range finders, and RGBD cameras) are combined through a distributed data fusion method, which is robust to perception noise and is scalable to multiple heterogeneous sensors. The reported experimental results, obtained both with real and simulated data, show the feasibility of the approach and the overall capabilities of the architecture. Automatic monitoring and detection of abnormal activities are possible and performance in this task can be good enough for an actual deployment. However, additional work must be done in order to make the techniques more precise and more robust.

A potential extension for the approach described in this book chapter consists in adding more types of sensors to the network, such as microphones, GPS receivers, and active RFID tags providing signal strength data. Moreover, the simulated scenario can be enriched by generating realistic error models specific to those additional sensors. In order to improve the multi-sensor architecture designed in this book chapter, the system can be tested by end-users to evaluate the usability and the feasibility of the proposed approach.

Acknowledgments This work has been partially supported by the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research and it has been carried out in cooperation with the SocioPatterns collaboration (www.sociopatterns.org).

References

1. Barrat, A., Cattuto, C., Szomszor, M., den Broeck, W.V., Alani, H.: Social dynamics in conferences: analysis of data from the live social semantics application. In: Proceedings of the 9th International Semantic Web Conference, pp. 17–33 (2010)
2. Becchetti, L., et al.: Population protocols on real social networks. In: Proceedings of the 5th Workshop on Social Network Systems, pp. 15:1–15:2 (2012)

3. Bloisi, D.D., Iocchi, L.: Independent multimodal background subtraction. In: Proceedings of the 3rd International Conference on Computational Modeling of Objects Presented in Images: Fundamentals, Methods and Applications, pp. 39–44 (2012)
4. Van den Broeck, W., Quaggiotto, M., Isella, L., Barrat, A., Cattuto, C.: The making of sixty-nine days of close encounters at the science gallery, Leonardo, pp. 285–285 (2012)
5. Chin, A., Xu, B., Wang, H., Wang, X.: Linking people through physical proximity in a conference. In: Proceedings of the 3rd International Workshop on Modeling Social Media, pp. 13–20 (2012)
6. Cui, J., Zha, H., Zhao, H., Shibasaki, R.: Laser-based detection and tracking of multiple people in crowds. Comput. Vis. Image Underst. 300–312 (2007)
7. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: efficient position estimation for mobile robots. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 343–349 (1999)
8. Hui, P., Chaintreau, A., Scott, J., Gass, R., Crowcroft, J., Diot, C.: Pocket switched networks and the consequences of human mobility in conference environments. In: Proceedings of ACM SIGCOMM First Workshop on Delay Tolerant Networking and Related Topics, pp. 244–251 (2005)
9. Isella, L., Romano, M., Barrat, A., Cattuto, C., Colizza, V., Van den Broeck, W., Gesualdo, F., Pandolfi, E., Ravá, L., Rizzo, C., Tozzi, A.E.: Close encounters in a pediatric ward: measuring face-to-face proximity and mixing patterns with wearable sensors. In: PLoS ONE, p. e17144 (2011)
10. Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., Boonstra, M., Korzhova, V., Zhang, J.: Framework for performance evaluation of face, text, and vehicle detection and tracking in video: data, metrics, and protocol. IEEE Trans. Pattern Anal. Mach. Intell. **31**(2), 319–336 (2009)
11. Liu, C., James, T.Y.: An analysis of dos attacks on wireless lan. In: Wireless and Optical Communications (2006)
12. Palmer, D., Kirschenbaum, M., Murton, J., Zajac, K., Kovacina, M., Vaidyanathan, R.: Decentralized cooperative auction for multiple agent task allocation using synchronized random number generators. In: Proceedings of International Conference on Intelligent Robots and Systems (IROS), vol. 2, pp. 1963–1968 (2003)
13. Pennisi, A.: Multi-robot surveillance through a distributed sensor network experiments. http://www.dis.uniroma1.it/pennisi/scij/
14. Shao, X., Zhao, H., Shibasaki, R., Shi, Y., Sakamoto, K.: 3d crowd surveillance and analysis using laser range scanners. In: International Conference on Intelligent Robots and Systems (IROS), pp. 2036–2043 (2011)
15. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: AAAI/IAAI, pp. 852–858 (2000)
16. Stehlé, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J.F., Quaggiotto, M., Van den Broeck, W., Régis, C., Lina, B., Vanhems, P.: High-resolution measurements of face-to-face contact patterns in a primary school. PLoS ONE **6**(8), e23176 (2011)
17. Xavier, J., Pacheco, M., Castro, D., Ruano, A.: Fast line, arc/circle and leg detection from laser scan data in a player driver. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3930–3935 (2005)
18. Ziparo, V.A., Iocchi, L., Nardi, D., Palamara, P.F., Costelha, H.: Petri net plans: a formal model for representation and execution of multi robot plans. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 79–86 (2008)