10

Multi-Robot and Multi-Camera Patrolling

Christopher King
University of Nevada, Reno

Maria Valera
Kingston University, London, UK

Raphael Grech
Kingston University, London, UK

Robert Mullen
Kingston University, London, UK

Paolo Remagnino
Kingston University, London, UK

Luca Iocchi
University of Rome "La Sapienza", Italy

Luca Marchetti
University of Rome "La Sapienza", Italy

Daniele Nardi
University of Rome "La Sapienza", Italy

Dorothy Monekosso
University of Ulster, UK

Mircea Nicolescu
University of Nevada, Reno

In this chapter we present a multi-camera platform to monitor the environment which is integrated to a multi-robot platform to enhance situation awareness. The multi-camera platform consist of two distinct stereo camera systems as use different vision approaches which will be described in detail. One of the

stereo vision systems is applied to reason on object manipulation events, while the other system is used to detect an event such as a person leaving a bag in a corridor. The results from either of these two systems are encapsulated in a string message and sent via wireless network to the multi-robot system which, on alarm, will dispatch a robot to monitor the region of interest. Our ultimate goal is that of maximizing the quality of information gathered from a given area thus implementing a Heterogeneous mobile and reconfigurable multi-camera video-surveillance system.

10.1 Introduction

The problem of detecting and responding to threats through surveillance techniques is particularly well suited to a robotic solution comprising of a team of multiple robots. For large environments, the distributed nature of the multi-robot team provides robustness and increased performance of the surveillance system. Here we develop and test an integrated multi-robot system as a mobile, reconfigurable, multi-camera video-surveillance system.

The main stages of the pipeline in a video-surveillance system are the moving object detection and recognition, tracking and activity recognition. One of the most critical and challenging components of a semi-automated video surveillance is the low-level detection and tracking phase. Data is frequently corrupted by the camera's sensor (e.g. CCD noise, poor resolution, motion blur, etc.), the environment (e.g. illumination irregularities, camera movement, shadows, reflections, etc.), and the objects of interest (e.g. transformation, deformation, occlusion, etc.). Even small detection errors can significantly alter the performance of routines further down the pipeline, and subsequent routines are usually unable to correct errors without using cumbersome, ad-hoc techniques. Compounding this challenge, low-level functions must process huge amounts of data, in real-time, over extended periods. To adapt to the challenges of building accurate detection and tracking systems, researchers are usually forced to simplify the problem. It is common to introduce certain assumptions or constraints that may include: fixing the camera [31], constraining the background [30], constraining object movement, applying prior knowledge regarding object-appearance or location [29], assuming smooth object-motion, etc. Relaxing any of these constraints often requires the system to be highly specified for the given task. Active contours may be used to track non-rigid objects against homogeneous backgrounds [3], primitive geometric shapes for certain simple rigid objects [10], and articulated shapes for humans in high-resolution images [25]. There has been a push toward identifying a set of general features that can be used in a larger variety of conditions. Successful algorithms include the Maximally Stable Extremal Region (MSER) , Harris-Affine, Hessian-Affine and Salient Regions [22]. Despite their recent successes, each algorithm has its own weaknesses, and achieving flexibility still requires the combination of multiple techniques [27]. Since most

of these approaches are either not real-time, or are barely real-time; running several in unison is usually not feasible on a standard processor.

Recently, to adapt to the challenges of building accurate detection and tracking systems, work has also been carried out using per-pixel depth information provided by stereo imagery devices to detect and track multiple objects [4, 9, 11, 16, 24, 33]. What is mainly thanks to 'improved performance' on software computing for depth imagery [26, 1, 2] and also more affordable stereo imagery hardware [1, 2]. In [4, 9, 11] the detected and tracked features are directly applied on the depth information itself, while in [16, 24] the detection and tracking is done after the analysis of depth information is integrated with the colour information.

In this Chapter we will mainly focus on two approaches to develop two different video surveillance systems. The first approach consists of applying a real-time, colour-based, MSER detection and tracking algorithm. In the second method, a multi-object tracking system is presented based on a ground plane projection of real-time 3D data coming from a stereo imagery, giving distinct separation of occluded and closely-interacting objects. The rest of the chapter is structured as follows: Section 10.2 presents the architecture of the whole integrated system. In Section 10.3 the pipeline of processes of the first camera system is described and in Section 10.4 the second camera system is presented. In Section 10.5 the high level process of the outputs from previous pipeline processes is provided. In Section 10.6, the results from the prototype system are provided to finish in Section 10.7 with the conclusions of this work.

10.2 System Architecture

We considered a highly heterogeneous system, where robots and cameras interoperate. These requirements make the problem significantly different from previous work. Figure 10.1 illustrates the architecture of the system. We also considered different events and different sensors and we will therefore consider different sensor models for each kind of event. We focused on the dynamic evolution of the monitoring problem, where at each time a subset of the agents will be in response mode, while the rest of them will be in patrolling mode. Therefore, the main objectives of the developed system are:

1. Develop environment monitoring techniques through behaviour analysis based on stereo cameras,
2. Develop distributed multi-robot coverage techniques for security and surveillance,
3. Validate our solution by constructing a technological demonstrator showing the capabilities of a multi-robot system to effectively self-deploy itself in the environment and monitor it.

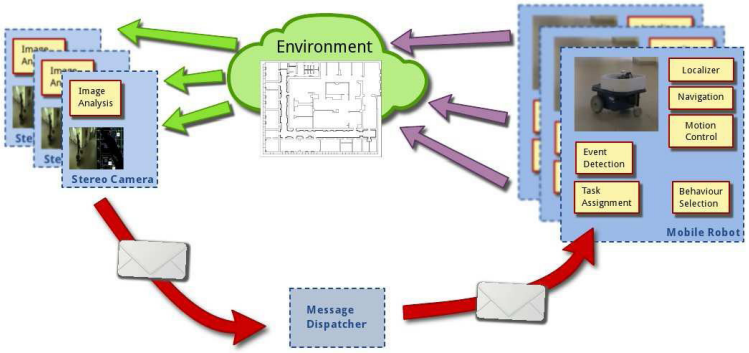

FIGURE 10.1 Block diagram of the proposed architecture.

10.2.1 Multi-Robot Monitoring Platform

As already mentioned, the problem of detecting and responding to threats through surveillance techniques is particularly well suited to a multi-robot platform solution comprising of a team of robots. Although this chapter does not focus on the description of these type of platforms, our approach has been concentrating on extending the work done in multi-robot patrolling, adding the capability for the robots to respond to events detected by visual and other sensors in a coordinate way.

Two issues are considered and solved through the project: Developing a general algorithm for event-driven distributed monitoring based on our previous work has solved these problems. We already developed and successfully implemented new dynamic distributed task assignment algorithms for teams of mobile robots: applied to robotic soccer [18] and for foraging-like tasks [12]. More specifically, in [18] we proposed a greedy algorithm to effectively solve the multi-agent dynamic and distributed task assignment problem, which is very effective in situations where the different tasks to be achieved have different priorities. In [12] we also proposed a distributed algorithm for dynamic task assignment based on token passing that is applicable when tasks are not known a priori, but are discovered during the mission. The problem considered here requires both finding an optimal allocation of tasks among the robots and taking into account tasks that are discovered at run-time. Therefore it is necessary to integrate the two approaches. As a result, we do not only specialise these solutions to the multi-robot surveillance and monitoring task, but also study and develop extensions to these techniques in order to improve the optimality of the solutions and the adaptively to an open team of agents, taking into account the physical constraints of the environment and of the task.

10.2.2 Multi-Camera Platform

The multi-camera platform consists of two stereo cameras. In one of the cameras a novel, real-time, colour-based, MSER detection and tracking algorithm is implemented. The algorithm synergistically Combines MSER-evolution with image-segmentation to produce maximally stable segmentation. Our MSER algorithm clusters pixels into a hierarchy of detected regions using an efficient line-constrained evolution process. Resulting regions are used to seed a second clustering process to achieve image-segmentation.

The resulting region-set maintains desirable properties from each process and offers several unique advantages including fast operation, dense coverage, descriptive features, temporal stability, and low-level tracking. Regions that are not automatically tracked during segmentation can be tracked at a higher-level using MSER and line-features. We supplement low-level tracking with an algorithm that matches features using a multi-phased, kd-search algorithm. Regions are modelled using transformation-invariant features that allow identification to be achieved using a constant-time hash-table. In the other stereo camera a multi-object tracking system is implemented, based on a ground plane projection of real-time 3D data coming from a stereo imagery, giving distinct separation of occluded and closely-interacting objects. This approach consists in tracking, using Kalman Filters [5], fixed templates that are created combining the height and the statistical pixel occupancy of the objects in the scene. These objects are extracted from the background using a Gaussian Mixture Model combining luminance and chroma signals (YUV-colour space [28]) and depth information obtained from the stereo devices used in this work. The mixture model is adapted over time and is used to create a background model that is also upgraded using an adaptive learning rate parameter according to the scene activity level on a per-pixel basis. The results presented in Figures 10.9 and 10.10 illustrate the validity of both approaches.

10.3 Maximally Stable Segmentation and Tracking for Real-Time Automated Surveillance

The feature detection and tracking algorithm proposed in this section was specifically designed to satisfy the existing need for a system that can robustly track multiple deformable objects, in a variety of settings, in real time (15 fps), on a modest processor (4 GHz). The algorithm can be used on both stationary and moving cameras, and provides seamless transitions between each. For increased flexibility, the algorithm tracks regions using complimentary features. These include: colour-blob features, which are typically more reliable for tracking unstructured or deformable objects through significant transformation. It also includes line-corner features, which offer better localisation and are less affected by partial object-occlusion. Features are detected in a way that optimizes performance and feature-stability (Section 10.3.1). Fea-

tures are tracked using an optimized, multi-phased, kd-tree-based approach (Section 10.3.2). Discriminating between foreground and background regions is achieved using a unique background model consisting of high-level features (Section 10.3.3). Modelling and identification of object-regions is achieved using a fast transformation-invariant modelling algorithm, and a constant-time hash-table-based search (Section 10.3.4).

10.3.1 Region Detection

The primary function of the region-detection phase is to massively reduce the amount of input data, while simultaneously preserving useful features. This is usually the most critical and error-prone step of processing. Even a modest 320×240 image contains 76,800 pixels, each of which can present 16,777,216 different values. To reduce unimportant data, detection algorithms typically search an input-image for a set of patterns that are both stable and unique. Stability ensures that the same feature will be detected in future frames, while uniqueness ensures that a tracker can distinguish between the features. Mikolajczyk provides a comparison of the most promising feature-detection techniques [22]. Among those tested, the MSER detector was found to be superior in all scene types and for every type of transformation. Additionally, the MSER detector operated appreciably faster than the competing algorithms, processing 800x640 pixel images at sub-second frame rates using a 4.2 GHz processor.

The MSER algorithm was originally developed by Matas et al. [21] to identify stable areas of light-on-dark, or dark-on-light, in grayscale images. The algorithm is implemented by applying a series of binary thresholds to an image. As the threshold value iterates, areas of connected pixels grow and merge, until every pixel in the image has become a single region. During this process, the regions are monitored, and those that display a relatively stable size through a wide range of thresholds are recorded. This process produces a hierarchical tree of nested MSERs. Unlike other detection algorithms, the MSER identifies comparatively few regions of interest. This is beneficial in reducing computational costs of subsequent phases, but can be problematic when used for general object tracking because there is no guarantee that an object of interest will be represented by a MSER.

To increase the number of detections and improve coverage, Forssen [13] redesigned the algorithm to incorporate colour information. Instead of grouping pixels based on a global threshold, Forssen incrementally clustered pixels using the local colour gradient. Forssen's method is based on the extension to colour bt looking at successive time-steps of an agglomerative clustering of image pixels. Therefore, this process identifies regions of similar-coloured pixels that are surrounded by dissimilar pixels. The seleciton of time steps is stabilsed against intensity scalings and image blur by modelling the distribution of edge magnitude. Although Forssen observed an increase in detections and an improvement in results, his algorithms had some limitations. First, the al-

gorithm deteriorates quickly when confronted with noise or non-edge gradients (occurring on curved surfaces or lightly-textured objects). This deterioration occurs because, at the pixel level, these gradients are nearly indistinguishable from object boundaries. To limit this effect, Forssen applied multiple types of smoothing to his data. This improved stability of some regions, but at the expense of others. The second limitation resulted from Forssen's comparison of adjacent pixels to determine merge criteria. In most video feeds, the spatial correlation of colour information is too high to offer reliable contrast, and MSER stability is greatly compromised. Forssen's response was to normalize edge weights in a way that ensured region growth occurred evenly across the maximum threshold-iteration interval. Although this reduced missed detections, it greatly increased the extent that regions were detected multiple times at slightly different scales. Multiple detections require additional post-processing culling operations, and when combined with the natural inconsistencies of MSER detection, make reliable tracking between frames almost impossible.

Our approach takes advantage of the increased detection offered by Forssen's colour-based approach, while greatly reducing the extent of compromise. Our algorithm offers the following improvements over Forssen's approach:

1. Region-growth is constrained using detected lines. This improves segmentation results on objects with high-curvature gradients.

2. Our MSER evolution process merges three-pixel units, instead of two-pixel units. This reduces computation costs, and allows the gradient to be measured with greater precision.

3. Our algorithm returns either a nested set of regions (traditional MSER-hierarchy formation), or a non-nested, non-overlapping set of regions (typical to image segmentation). Using non-nested regions significantly improves tracking speed and accuracy.

4. Regions in the flat MSER representation are completely filled in with pixels (every pixel in the image is assigned to exactly one region). This produces attractive segmentation and more accurate tracking.

5. Regions are constructed using both spatial and temporal information. This increases stability and speed of operation.

6. Region-tracking is partially achieved at the lowest level of MSER formation. This reduces the number of regions that must be tracked in subsequent phases of the algorithm.

7. The Canny-lines used in segmentation are available for other functions such a tracking or structure analysis.

8. The MSER segmentation portion of our algorithm uses only one threshold "MIN-SIZE", which constrains minimum region-size and MSER-stability. This is an improvement over traditional colour-based MSER algorithm, which requires users to set separate thresholds for minimum size, MSER-stability, nested-region overlap, and others.

Our MSER algorithm is a multi-phase process involving *Line Detection*, *MSER-Tree Construction*, *Region Expansion* and *Region Feed-Forward*.

Line Detection

The traditional colour-based MSER algorithm is largely limited by its strict dependence on the colour-gradient. Theoretically, even if two regions have high gradient measurements spanning all but one pixel of their shared border, that one-pixel break will cause the regions to be detected as one. This characteristic is particularly limiting when the algorithm is applied to real-world videos since noise, movement-blur, shadows, reflections, etc. can all degrade the gradient. The Canny is much more effective at identifying a continuous border between objects since it considers a larger section of the gradient. If a low-gradient gap interrupts a high-gradient border, the gap is labeled as part of the border.

The Canny is also superior to the MSER in its ability to ignore gradients caused by curvature. For example, consider an image containing a non-textured background and a similarly-coloured, curved object (e.g. a hand). The MSER would form a region corresponding to the table, but before the object could form its own stable cluster, its pixels would be stripped away by the table region. In contrast, the Canny would likely produce its strongest response along the table-object border. The resulting outline would isolate pixels within the object and allow them to cluster independently of the table.

Our system processes each frame with the Canny algorithm. Canny edges are converted to line-segments and the pixels corresponding to each line-segment is used to constrain MSER growth. Simply speaking, MSER evolution operates as usual, but is not permitted to cross any Canny lines. An example of detected lines is shown in Figure 10.3 (Right). Detected lines are displayed in green.

MSER-Tree Construction

Our MSER evolution algorithm uses the same basic principle as Forssen's approach [13]. For every current pixel p_c in the image, the colour-gradient is measured against adjacent pixels where p_{c-1} refers to the pixel on the left hand side of the current pixel p_c. Simalarly p_{c+1} is the adjacent pixel on the right. The outcome is then stored as horizontal (t_h) or vertical (t_v) texture elements using the following formula.

$$t_h = \sqrt{512 \times \sum_{c=\{r,g,b\}} \frac{(p_c - p_{c-1})^2}{(p_c + p_{c-1})} + \frac{(p_c - p_{c+1})^2}{(p_c + p_{c+1})}}$$

Texture elements (ranging from 0 to 255) are sorted using a constant-time counting-sort algorithm. They are then processed in order, starting with 0-valued texture elements. For every processed element, the corresponding pixel is merged with its vertical or horizontal neighbours (depending on the direction of the element). If any of the three pixels belong to an existing region, the regions are merged. After all texture elements of a particular value (e.g. $0, 1, 2...255$) are processed, rate-of-growth for all existing regions is measured for that iteration. As long as a region's growth consistently accelerates, or declines, it is left to evolve. If the rate of growth changes from decline (or stable) to acceleration (beyond a MIN-SIZE change), the state of the region before accelerated growth is stored as a MSER. The algorithm continues until all texture elements have been processed. At the end of the growth process, the set of all MSER regions will form a hierarchical tree. The tree-root contains the MSER node that comprises every pixel in the image, with incrementally smaller nested sub-regions occurring at every tree-branch. The leaves of the tree contain the first-formed and smallest groups of pixels. To reduce memory and processing demand, the MIN-SIZE threshold is applied to these regions. Our implementation uses a MIN-SIZE of 24 pixels. Figure 10.2 shows three stages of the clustering process.

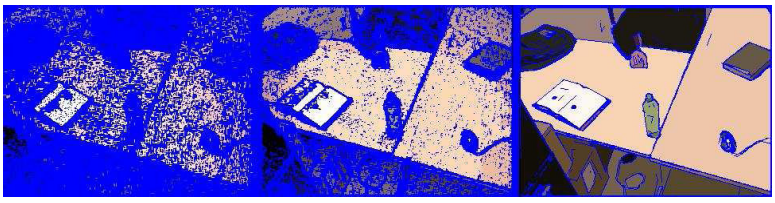

FIGURE 10.2 Pixel clustering during MSER formation. Clustered pixels are coloured using the region's average colour. Non-assigned pixels are shown in blue. Results represent clusters after iterations 2, 5, and 35 (left to right)

Region Expansion

The traditional MSER approach produces, as output, a hierarchical tree of nested nodes. Although this is desirable for certain applications (where over-detection is beneficial), we find it doesn't provide any significant advantages and makes other tasks unnecessarily complicated. Traditional MSER approaches apply various ad-hock devices to suppress the formation of nested regions, or to cull the regions once they occur. We choose to extract a segmented image from the MSER hierarchy instead of eliminating the problem by using a dual-pass MSER evolution process . Our algorithm can enforce that each pixel is contained in exactly one region instead of each image-pixel belonging to zero or more different regions. Both MSER and segmentation representations provide certain unique advantages and our algorithm allows a user to pick the representation that best suits their needs.

The first pass of our dual-pass algorithm was described in the previous section. This produces the traditional hierarchy of nested MSER regions, with tree-leaves representing initial pixel clustering. These leaves are sparsely distributed within the image and are both non-overlapping and non-nested. Using a merging process similar to the one used in the first pass, we iteratively add pixels to the leaves until every pixel in the image is contained in exactly one leaf. During this process, we do not allow leaves to merge with one another. Once all pixels have been added, the hierarchy structure derived from the first pass is used to propagate pixel information up the tree, from the leaves to the root. At this point, every horizontal cross-section of the tree can produce a complete segmented image comprising all pixels. Although regions corresponding to non-leaf nodes may be useful, we choose to ignore them. Our image segmentation results are derived only from regions corresponding to the leaf nodes. Figure 10.2 shows segmentation from a table-top scene. The center image displays the hierarchy of MSER regions, displayed as ellipses. The right image shows segmentation produced using the leaves of the MSER tree.

Region Feed-Forward

Most stable-feature detection algorithms generate an entirely new set of features from every frame of the video sequence. Tracking algorithms are then required to match features between successive frames. Although this is a useful strategy for tracking small or fast-moving objects, it may be unnecessary when tracking large, textureless, objects that are slow-moving or stationary. Without surface texture, pixels within the region's interior don't provide any useful information and re-computing their position every frame wastes resources. Resources would better be applied to pixels near the perimeter of a region, or to pixels that changed between frames. Since large textureless objects can make up significant portions of an image, we observe considerable performance increases using this approach.

In addition to speed advantages, our feed-forward algorithm improves spatial stability by integrating temporal information. Consider a slowly-moving (or stationary) homogeneously coloured object (like a person's wrinkled shirt) that contains enough surface texture to cause spurious MSER regions to form. The inherent instability of these regions makes them unsuitable for tracking or modeling, yet their removal is difficult without using ad-hock strategies. Using our feed-forward approach, any region that can't continually maintain its boundaries, will be assimilated into similarly-coloured adjacent regions. After several iterations of region-competition, many unstable regions are eliminated automatically without any additional processing.

Our feed-forward algorithm is a relatively simple addition to our MSER algorithm. After every iteration of MSER generation, we identify pixels in the current frame that are nearly identical (RGB values within 1) to the pixel in the same location of the following frame. If the majority of pixels in any given

MSER remain unchanged for the following video image, the matching pixels are pre-grouped into a region for the next iteration. This pixel-cluster is then used to seed growth for the next iteration of MSER evolution.

It should be mentioned that an additional constraint must be added for this feed-forward strategy to work properly. To illustrate the problem, consider a stationary scene with an unchanging image. In this example, every pixel will be propagated, and there will be no pixels left for MSER evolution. Every region in the image will remain unchanged and any errors in detection would be preserved indefinitely. A preferable strategy is to propagate pixels that contribute least to MSER evolution (low-gradient pixels), while allowing the MSER to evolve using more descriptive (high-gradient) pixels. To achieve this effect, we compute the average gradient value of pixels in each region, and propagate pixels with gradient values below that average (as an optimization, we also propagate pixels with gradients below a predefined threshold). This technique approximately allows at-least half the pixels in any non-moving region to be propagated forward, while leaving the other half to reconstruct an updated stable region. Figure 10.3 (Left) shows pixels designated for feed-forward. Dark-gray pixels are propagated to the next frame. Light-gray pixels are withheld.

FIGURE 10.3 Left: An example of the feed-forward process. Dark-gray pixels are preserved, Light-gray pixels are re-clustered. Center: MSERs are modeled and displayed using ellipses and average colour-values. Right: An example of MSER image segmentation. Regions are filled with their average colour, detected lines are shown in green, the path of the tracked hand is represented as a red line.

10.3.2 Region Tracking

Region tracking can be defined simply as determining the optimal way detected regions in one frame match regions in subsequent frames. Despite the simple definition, tracking is a challenging problem due to:

1. Object-appearance changes: illumination, transformation, deformation, occlusion
2. Detection errors: false detections, multiple detections, missed detections
3. Detection inconsistencies: inaccurate estimation of position, size, or appearance

Yilmaz et al. [32] reviewed several algorithm, and listed the strengths and weaknesses of each. Yilmaz emphasised that each tracking algorithm inevitably fails under a certain set of conditions and that greater robustness can be obtained by combining strategies. Although this concept works well in theory, implementation can be difficult in real time. Many of the available real-time region detection and tracking algorithms require a significant amount of computer resources to operate, often making the simultaneous operation of non-related algorithms impractical. Additionally, fusing information obtained from several algorithms may create additional problems.

Our tracking algorithm was designed to specifically operate on the complimentary set of features provided by our detection algorithm. As mentioned, our algorithm models regions using MSER features and line-corner features. Each of feature-type provides certain advantages and disadvantages, and our algorithm has been designed with the intent of exploiting the advantages of each. Our tracking algorithm applies four different phases. Each phase is best suited to handle a specific type of tracking problem, and if an object can be tracked in an early phase, later tracking-phases are not applied to the object. By executing the fastest trackers first, we can further reduce resource requirements.

The four phases of our tracking algorithm include: *Feed-Forward Tracking*, *MSER-Tracking*, *Line-Tracking* and *Secondary MSER-Tracking*.

Feed-Forward Tracking

Traditional tracking algorithms match features between successive frames using descriptor similarity measures. This assumes that descriptors do not change significantly between frames. MSER regions can pose significant problems in this regard, since small changes in the image can cause large changes in the descriptors. For example, consider a video sequence taken of a person reaching completely across a table. Immediately before the person's arm bisects the table, a traditional MSER algorithm will detect the tabletop as a single region. Immediately afterward, the tabletop will appear as two smaller regions. Since a MSER tracker only receives information regarding the size and positions of the centroids, resolving the actual path of the region as it splits into two, would likely be a cumbersome process.

Using our pixel feed-forward algorithm, resolving the table-bisection scenario becomes a trivial matter. Since the majority of the table's pixels remain unchanged during the bisection, these pixels will maintain their exiting clustering. Even if the cluster of pixels is non-contiguous, MSER evolution will produce a single region. Tracking becomes a trivial matter of matching the pixel's donor region with the recipient region.

MSER-Tracking

Tracking MSERs has traditionally been an ill-posed problem. It is difficult to control the degree that similarly-shaped regions are nested within one another

and fluctuations make one-to-one region correspondences nearly impossible. As described in the Section 10.3.1, we eliminated the problem of nesting by reducing the hierarchy of MSERs to non-hierarchical image segmentation. This representation theoretically makes one-to-one correspondences possible, and matches are identified using a greedy approach. The purpose of this phase of tracking is to match only those regions that have maintained consistent size and colour between successive frames.

Each image region is represented by the following features:

1. Centroid (x,y) image coordinates
2. Height and Width (second-order moment of pixel-positions)
3. RGB mean colour values

Matching is only attempted on regions that remained unmatched after the Feed-Forward Tracking phase (Section 10.3.2). Matches are only assigned when regions have similarity measures beyond a predefined similarity threshold. Matching is conducted as follows; for every unmatched region in frame t, a set of potential matches in frame $t+1$ is identified using a kd-search tree. Regions-matches that are not sufficiently similar in size, position, and colour, are removed from consideration. All other region-matches are sorted according to the feature-similarity measures of size, position, and colour. Potential matches are processed in order of their similarity measure (from most-similar to least). If both regions are available to be matched, then a tracking-link is provided to the pair. The algorithm proceeds until all potential matches have been considered.

Line-Tracking

Since a primary component of the MSER descriptor is its vertical and horizontal size, tracking can be highly sensitive to occlusion and bifurcation. This makes MSER descriptors unsuitable for use as the sole feature used in tracking. Ideally a second feature should be used, which doesn't require significant additional resources to detect. Since our algorithm already incorporates lines into region-detection, the line features become an ideal candidate for complimenting the MSER. Specifically, we use line-segment end-points, which typically occur on corners of image regions. Line-corners are desirable because they are stable, they are unaffected if a different part of the region is occluded, and they provide good region localization.

In this tracking phase, line-corners are matched based on their positions, the angles of the associated lines, and the colours of the associated regions. It should be mentioned that, even if a line separates (and is therefore associated with) two regions, that line will have different properties for each region. Specifically, the line angle will be 180 degrees rotated from one region to the other, and the left and right endpoints will be reversed.

Each line-end is represented by the following features:

1. Position (x,y) image coordinates
2. Angle of the corresponding line
3. RGB mean colour values of the corresponding region
4. Left / Right handedness of endpoint (Perspective of looking out from the center of the region)

Line-corner matching is only attempted on regions that remained unmatched after the MSER-Tracking phase. Also, matches are only assigned for objects that have similarity measures beyond a predefined similarity threshold. Matching lines is conducted using the same strategy that was described in Section 10.3.2.

Secondary MSER-Tracking

Tracking phases described in Sections 10.3.2 to 10.3.2 assume that features do not change significantly between frames. Although this may generally be the case; for example noise, illumination changes and occlusion may cause information to be degraded or lost in certain frames. To reduce the number of regions lost under these conditions, we conclude our tracking sequence by re-applying our MSER and line-tracking algorithms using looser similarity constraints. This phase uses a greedy-approach to match established regions (regions that were being tracked but were lost) to unassigned regions in more recent frames. Unlike the first three phases, which only consider matches between successive frames, the fourth phase matches regions within an n-frame window ('n' is usually fewer than 8). In this case, the established region's motion model is used to predict its expected location for comparison.

10.3.3 Foreground Detection

The sequence of steps of our approach to process a video feed is as follows:

1. Initially, construct a region-based background model
2. Cluster pixels in subsequent frames into regions
3. Track all regions
4. Identify regions in subsequent frames that differ from the background model
5. Update the background model using background regions

Since the background model in our approach comprises higher-level features, we can apply the algorithm in a greater variety of settings. For example, a motion model can be trained, allowing foreground detection to be performed on both stationary and panning surveillance systems. Additionally, since background features are continually tracked, the system is equipped to identify

unexpected changes to the background. For example, if a background region moves in an unexpected way, our system can identify the change, compute the new trajectory, and update the background model accordingly.

Although there may be several ways to achieve foreground detection using our tracking algorithms, we feel it would be appropriate in these early stages of development, to simply reproduce the traditional pipeline. To this affect, the first several frames in a video sequence are committed to building a region-based model of the background. Here, MSERs are identified and tracked until a reasonable estimation of robustness and motion can be obtained. Stable regions are stored to the background model using the same set of features listed in the tracking section. The remainder of the video is considered the operation phase. Here, similarity measurements are made between regions in the background model, and regions found in the current video frame. Regions considered sufficient dissimilar to the background are tracked as foreground regions. Matching regions are tracked as background regions. Since we employ both tracking information and background-model comparison, our system can identify when background regions behave unexpectedly. We then have a choice to either update the background model, or to track the region as foreground.

10.3.4 Object Modelling

Once the foreground is segmented from the background, a colour and shape-based model is generated from the set of foreground MSER features. This model is used to resolve collisions and occlusions, and to identify if a familiar object has re-entered the scene. A common modelling approach is to identify a set of locally invariant features. Lowe [19] proposed a technique where an image patch is sampled within a pre-specified distance around a detected region. The texture within the patch is binned into a 4×4 grid to form a Scale Invariant Feature Transform (SIFT). The resulting descriptor contains a 128 dimensional vector ($4 \times 4 \times 8$-bins). Despite its popularity, this technique is not effective when the object of interest undergoes significant transformation, or contains significant depth disparities. The 128 dimensional SIFT also requires significant computational resources for recording and matching.

Chum and Matas [8] describe a more efficient approach to modelling that reduces descriptor dimensionality to six features. The small dimensionality allows a constant-time hash table to be used for feature comparison. Chum's descriptors are based on MSER region pairs. Each MSER region is transformed into a locally invariant affine frame (LAF). The centroids are identified, as are two extremal points around the region's perimeter. The six-feature descriptor is formed using angles and distances between three point-pairs. Since there may be multiple transformations to the affine frame, each region may have multiple possible descriptors. A voting technique is implemented using the hash table to identify likely candidates. This is a constant time operation, making the technique orders of magnitude faster than patch-style algorithms. It is also less affected by depth discontinuities for foreground objects.

Our technique uses many of the principles presented by Chum, but our feature vectors were selected to provide improved robustness in scenes where deformable or unreliable contours are an issue. Chun was able to use descriptors with relatively low dimensionality because they provided a high degree of precision in estimating the transformation parameters of flat objects. We took the opposite approach by selecting a relatively large number of invariant features with low individual descriptive value. Even though individual features are likely to provide an inaccurate representation of our objects, the combined vote of many unrelated features should provide reasonably discriminatory abilities.

We propose an algorithm that represents objects using an array of features that can be classified into three-types: 1) MSER-pairs, 2) MSER-individuals, and 3) Size-position measure

- Our MSER-pair features are described using a 4-dimensional feature-vector. The first two dimensions (v_1, v_2) are computed by taking the ratio of colour values (in RGB colour space) between the two MSERs: $\frac{(red_1/grn_1)}{(red_2/grn_2)}$, and $\frac{(blu_1/grn_1)}{(blu_2/grn_2)}$. The third dimension (v_3) is the ratio between the area-square-roots: $\frac{\sqrt{area_1}}{\sqrt{area_2}}$. The fourth dimension (v_4) is the distance between ellipse-centroids, divided by the sum of the ellipse-diameters (cut along the axis formed by the line connecting the centroids). Descriptor values v for the MSER pair a and b are computed such that the ratio is between -1 and 1 for each vector dimension f:

$$v_f = \begin{cases} (1 - a_f/b_f) & \text{if } a_f < b_f \\ -(1 - b_f/a_f) & \text{otherwise} \end{cases}$$

- Our MSER-individual features are described using a 3-dimensional feature-vector. The first two dimensions (v_1, v_2) are a measure of the region's colour: red/grn, and blu/grn. The third dimension is a measure of curvature for the object's perimeter. Values range from one (parallelograms) to zero (regions not bound by lines).

- The final feature-set is only used for computing vote-tally. When models are generated, the relative size and position of the contained features are recorded. When features are tested against models in subsequent iterations, this information is used to approximate size and position for every object that receives a vote. To win the vote-tally, an object must receive a sufficient number of votes, which agree on these approximations. The first feature-dimension (v_1) is the ratio between the square-root of the MSER's area, and the square-root of the area containing all object-MSERs. The second dimension (v_2) represents the position of the MSER, in relation to the other MSERs in the object. This feature is only used when a consistent object orientation is expected. Since people in surveillance videos are not likely

to display vertical-symmetry, the value for the features is a function of its vertical position in the object (negative-one at the bottom, positive-one at the top).

10.4 Real-Time Multi-Object Tracking System Using Stereo Depth

In this section, a multi-object tracking system is presented based on a ground plane projection of real-time 3D data coming from a stereo imagery, giving distinct separation of occluded and closely-interacting objects. Our approach consists in tracking, using Kalman Filters [5], fixed templates that are created combining the height and the statistical pixel occupancy of the objects in the scene. These objects are extracted from the background using a Gaussian Mixture Model combining luminance and chroma signals (YUV-colour space) and depth information obtained from the stereo devices used in this work. The mixture model is adapted over time and it is used to create a background model that is also upgraded using an adaptive learning rate parameter according to the scene activity level on a per-pixel basis. The results presented illustrate the validity of the approach.

The next section illustrates the segmentation algorithm used to achieve the foreground detection. Section 10.4.2 explains the idea behind the creation of plan-views used to track objects. Section 10.4.3 presents the tracking procedure and data association.

10.4.1 Foreground Detection

The background subtraction model presented follows the excellent work by M. Harveille et al. [15, 14, 28]. It applies the well-known statistical method for clustering called Gaussian Mixture Model, per-pixel, dynamically adapting the expected background using four channels: three colour channels (YUV colour space) and a depth channel. The input to the algorithm is a synchronised pair of colour and depth images extracted from a single fixed stereo rig* The depth is calculated by triangulation knowing the intrinsic parameters and the disparity of the stereo rig, as shown in the first left part of the Equation 10.5. A 3D world point is projected at the same scan line to the left and right image of the stereo rig once it is calibrated. The displacement between each camera of theis projection point is called disparity. The data set observation for a pixel i at time t is composed as follows: $X_{i,t} = [Y_{i,t} \quad U_{i,t} \quad V_{i,t} \quad D_{i,t}]$ and the observation history data set for pixel i at the current observation is as follows:

*www.videre.com

$[X_{i,1}...X_{i,t-1}]$. Therefore, the likelihood of $X_{i,t}$ taking into account the prior observations is defined as:

$$P(X_{i,t}|X_{i,1}...X_{i,t-1}) = \sum_{j=1}^{K} \delta_{i,t-1,j}\varphi(X_{i,t}; \theta_j(\mu_{i,t-1}, \Sigma_{i,t-1})) \tag{10.1}$$

where δ is the mixing weight of past observations:

$\sum_{j=1}^{K} \delta_{i,t-1,j} = 1$ and $\delta_j > 0$;

$\theta_j(\mu_{i,t-1}, \Sigma_{i,t-1})$ is the gaussian density function component.

The number of gaussians used (i.e.K) initially was 5, although results obtained from posterior experiments showed that using 4 gaussians were equally as good. Assuming the independence between measurements, the covariance matrix is constructed as a diagonal matrix whose diagonal components are the variance of each component in the data set illustrated above. In order to reduce the computation time, the matching process between the current observation (per pixel) and the appropriate gaussian is completed following an on-line K-means approximation, as it is done in [15]. The first step in the matching process is to sort all the gaussians in a decreasing weight/variance order which implies to give preference to the gaussians that have been largely supported by previous consistency observations. Only the variance corresponding to luminance is used in the sorting as depth and chroma data may be unreliable sometimes. The second step in the matching process is to select the first gaussian that is close enough to the new observation by comparing the square difference between the gaussian's mean and the current observation with a fixed threshold value. If this difference is below the threshold the gaussian is selected. The value of the threshold, after several experiments, was set to 4. Then, if a match is found, the parameters of the selected gaussian (i.e. the mean and its variance) are updated taking into account the new observation. As stated before, the depth measurements are sometimes unreliable due to lighting variability or lack of texture in the scenes which implies that the gaussians used to represent the background can contain observations whose depth measurements can be valid or invalid. If many of these observations of a particular gaussian are depth error measurements, the depth mean and variance of the gaussian is considered unreliable and therefore its statistics cannot be used for the comparision with current observations. For that reason, only depth statistics of a gaussian are taken into account if a fraction of its valid depth observations are above the fixed threshold (i.e. 0.2). The square difference regarding depth is calculated once the current depth observation and the depth statistics of the gaussian are validated. If the difference is below the threshold, this indicates high probability that the pixel belongs to the background, so the fixed threshold is augmented by a factor 4, increasing the

colour matching tolerance. This addition allows dealing with cases for example where shadows appear, which match the background depth but not so well the background colour. On the contrary, if the difference is above the threshold, this indicates high probability that the pixel belongs to the foreground, and a foreground flag is set. Before proceeding to calculate the luminance difference, the luminance component of the current observation and the gaussian's mean are checked to be above minimum luminance value, which would imply that the chroma data is reliable and therefore it can be used for the comparison. If a match it is not found and the foreground flag has not been set, the last gaussian in the sorting process is replaced by a new gaussian with a mean equal to the new observation and low initial weight. The update equations for the selected gaussian and for the weights for all the gaussians are described as follows:

$$\begin{aligned}
\mu_{Y,i,t,k} &= (1-\alpha)\mu_{Y,i,t-1,k} + \alpha Y_{i,t,k} \\
\mu_{U,i,t,k} &= (1-\alpha)\mu_{U,i,t-1,k} + \alpha U_{i,t,k} \\
\mu_{V,i,t,k} &= (1-\alpha)\mu_{V,i,t-1,k} + \alpha V_{i,t,k} \\
\mu_{D,i,t,k} &= (1-\alpha)\mu_{D,i,t-1,k} + \alpha D_{i,t,k} \\
\sigma^2_{Y,i,t,k} &= (1-\alpha)\sigma^2_{Y,i,t-1,k} + \alpha(Y_{i,t} - \mu_{Y,i,t-1,k})^2 \\
\sigma^2_{C,i,t,k} &= (1-\alpha)\sigma^2_{C,i,t-1,k} + \alpha((U_{i,t} - \mu_{U,i,t-1,k})^2 + (V_{i,t} - \mu_{V,i,t-1,k})^2) \\
\sigma^2_{D,i,t,k} &= (1-\alpha)\sigma^2_{D,i,t-1,k} + \alpha(D_{i,t} - \mu_{D,i,t-1,k})^2
\end{aligned} \tag{10.2}$$

The weight update equation for all gaussians is as follows:

$$\delta_{i,t,k} = (1 - \mathit{ActivityRecognition})\delta_{i,t-1,k} + \alpha M_{i,t,k} \tag{10.3}$$

where $M_{i,t,k} = 1$ for the matched gaussian and zero for the rest of gaussians.

Finally, once the gaussians are updated, every pixel in each processed frame is labelled as foreground if it was not matched to any gaussian belonging to the background model. Morphological operations are applied to remove isolated regions to fill small foreground holes.

Adaptive Learning Rate Parameter

In the equations illustrated above, α can be seen as a learning rate parameter as its value indicates how quickly the gaussians will adapt to the current observation; if α has a big value, it implies that the gaussians will get close to the new observations in faster incremental steps. In other words, static changes in the background are incorporated to the background model quickly. However, it also implies that foreground objects which have remained static for a certain time are quickly added to the background. A good compromise with α factor is found in [15] where its dynamic value is directly linked with

the amount of activity level of the scene (as the authors called it). The activity level indicates the luminance changes between frames:

$$Ac_{i,t,k} = (1 - \rho)Ac_{i,t-1,k} + \rho|Y_{i,t} - Y_{i,t-1}| \tag{10.4}$$

Initially as follows in [16] the activity level (Ac) defined by Equation 10.4 is set to zero, and after it is computed as the difference in luminance between current frame and previous frame. Therefore, if the activity threshold is above the fixed threshold, which in this study was experimentally fixed to 5 , the α factor used to update the gaussians' statistics is reduce by a experimental factor of 5.

10.4.2 Plan-View Creation

In this section the algorithm that renders 3D foreground cloud data as if the data was viewed from an overhead, orthographic camera is presented. The main reason to apply this transformation is for the computational performance increase, by reducing the amount of information when the tracking is done onto plan-view projection data rather than onto 3D data directly. The projection of the 3D data to a ground plane is chosen due to the assumption that people usually do not overlap in the direction normal to the ground plane. Therefore, this 3D projection allows to separate and to track easily than in the original camera-view. Any reliable depth value can be back projected to its corresponding 3D point knowing the camera calibration data and the perspective projection. Therefore, the first step on the creation of the plan views is to only back project the foreground pixels detected in the previous algorithm, creating a 3D foreground cloud of visible points to the stereo camera. Then, the space of the 3D cloud points is quantised into a regular grid of vertically orientated bins. Looking these vertical bins as the direction normal to the ground plane, some statistics regarding the 3D cloud points can be calculated within each bin. Therefore, a plan view image is constructed as a binary image where each pixel represents one vertical bin and the value of the pixel is some statistics of the 3D cloud point stored in the vertical bin. Two types of plan view images are creating regarding to the two interesting statistics of the 3D could points stored in the vertical bins: the occupancy, i.e. the number of points accumulated in each vertical bin, and the height, i.e. the highest height of the 3D point cloud within each vertical bin. The first statistics indicates the amount of foreground projected on the ground plane and the second statistics indicates the shape of the 3D foreground could. In order to compensate for the smaller camera-view effect appearance of distant object, the first statistic (i.e. the occupancy) is constructed as a weighted number of points accumulated in each bin. The factor used to calculate the occupancy map as suggested in [16, 23] is Z^2/f, where Z is the depth value and f is the focal length. The following equations describe the steps used to create the maps. Figure 10.4 and Figure 10.5 graphically describe the process

of the projection onto ground plane of the 3D foreground cloud points and the creation of the binary plan view image. Once the plan view binary images are created, a posterior refinement is applied to the images in order to remove much of the noise that appear in the occupancy and height maps (see the plan view occupancy on the right side of Figure 10.5).

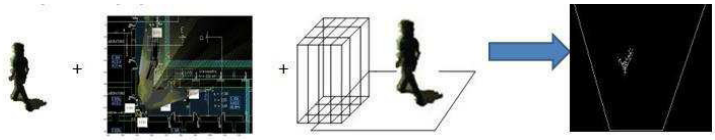

FIGURE 10.4 Illustrates the process of the creation of a plan view.

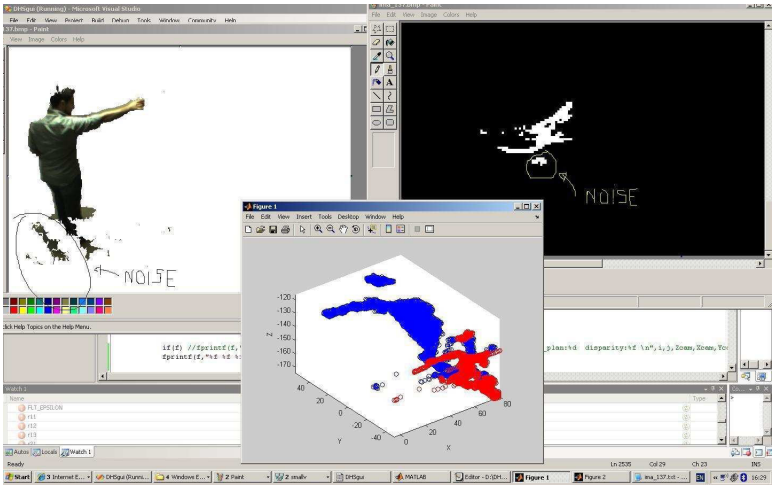

FIGURE 10.5 Illustrate the projection to the ground plane of the valid depth image points of the back projected 3D foreground cloud of points.

Using the internal calibration parameters, any foreground pixel can be back-projected to a 3D cloud point:

$$Z_{cam} = \frac{bf_u}{disparity}, X_{cam} = \frac{Z_{cam}(u - u_o)}{f_u}, Y_{cam} = \frac{Z_{cam}(v - v_o)}{f_v} \qquad (10.5)$$

where,(u, v) is a pixel in the image plane, (u_o, v_o) is the image centre of the projection, f_u and f_v are the horizontal and vertical focal lengths, b (baseline) is the distance between left and right stereo camera and *disparity* is the difference between the pixel value seen from the left camera and the corresponding pixel value seen from the right camera.

We render the 3D cloud point obtained to an overhead camera view (X_w, Y_w, Z_w):

$$\begin{bmatrix} X_W & Y_W & Z_W \end{bmatrix}^T = -R_{cam} \begin{bmatrix} X_{cam} & Y_{cam} & Z_{cam} \end{bmatrix}^T - T_{cam} \quad (10.6)$$

where,

Z_W is aligned with the direction normal to the ground plane, X_W and Y_W are the ground plane axis, R_{cam} and T_{cam} are the rotation and translation matrices

We discretise the vertical bins:

$$\begin{aligned} x_{plan} &= \lfloor (X_W - X_{min})/\lambda + 0.5 \rfloor \\ y_{plan} &= \lfloor (Y_W - Y_{min})/\lambda + 0.5 \rfloor \end{aligned} \quad (10.7)$$

where λ is the resolution factor. In this case the value was set to 2cm/pixel.

10.4.3 Tracking Plan-View Templates

This section describes the tracking algorithm applied which is also based on the work presented in [16, 23]. The Gaussian and linear dynamic prediction filters used to track the occupancy and height statistics plan view maps are the well-known Kalman Filters; more precisely, the OpenCV implementation of Kalman Filters [17]. The state vector is conformed to the 2D position (center of mass) of the tracked object in the plan view, to the 2D velocity component of the object and to the shape configuration of the object, which can be defined with the occupancy and height statistics, as it is described in previous section. In this application, an object could be a robot, a person or a bag. The input data to the filter is simple fixed templates of occupancy and height plan view maps. These templates (τ_H, τ_O) are small areas of the plan view binary images extracted at the estimated location of the object. To create these templates, it is assumed that the statistics of an object are quite invariant to ground plan location relative to the camera. Therefore, the size of the template (40 pixels) remains constant all the time for all the objects. Moreover, to avoid sliding the templates through time, after the tracking process has been applied, the template is centered back to the 2D position of the object i, rather than to the estimated position of object i.

Correspondence

The area to search is centered to the estimated 2D position of the object. The correspondence is resolved via match score, which is computed at all locations within the search zone. A lower match score value implies a better match. The following equation illustrates the computation of the match score:

$$\begin{aligned} \varphi(i, X) &= \rho SAD(\tau_H, H_{masked}(X)) + \omega SAD(\tau_o, \theta_{sm}(X)) \\ &\quad + \beta \sqrt{(x - x_{pred})^2 + (y - y_{pred}^2)} + \alpha \sum_{j<i} \theta_j(X, 40) \end{aligned} \quad (10.8)$$

SAD refers to the sum of absolute differences between the height and occupancy templates and the occupancy and height plan view maps created from the current frame. The $(\rho, \omega, \beta, \alpha)$ weights are deduced in [16] based on physical principals. The first two weights are formulated in a way that the height and occupancy *SAD*s have similar contribution. The third and fourth weights are formulated reflecting the area contribution of the foreground pixels to each vertical bin. It is deduced, from Eq. 10.7, that the match score of person i at 2D position X is the sum of several contributions. The contribution of the weighted sum of the difference between the shape and visible region of tracked object i at location X from the shape and visible region of the object in the current frame. It is also the contribution of the distance between the current object's position and the estimated location of the object. The last contribution to the match score corresponds to the convergence distance of X to the rest of predicted locations of the rest of objects in the current frame. Once the best match score is chosen (i.e. the smallest value), it is compared to a threshold track, where it is related to the minimum amount of foreground pixel area needed to consider the tracked region a valid object. Only if the match score is below the threshold track will the object's Kalman state be updated with the new measurements. In Figure 10.6 some results are presented. A few key frames from a sequence (2300 frames) are presented in this figure, where three different types of objects interact and an occlusion between two of the objects (a person and a robot) appears and the tracker is able to solve the occlusion. The images are captured using a videre stereo camera with VGA resolution.

10.5 Activity Recognition

One objective of a visual surveillance system is to identify when people leave, take, or exchange objects. To test the capabilities of our detection and tracking system, we are implementing a simple scenario involving the exchange of baggage. Specifically, the system will be designed to send a report if a person is observed leaving an object, taking another person's object, or removing something from the environment. Recognizing these types of actions can be done without sophisticated algorithms, so for this demonstration, we use a simple rule-sets based only on proximity and trajectories:

1. If an unknown object appears in the environment, models will be generated for that object and for the nearest person. If the associated person is observed moving away from the object, it will be considered "abandonment", and a report of the incident will be generated. If the same person is observed reacquiring the object, the report will be canceled.

2. If an object is associated with one person, and a second person is observed moving away with the object, it will be considered an "exchange" and a report will be generated containing models of the object and both involved

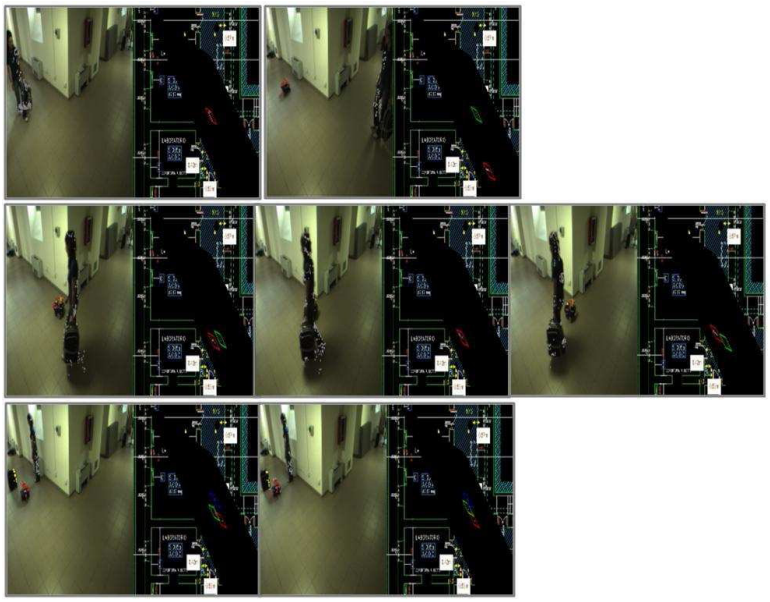

FIGURE 10.6 Seven frames of a sequence that shows tracking different type of objects (robot, person and bag), including an occlusion. Each plan view map has been synchronized with its raw frame pair and back projected to the real plan of the scene (right side of each image).

people.

3. If a person is observed moving away with an object that was either present at the beginning of the sequence, or left by another, the incident will be considered a "theft" and a report will be generated containing the person and the object.

Figure 10.7 shows images taken during table-top (Left), and trash-fire (Right) scenarios. In both demonstrations, the system used colour-based models to represent the object and a Hidden Markov Model was used to generalize object interactions. The system was able to accurately identify interactions over extended periods, in real-time.

10.6 System Integration

In this chapter we present the integrated system on monitoring a large area with the following characteristics:

1. The system is composed of a number of agents, some of them having mobile capabilities (mobile robots) whilst others are fixed (video cameras).

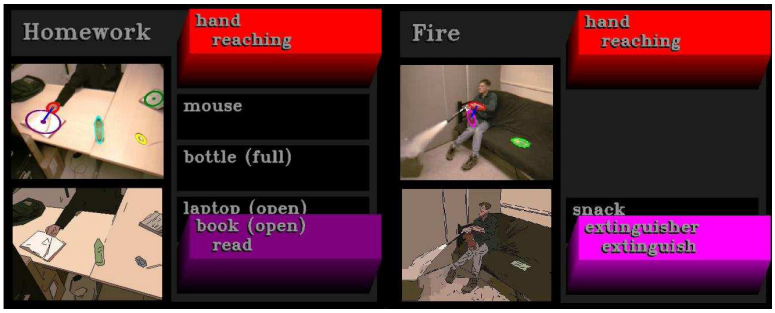

FIGURE 10.7 An example of activity recognition using our system. Each object is associated by a colour bar at the right of the image. The apparent height of the bar corresponds to the computed probability that the person's hand is interacting with that object. In the scenario shown on the left, a person engaged in typical homework-type behaviors including: typing on a laptop; turning pages in a book; moving a mouse; and drinking from a bottle. In the scenario on the right, a person reached into a bag of chips multiple times, and extinguished a trash-fire with a fire extinguisher.

2. The system is required to monitor and detect different kinds of predefined events at the same time.
3. Each agent has a set of sensors that are useful to detect some event. Sensors are of a different type within the entire system.
4. The system is required to operate in two modes:
   (a) patrolling mode
   (b) response mode

These requirements make the problem significantly different from previous work. First of all, we consider a highly heterogeneous system, where robots and cameras inter-operate. Second, we consider different events and different sensors and we will therefore consider different sensor models for each kind of event. Third, we will study the dynamic evolution of the monitoring problem, where at each time a subset of the agents will be in response mode, while the rest of them will be in patrolling mode.

10.6.1 Experimental Scenario

The scenarios used for the experimental validation were tested in the campus of the Department of Computer and System Science (DIS) of Sapienza University in Rome, Italy* . Two scenarios were used to test the capabilities of the

*www.dis.uniroma1.it

multi-camera and robot platform. In the first scenario (i.e. unattended baggage event), the system was designed to send a report if a person was observed leaving a bag. In the second scenario (i.e. object manipulation), the system should send a report if a person manipulated an unauthorised object from the environment. The selected scenario shown in Figure 10.8 was an indoor corridor(left) to simulate the unattended baggage event and a lab room(right) to simulate the object manipulation.

FIGURE 10.8 Experimental scenario at DIS

Once a report is sent, a guard robot would be commissioned to go and take a high-resolution picture of the scenario. Recognising these types of actions may be done without sophisticated algorithms, so for this demonstration, we use simple rule-sets based only on proximity and trajectories:

Scenario 1:

- If a person is observed manipulating an object that was either present at the beginning of the sequence, or left by another person (i.e. unauthorised object), the incident will be considered an "allert" and a report will be generated and sent to the multi-robot system.

Scenario 2:

- If a bag appears in the environment, models will be generated for that bag and for the nearest person. If the associated person is observed moving away from the bag, it will be considered a "left bag", and a report of the incident will be generated.
- If a bag is associated with one person, and a second person is observed moving away with the bag, it will be considered a "bag taken" and a report will be generated and sent to the multi-robot system.

10.6.2 Multi-robot Environmental Monitoring

The implementation of the Multi-Robot Environmental Monitoring used in this project has been implemented on a robotic framework and tested both

on 2 Erratic robots* and on many simulated robots in the Player/stage environment*. Figure 10.1 shows the block diagram of the overall system and the interactions among the developed modules. In particular, the team of robots monitors the environment while waiting for receiving event messages from the vision sub-system. In our system, we use a Bayesian Filtering method to achieve the Sensor Data Fusion. In particular, we use a Particle Filter for the sensor filters and event detection layer. In this way, the probability density functions (PDFs) describing the belief of the system about the events to be detected are described as sets of samples, providing a good compromise between flexibility in the representation and computational effort. The implementation of the basic robotic functionalities and of the services needed for multi-robot coordination are realized using the OpenRDK toolkit* [6]. The mobile robots used in the demonstrator have the following features:

- Navigation and Motion Control based on a two-level approach: a motion using a fine representation of the environment and a topological path-planner that operates on a less detailed map and reduces the search space; probabilistic roadmaps and rapid-exploring random trees are used to implement these two levels [7]:
- Localization and Mapping based on a standard particle filter localization method and a well-known implementation GMapping* that has been successfully experimented on our robots also in other applications [20].
- Task Assignment based on a distributed coordination paradigm using utility functions [18] already developed and successfully used in other projects.

Moreover, to test the validity of the approach, we replicate the scenarios in the Player/Stage simulator, defining a map of the real environment used for the experiments, and several software agents, with the same characteristics of the real robots. The combination of OpenRDK and Player/Stage is very suitable for development and experimentation in multi-robot applications, since they provide a powerful but yet flexible and easy-to-use robot programming environment.

10.6.3 Results

In this section we present the outputs of the recognised scenarios by both camera systems mentioned above. As stated, both systems are integrated to a multi-robot system to increase the situation awareness. The integration of

*www.videre.com
*playerstage.sourceforge.net
*openrdk.sf.net
*openslam.org/gmapping.html

both platforms is done through a TCP client-server communication interface. Each static stereo camera is attached to a PC computer and they communicate between them and the robots via a private wireless network. Each static camera and its PC act as a client and one of the PCs also acts as a server. The PC video-server is the only PC that communicates directly with the robots; the PC video-server becomes a client however, when it communicates with the multi-robot system, and then, the robots act as servers. Once the video surveillance has recognised an event; the PC client camera sends to the PC video-server the event name and the 3D coordinates. The video-server then constructs a string with this information (first transforming the 3D coordinates to a common platform coordinate system) and sends the message via the wireless network to the robots. Then, one of the robots is assigned to go and guard the area and take a high-resolution picture if the event detected is "bag taken". Figure 10.9 and Figure 10.10 show the results in Scenarios 1 and 2 respectively. Figure 10.9 illustrates a sequence of what may happen in

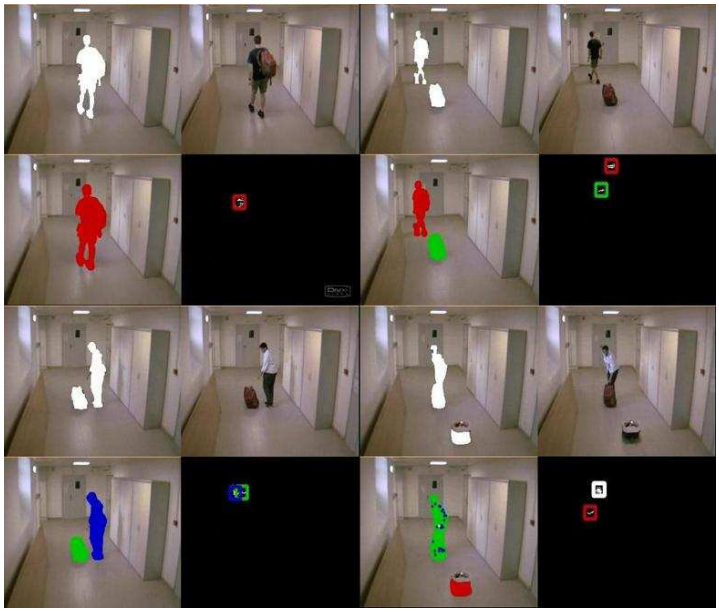

FIGURE 10.9 This figure illustrates a sequence of what may happen in Scenario 1. Person A walks through the corridor with the bag and leaves it in the middle of the corridor. Person B approaches the bag and takes it, raising an alarm in the system causing the patrolling robot to go and inspect the area.

Scenario 1. On the top-left image of the figure a person with an object (bag) is walking through the corridor. On the top-right image of the figure, the video system detects that the person left the bag. Therefore a message is sent as "left bag". On the bottom-left image another person walks very closed to the bag. On the bottom-right image the visual surveillance system detects that a

person is taking a bag and a message "bag taken" is sent to the robots, and as it can be seen one of the robots is sent to inspect the risen event. Figure 10.10

FIGURE 10.10 This figure illustrates a sequence of what may happen in Scenario 2. Person B places a book(black) and a bottle(green) on the table and manipulates them under the surveillance of the system; until Person B decides to touch an unauthorised object (ie.laptop)(grey) raising an alarm in the system causing the patrolling robot to go and inspect the area.

illustrates a sequence of what may happen in Scenario 2. On the top-left, a laptop is placed on the table and one of the robots can be seen patrolling. In the top-right and bottom left images of the figure there is a person who is allowed to manipulate different objects. On the bottom-right the person is touching the only object, which is not allowed, therefore an alarm "allert" is raised.

10.7 Conclusion

The extent that traditional surveillance system can cover large areas, is primarily limited by the number of video feeds a human operator can monitor. This limitation has generated demand for an automated surveillance system. This work is part of a funded project which aims to overcome the current limitations in static visual surveillance by incrementing the situation awareness monitoring, making it flexible and dynamic. These enhanced, integrated multi-robot coordination and vision-based activity monitoring techniques, ad-

vance the state-of-the-art in surveillance applications. Using a group of mobile robots combined with fixed surveillance camera systems has several significant advantages over solutions that only use fixed surveillance camera system. For example, the robots in the team have the power to collaborate on the monitoring task and are able to pre-empt a potential threat. Moreover, the multi-robot platform can communicate with a human operator and receive commands about the goals and potential changes in the mission, allowing for a dynamic, adaptive solution. In this chapter, we introduced a maximally stable segmentation algorithm that efficiently divides image-sequences into spatially and temporally stable regions. By tracking these regions, our system can more quickly discriminate between local and global changes in the image, and can use that information to intelligently update environment and object-models. We have successfully tested our system in a number of real-world activity-recognition scenarios, and are currently working to apply it to a multi-camera surveillance system. Also, a real-time multi-object tracking system for a stereo camera has been presented. Furthermore, these computer vision algorithms presented in Sections 10.3 and 10.4 can eventually be portable to a mobile robot platform. Therefore our solution could be used in environments that previously have not been equipped with a camera-based monitoring system: the robot team could be deployed quickly to obtain information about an unknown environment, allowing the robots to position themselves within the environment in order to best acquire the necessary information.

Acknowledgements

This publication was developed under Department of Homeland Security (DHS) Science and Technology Assistance Agreement No. 2009-ST-108-000012 awarded by the U.S. Department of Homeland Security. It has not been formally reviewed by DHS. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security. The Department of Homeland Security does not endorse any products or commercial services mentioned in this publication.

References

1. Point grey stereo cameras. `http://www.ptgrey.com/`.
2. Videre stereo cameras. `http://www.videredesign.com/`.
3. A. Baumberg and D.C. Hogg. Learning deformable models for tracking the human body. *Motion-Based Recognition, Shah, M., Jain, R. (Eds.)*, pages 39–60, 1996.
4. D. Beymer and K. Konolige. Real-time tracking of multiple people using conti-

nous detection. *In Proc. International Conference on Computer Vision (ICCV'99), Frame-Rate Workshop*, 1999.
5. R.G. Brown and P.Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*, volume 2nd ed. John Wiley & Sons, 1997.
6. D. Calisi, A. Censi, L. Iocchi, and D. Nardi. OpenRDK: a modular framework for robotic software development. In *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1872–1877, 2008.
7. D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Autonomous navigation and exploration in a rescue environment. In *Proceedings of the 2nd European Conference on Mobile Robotics (ECMR)*, pages 110–115, 2005.
8. O. Chum and J. Matas. Geometric hashing with local affine frames. *CVPR*, pages 879–884, June 2006.
9. R. Cipolla and M. Yamamoto. Stereoscopic tracking of bodies in motion. *Image Vision Comput.*, 8(1):85–90, 1990.
10. D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, 2003.
11. T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. *Eighth IEEE International Conference on Computer Vision (ICCV 2001)*, 2:628–635, 1990.
12. A. Farinelli, L. Iocchi, D. Nardi, and V. A. Ziparo. Assignment of dynamically perceived tasks by token passing in multi-robot systems. *Proceedings of the IEEE*, 94(7):1271–1288, 2006.
13. P.-E. Forssén. Maximally stable colour regions for recognition and matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007. IEEE Computer Society, IEEE.
14. M. Harville. Stereo person tracking with adaptive plan-view statistical templates. *Image and Vision Computing*, 22:127–142, 2002.
15. M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. *Detection and Recognition of Events in Video, IEEE Workshop on*, 0:3, 2001.
16. M. Harville and D. Li. Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:398–405, 2004.
17. Intel. Opencv: Open source computer vision library. `http://opencv.willowgarage.com/documentation/ index.html`.
18. L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa. Distributed coordination in heterogeneous multi-robot systems. *Autonomous Robots*, 15(2):155–168, 2003.
19. D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
20. L. Marchetti, G. Grisetti, and L. Iocchi. A comparative analysis of particle filter based localization methods. In *Proc. of RoboCup Symposium*, 2006.

21. J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. of British Machine Vision Conference*, volume 1, pages 384–393, 2002.
22. K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, November 2005.
23. R. Muñoz Salinas, E. Aguirre, and M. García-Silvente. People detection and tracking using stereo vision and color. *Image Vision Comput.*, 25(6):995–1007, 2007.
24. R. Muñoz Salinas, R. Medina-Carnicer, F. J. Madrid-Cuevas, and A. Carmona-Poyato. People detection and tracking with multiple stereo cameras using particle filters. *J. Vis. Comun. Image Represent.*, 20(5):339–350, 2009.
25. H. Z. Ning, L. Wang, W. M. Hu, and T. N. Tan. Articulated model based people tracking using motion models. *in Proc. Int. Conf. Multi-Model Interfaces*, pages 115–120, 2002.
26. P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, pages 754–760, Washington, DC, USA, 1998. IEEE Computer Society.
27. J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *ICCV*, 2, 2003.
28. C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:246–252 Vol. 2, August 1999.
29. T. N. Tan, G. D. Sullivan, and K. D. Baker. Model-based localization and recognition of road vehicles. *International Journal Computer Vision*, 29(1):22–25, 1998.
30. T. Tian and C. Tomasi. Comparison of approaches to egomotion computation. *Computer Vision and Pattern Recognition*, pages 315–320, 1996.
31. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Intl. Conf. on Computer Vision*, pages 255–261, 1999.
32. A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computer Survey*, 38:13, 2006.
33. T. Zhao, M. Aggarwal, R. Kumar, and H. Sawhney. Real-time wide area multi-camera stereo tracking. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 976–983, Washington, DC, USA, 2005. IEEE Computer Society.