4.11

2D and 3D Motion Tracking in Digital Video

Georgios Stamou,
Michail Krinidis,
Evangelos Loutas,
Nikos Nikolaidis,
and Ioannis Pitas
Aristotle University of Thessaloniki, Greece

1 Introduction

The basic idea behind tracking objects in digital video is to derive object or even camera motion information. On the basis of the results derived for a video frame, we can predict (or estimate) rather than detect the object position and/or orientation. This is in contrast with frame-based video object detection, where the goal is to find the location of the object of interest in the scene without using its motion information. An obvious advantage of object tracking over object detection is that, in the case of multiple objects, the former can often provide automatic object labelling as they move over time. Additionally, one of the motivations behind object tracking is that object detection is frequently computationally slow or prone to detection errors. Even in the case that object detection is the final goal, tracking can significantly reduce the search region within a frame and, hence, the computations required. The output of an object tracking algorithm depends on the application and the representation used to describe the object that is being tracked. It can therefore be, for example, the contour of the object (a closed curve), the two-dimensional (2D) image coordinates of its center of mass, its three-dimensional (3D) position in world coordinates, the posture of the articulated object (i.e., the set of joint angles for articulated structures), and so forth.

Object tracking has received considerable attention in the past few years mainly due to the wide range of its potential applications. One important application domain is advanced human–machine interfaces, where tracking is closely coupled with human motion analysis and behavior understanding. Artificial vision has received increasing attention in the past years toward building effective human–machine interfaces, in a role complementary to speech recognition and natural language understanding [1]. In such a way, more intelligent and natural communication can be achieved between humans and machines. In this context, gesture recognition, body and face pose estimation, and facial expression analysis and recognition have been used in an effort to enable machines to interact more cleverly with their environment and their users. In all these applications, tracking constitutes an essential part of the overall process.

Another important application domain is smart surveillance, obviously due to the huge number of security-sensitive areas such as banks, department stores, parking lots, and so forth that need to be monitored efficiently. Such a "smart" system does not simply detect motion, which alone might lead to false alarms, but it can classify the motion (e.g., human or nonhuman motion) and perhaps perform face recognition and tracking for access control purposes or for wide field tracking across multiple cameras. Other systems proceed even further, performing human behavior analysis to signal suspicious behavior (e.g., in a car parking area, or in front of an automatic teller machine). In this way, the huge number of cameras already installed in security-sensitive areas could be used as a tool for efficient real-time automated or semiautomated surveillance (e.g., to alert security officers

or human operators). Furthermore, human operators could search archived video for specific activity patterns without actually viewing the video. The benefits of such systems, however, could be counterbalanced by possible drawbacks, such as privacy violations, since sensitive and complex sociologic issues are involved in surveillance and in computer interfaces.

Virtual reality and computer graphics have substantially benefited from motion tracking and analysis. To insert and animate an avatar in a virtual reality environment, one needs to "capture" the motion of the entity (e.g., a person) represented by the avatar in the real environment and use it to drive the avatar. Moreover, if one wishes to include an autonomous entity (e.g., a virtual dog), it would be best to capture its behavior and motion in the real world. Similarly, animating humanlike characters is often performed by synthesizing human movements based on motion data acquired off-line by tracking real humans, retrieving their body poses, and recognizing their gestures. A very powerful example of the successful application of the above can be found in the computer games and motion pictures industry (e.g., in movies involving both real and computer-generated actors), where the degree of realism of the virtual actor motion is surprisingly high. Training athletes and analyzing their performance can also benefit from vision-based tracking of human motion [2, 3]. Additionally, medical diagnosis and treatment support can be performed by gait analysis [4, 5].

Image coding is another important application domain. For example, when using a videophone, tracking the user's face in the video would make it possible to code it with more detail than the background and can result in more effective video storage and transmission. Object tracking is also involved in model-based coding. In the videophone application example, tracking can be used to derive pose and deformation parameters of a 3D head model at the transmitter side. These parameters are then sent to the receiver, which uses them to animate a similar model. This way, the amount of information to be transmitted is limited to the vector of animation parameters over time. Finally, content-based querying, indexing, and retrieval in multimedia databases can also benefit from advanced object-tracking techniques. For instance, motion path data obtained by tracking and analyzing the motion of players in sports video footage can be used for content-based indexing and retrieval of such data.

Video-based object tracking is just one of the many techniques devised for tracking objects. It belongs to the broader class of passive object tracking techniques that rely on measuring natural signals, such as light or sound [6]. Another broad category of tracking techniques is active object tracking, which involves placing devices (i.e., sensors) on the object and in the environment that transmit or receive appropriate signals, respectively [7]. Active object tracking techniques include [8, 9]:

- Mechanical trackers that are based on a kinematic structure (either serial or parallel), which consists of links interconnected with sensorized joints.
- Inertia trackers, which are devices that consist of gyroscopes and accelerators and measure the rate of change of the translation velocity and the angular velocity of an object.
- Ultrasonic trackers, where transmission and sensing of ultrasonic waves is used. The time taken for a brief ultrasonic pulse to travel from a stationary receiver placed in the environment to a receiver attached to the moving object is measured and used to identify the object position.
- Magnetic trackers, which are noncontact devices that use the magnetic field produced by a stationary transmitter to measure the real-time position of a receiver placed on the moving object.
- Radio and microwave trackers, where the time-of-flight of the corresponding type of waves from a stationary transmitter to a moving receiver on the object of interest is measured to determine the range of an object.
- Hybrid trackers, which use more than one of the above position measurement technologies to track objects more accurately than a single technology would allow.

The use of such sensors simplifies further processing. However, active trackers are "intrusive" and mainly suitable for well-controlled environments. Therefore, passive trackers are preferable (but more difficult to devise) than active ones.

Computer vision researchers have been trying to achieve results comparable to active object tracking using passive techniques for a long time in an effort to produce widely applicable motion tracking systems, free of intrusive devices, able to function in uncontrolled (indoor or outdoor) environments. In some cases, video-based object tracking using simple markers placed on the object of interest is used. Other systems employ light emitting diodes (LEDs). They are placed on the moving object with multiple stationary infrared or other cameras sensing the transmitted light or in the environment (e.g., the ceiling) with a camera placed on the moving object sensing the emitted light. However, these two variations can be considered as belonging to the category of active object-tracking technologies.

We limit our discussion to computer-vision–based object-tracking techniques that use passive sensing without markers. Various criteria can be used to provide a classification of object-tracking algorithms. In the rest of this section, a number of such classification schemes will be presented. The main features of object-tracking algorithms will be described along with various classification schemes.

A very important distinction, which is adopted in this chapter aiming at the coarse classification of the presented

techniques, is the dimensionality of the tracking space and the structure of the object to be tracked. According to the first criterion, the object-tracking algorithms are classified as 2D or 3D ones. 2D object-tracking aims at recovering the motion in the image plane of the projection of objects that in the general case move in the 3D space. 3D object tracking, on the other hand, attempts to estimate the actual 3D object movement using the 2D information conveyed by an image sequence. The structure of the object to be tracked is another characteristic that affects the type of motion that needs to be estimated. Therefore, rigid and deformable object trackings refer to estimating the motion of rigid and deformable objects, respectively, whereas articulated object tracking refers to estimating the motion of articulated objects [i.e., objects composed of rigid parts (links) connected by joints allowing rotational or translational motion in 1, 2, or 3 *df*]. In other words, articulated motion can be defined as piecewise rigid motion, where the rigid parts conform to the rigid motion constraints, but the overall motion is not rigid [10].

A different classification of object-tracking methods is based on their mode of operation. That is, tracking can either be performed online or offline. Trackers in the former category can use information about the object coming from one or more previous frames to predict its location in the current frame (i.e., information from future frames is not available). Trackers from the latter category can potentially make use of the entire image sequence, prior and posterior to the frame of interest. Offline object tracking can potentially provide better results since more information is available. However, future frames are often not available. Furthermore, using this extra information comes at the expense of increased computational load.

Tracking over time involves matching objects in consecutive frames using some kind of information. Essentially, object-tracking methods attempt to identify coherent relations of image information parameters (position, velocity, color, texture, shape, and so forth) between frames. Therefore, an alternative classification of object-tracking algorithms can be based on the type of information they use for matching. More specifically, object-tracking techniques can be classified to color-based, contour-based, feature-based, and template-based techniques. Another important characteristic of object-tracking algorithms is whether a model (geometric or other) of the object that needs to be tracked (e.g., a human body model when tracking people in video sequences) is used. Consequently, tracking algorithms can be classified to model-free or model-based ones. The decision to use a model, whether this model would be a 2D or a 3D one, as well as its complexity depends on the application. For example, in surveillance applications [11, 12], models of the object to be tracked are hardly necessary, since the parameters of interest involve only the presence and the spatial position of humans. In contrast, in a motion capture application aiming at obtaining data for the animation of a virtual actor, a detailed 3D face and body model is required. In general, 2D object-tracking models would usually consist of a number of image templates representing the class of objects that are to be tracked. Three-dimensional object tracking models can either be volumetric or surface ones [12].

Particular attention has been paid to motion models, especially for humans. On one hand, humans move in complex and rich patterns. On the other hand, many of the human activities involve highly structured motion patterns. Some motions are repetitive in nature (e.g., walking or running [13]), while others represent "cognitive" routines (e.g., crossing the street by checking for cars to the left and to the right). It is safe to assume that, if such models could be identified in the images, they would provide strong constraints for the tracking process, with image measurements being used to fine tune the estimates of the object motion parameters. However, this is not a trivial task. Human activities are often affected by unforeseen factors that are usually impossible to recover from image data. Also, several activities combine more than one motion models. Attempts have been made to tackle this problem (learning individual motion models, switching models, and so forth), but at the expense of computational complexity.

It is often the case that tracking objects in consecutive frames is supported by a prediction scheme. Based on information extracted from previous frames (and any high-level information that can be obtained), the state of the object (e.g., its location) is predicted and compared with the state of objects identified in the actual frame in question. Regardless of the type and number of parameters used to describe the object state, a model of its evolution in time is required. An excellent framework for prediction is the Kalman filter [14–16], which additionally estimates prediction errors. In complex scenes, however, it is most likely that deriving a single hypothesis for the next state of the object is impossible. For this reason, alternative prediction schemes have been devised that are capable of keeping track of multiple hypotheses, such as the well-known Condensation algorithm [17]. This leads to another possible taxonomy of object tracking algorithms (i.e., single-hypothesis vs. multiple-hypothesis trackers).

Devising a tracking algorithm that is capable of deriving object motion information under all possible conditions and environments is a very difficult task, since it requires tackling a number of difficult situations that include but are not limited to projection ambiguities, occlusion and self-occlusion, unconstrained motion, clutter, poor or varying lighting conditions, use of a single camera, the deformable clothing of humans (which produces variability in the body shape and appearance), and so forth. These difficulties have led researchers to adopt a number of assumptions to focus on tackling specific aspects of an overall, very complex problem. Assumptions can be either related to the motion of the camera or subjects (fixed camera, single-person scenes, occlusion-free scenes, known motion models, e.g., front-to-parallel

movement with respect to the camera, etc.) or refer to the appearance of the environment (constant lighting conditions, uniform or static background, etc.) or the subject(s) (tight clothing, tracking of specific type of objects, e.g., cars, etc.). Any of the above assumptions can provide constraints that can greatly facilitate the solution of the tracking problem. The constraints incorporated in an object-tracking algorithm can be used for its characterization. Therefore, tracking algorithms can be characterized on the basis of whether they focus on tracking specific objects, such as car tracking or tracking of human body parts (face, hand, etc.) or not, the number of views available (single-view, multiple-view, and omni-directional view tracking techniques), the state of the camera (moving vs. stationary), the tracking environment (indoors vs. outdoors), the number of tracked objects (single object, multiple objects, groups of objects), and so forth.

An important step prior to applying any tracking algorithm is proper initialization. This can be performed offline or online as the first stage of a tracking algorithm and aims at recovering information about the camera and/or the scene and/or the object to be tracked [6]. The first is most often dealt by offline camera calibration that identifies the intrinsic (focal length, radial distortion, etc.) and the extrinsic (scene geometry) camera parameters. This, however, requires a fixed-camera setup. If the camera setup changes, recalibration is necessary. Online (or self) calibration is also possible [18, 19]. Recovering information about the scene can be important for subsequent tracking. For example, in tracking algorithms where the initial object position in the image is found by background subtraction, the initialization step might include capturing the background reference images. Finally, the third initialization goal might include the initialization of the model used in a model-based object tracking algorithm or the estimation of the initial pose and position of the object. Obviously, the scope of the initialization step differs between different tracking algorithms. For example, in a 2D contour-based object-tracking method, the initialization could be an object detection step, aiming at finding the contour of the cobject in the first frame of the video sequence. If the method is a feature-based one, the initialization should provide the 2D coordinates of all the object features that are to be tracked.

Another important feature of a tracking algorithm is its ability to handle occlusion. It is usually distinguished in partial and total occlusion, where the object of interest is partially or totally occluded by another object (fixed or moving). Self-occlusion is also of particular interest. In this case, parts of the object are occluded by the object itself (e.g., limb occlusion when walking humans are being tracked or face occlusion caused by hand gestures during a conversation). Several object-tracking techniques assume no occlusion at all, whereas others provide occlusion handling mechanisms. One way of handling occlusion is by means of reinitialization [i.e., the initialization phase (e.g., object detection)] is applied again, usually at a computational expense that may affect the real-time capabilities of the algorithm. Alternatively, object position prediction schemes could be used for the whole duration of the occlusion.

This chapter aims at describing the basic principles behind 2D and 3D object-tracking algorithms and provides a basic literature overview of the subject. However, the literature is very rich. Therefore, certain categories of object-tracking methods, such as deformable object tracking (used for example in facial expression tracking) are not thoroughly covered due to space limitations. Additional information and further details about the topics described in this chapter can be found in the excellent reviews that have appeared in the literature [1, 6, 10, 20–23].

2 Rigid Object Tracking

2.1 Two-Dimensional Rigid Object Tracking

Two-dimensional rigid object tracking tries to determine the motion of one or more rigid objects in the image plane (i.e., its/their position over time). The image plane motion is induced by the relative motion between the viewing camera and the observed scene. A basic assumption behind 2D rigid motion tracking is that there is only one, rigid, relative motion between the camera and the observed scene [24]. This is the case of, for example, a moving car. This assumption rules out flexible objects (i.e., articulated objects) like a moving human body, or deformable objects like a piece of cloth.

Methods for 2D rigid object tracking can be classified in different categories according to the tools that are used in tracking:

- Region-based methods
- Contour-based methods
- Feature-based methods
- Template-based methods

Two-dimensional rigid object tracking methods constitute the basic building blocks for other categories of tracking algorithms. For example, an articulated object tracking algorithm may include a rigid object tracking module in order to track the rigid parts that make up the articulated structure. In the following, we will review the basic principles of the main categories of 2D rigid object tracking algorithms, describe the Bayesian framework frequently employed in object tracking algorithms and discuss the crucial topic of occlusion handling.

2.1.1 Region-based Object Tracking

It is usually an efficient way to interpret and analyze motion observed in a video sequence. An image "region" can be defined as a set of pixels having homogeneous characteristics. It can be derived by image segmentation, which can be based on distinctive object features (e.g., color, edges) and/or on the motion observed in the frames of a video sequence. Essentially, a "region" would be the image area covered by the

projection of the object of interest onto the image plane. Alternatively, a region can be the bounding box or the convex hull of the projected object under examination.

Color information proved to be very effective in region-based object tracking, because it enables fast processing while providing results robust enough to perform real-time tracking (i.e., 20–30 frames per second). Color segmentation is the core of color-based object-tracking algorithms. If the color of the object to be tracked can be modelled efficiently and distinguished from the color of other objects in the scene and the color of the background, it can be a very useful tracking cue, combined of course with an appropriate color similarity metric. The major problem with color segmentation and tracking is to provide for robustness again illumination changes. This can be achieved, for example, by controlling the illumination conditions, which is, of course, impossible in real-world environments especially for outdoor scenes. Alternatively, illumination invariance or color correction can be used. The former aims at representing color information in a way that is invariant to illumination changes, whereas the latter attempts to map the color responses of a camera obtained under unknown illumination conditions to illumination independent descriptors. For example, one common way to obtain illumination invariance is by using only the chromaticity values in a suitable color space (e.g., the chromaticity components H,S in the Hue-Saturation-Value space). Alternatively, one can normalize the color space. For example, in the Red Green Blue (RGB) space, this would mean that instead of using the R,G,B component values, one can use the values $\frac{R}{R+G+B}$, $\frac{G}{R+G+B}$ and $\frac{B}{R+G+B}$, respectively (which is the widely used normalized RGB space) or the values $\frac{R}{max\{G,B\}}$, $\frac{G}{max\{R,B\}}$, $\frac{B}{max\{R,G\}}$ [25]. For color constancy, a number of well-known algorithms, such as variants of the Grey World Algorithm [26], color by correlation [27], and so forth can be applied.

Many tracking algorithms focus on tracking humans (the whole body or body parts, such as hands, face, etc.). It becomes obvious that, in this case, the distinctive color of the human skin can serve as an appropriate means of locating and tracking people in video sequences (Fig. 1). A color segmentation algorithm can be devised in three steps, namely the choice of a suitable color space, the modelling of the skin (or any other object), color distribution over the selected color space, and the method used to classify the individual pixels to object (skin) and nonobject (nonskin) pixels. Methods for color modelling can be roughly classified as parametric (using a single Gaussian or a mixture of Gaussians) and nonparametric (histogram based, such as lookup tables and Bayes skin probability maps). The selection of an appropriate color space has been proven to be coupled to the model used for the skin color distribution. For parametric techniques, it was determined that spaces normalized with respect to illumination (e.g., normalized RGB) perform better with the single Gaussian model, while mixtures of Gaussians can produce comparable results when applied to spaces with no illumination normalization [28]. For nonparametric modelling methods, the HS color spaces (Hue Saturation Value (HSV), Hue Saturation Intensity (HSI), Hue Luminane Saturation (HLS), etc.), which are inherently related to the human perception of color, perform better when used in methods based on lookup tables. On the other hand, the choice of color space does not affect methods based on skin probability maps [29]. Experimental studies have determined that, for a specific camera, the skin color distribution forms a cluster (the so-called *skin locus*) in various color spaces. The authors in [30] argue that for every color space, there exists an optimum skin detector scheme, such that the performance of all these skin detector schemes is the same, thus rendering the choice of a specific color space irrelevant.

A simple scenario that allows robust color-based object tracking is chroma-keying. In this case, the background is single colored (most often blue) and the object colors are very different from the background (Fig. 2), or the object is single colored (e.g., a moving person that wears single-color clothes, most often dark-colored) and this specific color does not appear in the background. In both cases, simple color thresholding can be used to separate the object from the background and track it. The same technique can be used along with markers of distinctive color that have been placed on the object.

One of the most often-used approaches in region-based object tracking is color histograms [31–33], which fall in the

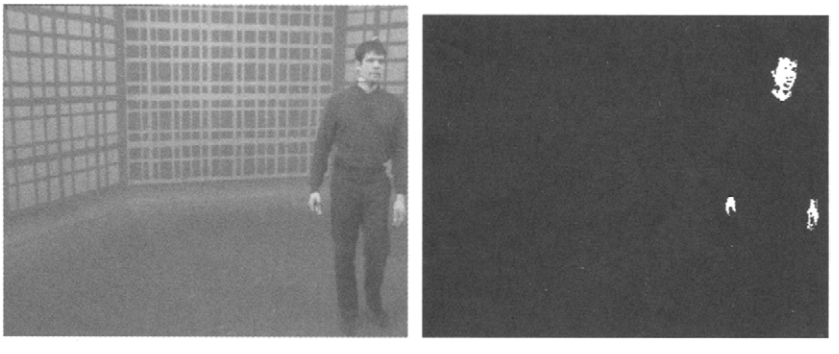

FIGURE 1 Skin detection based on simple thresholding in the Hue-Saturation components of the HSV color space. (See color section.)

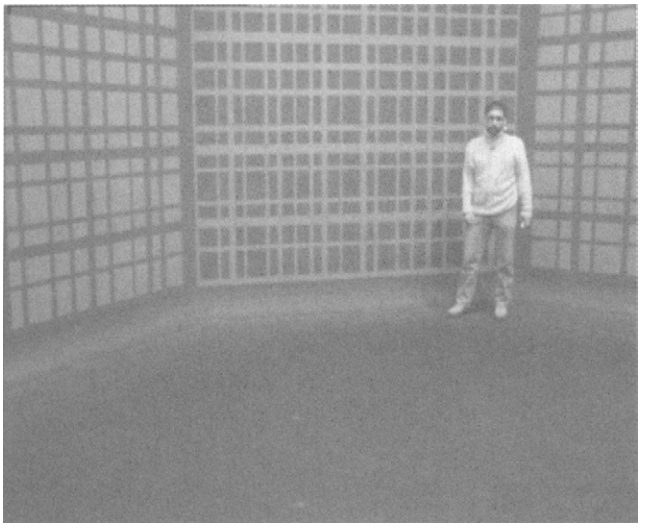

FIGURE 2 A typical environment for color-based object tracking using chroma keying. (See color section.)

category of nonparametric techniques. The color histogram of the ith region A_i in an image is denoted by $\mathbf{O}_i$. In the initialization step, color histograms of all the objects (i.e., regions) of interest in the scene are computed from a number of frames of a video sequence, stored in a database as reference color histograms, denoted by $\mathbf{O}_i^r$, and are used later in the matching process. In each new frame of the video sequence, for each of the tracked objects, a color histogram, denoted by $\mathbf{O}_i^t$, is calculated for every candidate object position. Each derived histogram, i.e., target histogram, is compared against the reference color histogram of the object in order to determine the best match and find the position of the tracked object in the current frame. Various criteria, which depend on the specific algorithm employed, are used to measure histogram distance or similarity. These include the histogram intersection measure [31], which performs a bin-by-bin comparison between two histograms and returns a relative match score based on the portion of pixels that are found in the same color bin of each histogram. It can be defined as:

$$\bigcap(\mathbf{O}_i^t, \mathbf{O}_i^r) = \sum_{n=1}^{U} min\{\mathbf{O}_{i,n}^t, \mathbf{O}_{i,n}^r\}, \tag{1}$$

where U is the number of bins in the histogram and n is the corresponding bin index. The sum of squared differences (SSD) can also be used for example,

$$\mathrm{SSD}(\mathbf{O}_i^t, \mathbf{O}_i^r) = \sum_{n=1}^{U} (\mathbf{O}_{i,n}^r - \mathbf{O}_{i,n}^t)^2. \tag{2}$$

It is obvious that the reference histograms described in Eq. 1 and Eq. 2 can handle only a fixed color distribution, thus not being able to account for changes in illumination that could potentially lead to tracking drift or failure. To overcome this, the number of bins of the reference histograms as well as the bin content can be dynamically updated at regular intervals during the tracking process, by using color information from the frames of the video sequence. This, however, comes at the expense of computational burden. The selected color quantization level can affect the result (i.e., the accuracy of determining the object position in the frame under examination). Histograms have been shown to be effective only when the number of bins is neither too low nor too high.

For the parametric techniques [34–39] which use parametric color reference models, the operating principle is the following: The current frame is searched for a region, namely a window of variable size but fixed shape, whose color content best matches a reference color model, e.g., a mixture of Gaussians [39]. More specifically, in this case, the color distribution of the object is considered as multimodal and, as such, is approximated by a number of Gaussian functions in some color space, e.g., the Hue-Saturation color space. Starting from the object location in the previous frame, the method proceeds iteratively at each frame so as to minimize a distance measure to the reference color model. Since object color can often change due to the illumination conditions, (i.e., the same object can be perceived as having two different colors when such conditions change abruptly), the model is adapted to reflect the changing appearance of the tracked object. A statistical approach is used in which color distributions are estimated over time by sampling from the object pixels to obtain a new pixel set that is used to update the Gaussian mixture model.

An example of an easily implemented region-based object-tracking algorithm is the one introduced in [40]. The algorithm tracks colored regions from frame to frame. To cope with color changes due to illumination changes, the color space used is normalized. The object to be tracked is divided into L image regions $R_1, \ldots, R_L$. The regions are assumed to be fixed with respect to size and relative position. When initializing the algorithm, every region R_i is assigned a reference color vector $\mathbf{g}_i$, which represents the averaged color of pixels within the region. A color vector $\mathbf{g}_i$ is also computed in a similar manner for each region in every frame of the video sequence. The reference and the computed color vectors are compared using a goodness-of-fit criterion. For each of the three color components (RGB), the ratio of the component values in the computed and the reference color vectors (e.g., $\frac{r_i}{\bar{r}_i}$ for the red component) is computed. The goodness-of-fit criterion is then chosen as the ratio of the maximum of these three ratios to the minimum of the ratios. Values close to 1 correspond to good matches.

A region-based tracking system that uses color as a tracking cue is Pfinder, which was presented in [41]. The system is capable of tracking a single person in scenes with complex background, captured by a fixed camera. It uses blob representations (i.e., coherent connected regions where pixels

have similar image properties). For each pixel in a blob, its spatial coordinates (x, y), along with its textural (color) components are used to form a feature vector. The corresponding spatial and color distributions are assumed to be independent. The statistics of each blob are updated with the new information coming from the recently acquired images. In an initialization step, the algorithm builds a model for the scene without any person in it. When a person enters the scene, large changes in the scene are detected and used to build a model for the person. Since the empty scene model is known, the feature vectors at each pixel are compared with the feature vectors of the scene model using the Mahalanobis distance in the YUV color space. The person is localized based on the detected changes.

Another popular region-based object tracking approach is based on background subtraction [42–45]. In an initialization step, the model of the scene is built without the presence of any moving object. It is then assumed that the background remains static during the acquisition of the video sequence. Each new frame is subtracted from the scene model to segment any foreground (i.e., moving) objects. However, the subtraction itself is not sufficient to obtain clear information due to the noisy measurements captured by the camera, as well as changes in the scene environment (e.g., illumination changes). This problem, which is common in outdoors video sequences, is tackled by using e.g., a suitable combination of morphologic operations (i.e., dilations and erosions) [46]. The basic idea is that pixels that have been erroneously assigned to the foreground (i.e., outliers) can be eliminated by erosion, while a combination of dilations and erosions can smooth the image regions corresponding to the foreground objects. The scene model can also be dynamically updated, as in [46]. In this paper, when there are no foreground objects present in the scene, the scene model is updated in order to encompass any new information that the new frames might contain.

2.1.2 Contour-based Object Tracking

An alternative way of devising an object tracking algorithm is by modelling an object using outline contour information and tracking it over time, thus retrieving both the position and the shape of the projected object. Such a modelling method is more complicated than modelling entire regions (e.g., using color). However, the tracking algorithm is more robust than region-based object tracking algorithms, since it can be adapted to cope with partial occlusions. Contour tracking has found numerous applications in surveillance, medical diagnosis, and audiovisual speech recognition. Apart from tracking rigid objects, contour-based tracking can also be used for tracking deformable objects.

Active contours, also known as "snakes" have been extensively used by researchers to perform contour delineation and tracking. An active contour algorithm deforms a contour to "lock" onto features, which may be lines, edges, boundaries, and so forth. They were first introduced in [47], where the authors tried to perform robust segmentation and region tracking by modelling an object using outline information, since the latter is relatively insensitive to illumination variations. Snakes consist of an elastic parametric curve that can be dynamically deformed to match object shapes. The deformation is subject to internal forces (contour elastic forces) and external forces (due to image content and other constraints). More formally, an active contour is a collection of n points in the image plane that define a polygonal line:

$$\Psi(\gamma) = \psi(\gamma)_1, \ldots, \psi(\gamma)_n, \tag{3}$$

$$\psi(\gamma)_i = (x(\gamma)_i, y(\gamma)_i), i = 1, \ldots, n. \tag{4}$$

Since active contour models are a special instance of deformable models [47], we can also define them over a space of allowed contours by using a functional to be minimized. The latter represents the energy of the model and, as already mentioned, consists of two terms:

$$E_{total}(\Psi(\gamma)) = E_{int}(\Psi(\gamma)) + E_{ext}(\Psi(\gamma)), \tag{5}$$

where E_{int} is an energy function dependent on the shape of the contour and E_{ext} is an energy function that depends on the image features of interest and other user-defined constraints. They are defined by:

$$E_{int} = \int_0^1 (\phi_1 \|\Psi'(\gamma)\|^2 + \phi_2 \|\Psi''(\gamma)\|^2) d\gamma \tag{6}$$

and

$$E_{ext} = \int_0^1 F(\Psi(\gamma)) d\gamma, \tag{7}$$

where ϕ_1 and ϕ_2 are constant or dynamically changing parameters associated with the internal energy that control the behavior of the snake. More specifically, ϕ_1 controls the "tension" of the contour (i.e., its ability to resist to stretch), whereas ϕ_2 controls the "stiffness" of the contour (the flexibility and the smoothness of the snake). Also, F in (7) is the potential related to the external forces applied to the snake. It depends on the object features that we are interested in. For example, if edges are the features of interest in gray scale images, then it can be defined as in [47]:

$$F = -\|\nabla I\|^2, \tag{8}$$

where I is the intensity of the image.

The points on the active contour iteratively approach the boundary of an object through the solution of the energy minimization problem (5). Many object tracking algorithms use this concept, e.g., [48]. When the internal and the external forces applied to the snake counterbalance each other, the snake is said to have reached its equilibrium state. The equation describing the equilibrium state is:

$$\left(\frac{\partial(\phi_1\Psi)}{\partial\gamma} - \frac{\partial^2(\phi_2\Psi)}{\partial\gamma^2}\right) + \nabla F = 0. \quad (9)$$

A snake can be made to represent specific image features by a suitable choice of F and the internal shape parameters ϕ_1, ϕ_2. To perform contour tracking, the energy minimization process (5) is repeated for successive frames. Since tracking based on snakes is sensitive to initialization (i.e., the snake needs to be initially placed close to the contour that needs to be tracked, otherwise it will fail), a temporal prediction module capable of estimating the position of the object outline in the next frame is often used in conjunction with the snake algorithm, thus achieving, at the same time, a reduction in the computational complexity of the snake algorithm.

A robust algorithm for tracking the visible boundary of an object in the presence of occlusion is presented in [48]. First, an initial outline of the object contour is specified by the user and is automatically refined by using intraenergy terms. Then, a number of node points are selected along the contour to define the initial snake. Afterward, the snake is segmented into nonoverlapping pieces by selecting a set of feature nodes whose curvature or color variation exceeds a predefined threshold. For each contour segment, two predicted locations are obtained, one based on local motion vectors within the object and one based on motion vectors on the background side of the contour. The next location is the one that results in the smaller prediction energy. Finally, the predicted contour is refined using inter-frame and intra-frame energy terms.

Wang et al. [49] use adaptive dynamic contours that can track 2D objects outlines in image sequences. Objects are modelled as a curve (or a set of curves) and represented at time t by an image curve $\chi(\mu, t)$ parameterized in terms of B-splines defined by a number of control points,

$$\chi(\mu, t) = (\Theta(\mu)\cdot\Gamma^x(t), \Theta(\mu)\cdot\Gamma^y(t), \text{ for } 0 \le \mu \le L), \quad (10)$$

where $\Theta(\mu)$ is a vector $[B_0(\mu), \ldots, B_{N_B-1}(\mu)]^T$ of B-spline basis functions, Γ^x and Γ^y vectors of control points coordinates and L is the number of spans. The positions of control points are continuously adjusted to have them evenly distributed and prevent them from crashing together or leaving far away from one another. Moreover, the control points are not allowed to cross over during the tracking process. Temporal prediction is based on the movement of the centroid of the object and is performed using a simple prediction filter. Another approach, based on B-splines, is presented by Bascle et al. [50] who integrate the ideas of snake-based contour tracking and region-based motion analysis. They use a snake to track the region outline and perform segmentation. Afterward, the motion of the extracted region is estimated by a dense analysis of the apparent motion over the region, using spatio-temporal image gradients.

A different approach is presented in [51]. This algorithm uses graph cuts based on active contours to track object contours in video sequences. The minimum cut of a graph G that separates a source set $\{s_1, s_2, \ldots, s_n\}$ and a sink set $\{t_1, t_2, \ldots, t_m\}$ is exactly the $s-t$ minimum cut of the resulting graph after identifying $s_1, s_2, \ldots, s_n$ to a new source s and identifying $t_1, t_2, \ldots, t_m$ to a new sink t [52]. The algorithm separates the area of interest into an inner and an outer boundary, presents the information contained in the region of interest by an adjacency graph, identifies all the nodes on the inner boundary as a single source s and all the nodes on the outer boundary as a single sink t. It then proceeds to compute the $s-t$ minimum cut and repeats the whole procedure until the algorithm converges, i.e., given an initial boundary near the object under consideration, the method can iteratively deform to match the desired object boundary. The result in each frame is used to initialize the contour in the next frame. It uses both the intensity information within the current frame and the intensity difference between the current and the previous frame to find the next position of the object contour.

2.1.3 Feature-based Object Tracking

Feature-based object tracking can be defined as the attempt to recover the motion parameters of a feature point in a video sequence, more specifically the parameters associated with the planar translation of a point, since points in 2D space neither rotate or translate with respect to depth. More formally, let $A = A_0, A_1, \ldots, A_{j-1}$ denote the j images of a video sequence and $\mathbf{m}_i(x_i, y_i)$, $i = 0 \ldots j-1$ denote a feature point in those frames. The task at hand is to determine a motion vector $\mathbf{d}_i(d_{x,i}, d_{y,i})$ that best determines the position of the feature point in the next frame, $\mathbf{m}_j(x_i, y_i)$, that is: $\mathbf{m}_j = \mathbf{m}_{j-1} + \mathbf{d}_j$. The object to be tracked is usually defined by the bounding box or the convex hull of the tracked feature points.

Feature-based object tracking, although more prone to individual outliers and tracking drift, can be implemented very efficiently [53]. However, most of the methods are sensitive to partial occlusions. An important problem in feature-based tracking is to determine and consequently track salient feature points in an image. These points can be pixels with high curvature information, i.e., corners, edges, and so forth. It is well known that such points play a dominant role in shape perception by humans [54]. Generally, the feature-based object-tracking methods select pixels that have a distinctive characteristic with respect to their neighbors, such as

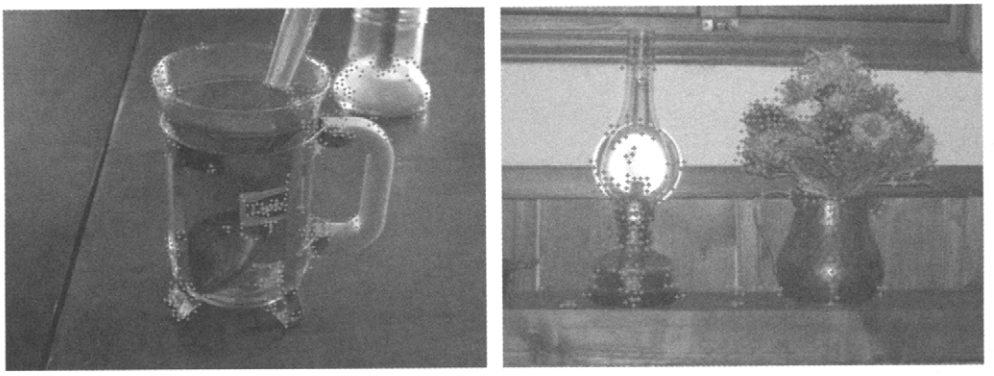

FIGURE 3 Feature selection based on [57]. (See color insert.)

brightness (most often), contrast, and so forth. Alternatively, information originating from the type of object that is to be tracked can be exploited, as in [55], where a combination of the adaptive Hough transform, the block matching algorithm, and active contours is used to extract salient facial features.

Many feature-based object tracking approaches exist [56–69]. Shi and Tomasi [57] proposed a method for finding good features to track. For each candidate feature point, the following 2×2 matrix is constructed:

$$\mathbf{Z} = \begin{bmatrix} \sum_W J_x^2 & \sum_W J_x J_y \\ \sum_W J_x J_y & \sum_W J_y^2 \end{bmatrix}, \tag{11}$$

where J_x and J_y are the image gradients evaluated on the point under consideration in the x and y direction respectively and W is a $n \times n$ window centered on the candidate feature point. A good feature is defined to be a point where the minimum eigenvalue of its matrix $\mathbf{Z}$ is larger than a predefined threshold. Such features represent corners or salt-and-pepper textures (Fig. 3). To measure the quality of image features during tracking and make sure that the same features are tracked throughout the video sequence, the authors used a measure of feature dissimilarity that quantifies the change of appearance of a feature between the first and the current frame.

The feature selection process is tightly coupled with the proposed object tracking algorithm, described in [56]. In this algorithm, the displacement of a feature point is chosen to minimize the dissimilarity defined by the following double integral over the given window W, centered at the pixel under consideration:

$$\epsilon = \int\int_W [A_i(\mathbf{m} - \mathbf{d}) - \mathbf{A}_i + \mathbf{1}(\mathbf{m})]^2 \omega(\mathbf{m}) d\mathbf{m}, \tag{12}$$

where A_i and A_{i+1} are two successive frames, $\mathbf{d}$ is the displacement vector and $\omega(\mathbf{m})$ is a weighting function (in the simplest case it is set equal to 1). As it was firstly introduced in [70], when the interframe motion is sufficiently small, the displacement vector can be written approximately as the solution to a 2×2 linear system of equations:

$$\mathbf{Zd} = \mathbf{e}, \tag{13}$$

where

$$\mathbf{e} = 2 \int\int_W [A_i(\mathbf{m}) - A_{i+1}(\mathbf{m})] g(\mathbf{m}) \omega(\mathbf{m}) d\mathbf{m}. \tag{14}$$

$\mathbf{Z}$ is the matrix (11), $\omega(\mathbf{m})$ is a weighting function and

$$g(\mathbf{m}) = \begin{bmatrix} \frac{\partial(A_i(\mathbf{m}) + A_{i+1}(\mathbf{m}))}{\partial x} \\ \frac{\partial(A_i(\mathbf{m}) + A_{i+1}(\mathbf{m}))}{\partial y} \end{bmatrix}. \tag{15}$$

A cross-correlation method was used in [68] to track feature points from frame to frame. Assuming relatively small displacements between adjacent frames (up to 10 pixels), the tracking procedure follows the method described in [71]. The algorithm begins with some feature points selected on the initial frame. Every n frames new feature points are selected to maintain the overall point number. For each feature point, the algorithm searches for the best candidate in the square neighborhood of the point position under consideration in the previous frame (search window). The criterion that is used is the maximum cross-correlation of square neighborhoods in the current frame. Consequently, a refinement of the position of each feature point is performed, based on the fact that all features belong to the same rigid object and, thus, they should move in a consistent way.

In [67], feature selection is based on Gabor wavelets, because they exhibit a number of desirable properties [72]. The convolution of an image A with Gabor wavelets leads to the Gabor wavelet transform of the image:

$$\hat{A}(x, y, \iota, \tau) = \int A(x', y') g_{\iota,\tau}(x - x', y - y') dx' dy', \tag{16}$$

where $g_{\iota,\tau}(x,y)$ is a 2D Gabor function:

$$g_{\iota,\tau}(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}+2\pi jWx\right)\right], \quad (17)$$

where σ_x and σ_y are the standard derivations of $g(x,y)$ along the corresponding directions. One can obtain Gabor wavelets by scaling and rotating (17), essentially using Gabor functions with different frequency centers and orientations. If we use S' different frequencies and T different orientations, each image pixel will be associated with $S' \times T$ coefficients, the amplitudes of which form a feature vector at the specific pixel. The energy of the coefficients is used in [67] to determine the discriminating power of each pixel and select the pixels to be tracked. A mesh is then created using the selected feature points. To track the object in the next frame of a video sequence, the authors use a 2D golden section algorithm that calculates the best possible translation of the center of the formed mesh. To allow for deformation, they also allow the nodes of the mesh to perturbate locally. In both cases, appropriately defined similarity functions are used to determine the best possible match.

2.1.4 Template-based Object Tracking

Template-matching techniques are used by many researchers to perform 2D object tracking. They follow the same principles with the template matching techniques used in object recognition. The first step (initialization step) is to select the template that will be used (i.e., to essentially create an image model of the object to be tracked). Such models can be acquired before tracking in a number of ways. First, a template that is specific for a particular instance of a class of objects can be created. For example, in a face tracking module used in a videophone application, the user might be asked to face the camera for a period of time, enabling the system to detect his or her face and use the corresponding image region as a template. A template can be alternatively created offline by using statistical methods. For example, in a face tracking application, a generic face template can be created by incorporating information from the various existing face databases, (e.g., by evaluating an "average" face). Such a template can be obtained, for example, by the use of eigenfaces [73] which have been extensively used in face recognition, verification, and tracking tasks [74]. Eigenfaces are essentially the eigenvectors of the high-dimensional vector space of possible faces of human beings. To generate a set of eigenfaces, a large set of human face images are normalized (i.e., the eyes and mouth are aligned), resampled at the same pixel resolution (e.g., $q_1 \times q_2$ pixels), and then treated as q_1q_2-dimensional vectors whose components are their pixel intensities. The eigenvectors of the covariance matrix of the face image vectors are then extracted. Since the eigenvectors belong to the same vector space as the face images, they can be considered as $q_1 \times q_2$ pixel face images (called eigenfaces). When properly weighted, they can be averaged together into a gray-scale rendering of a human face, as shown in Fig. 4. Instead of bearing intensity or color values, the pixels of a template can be assigned feature vectors that consist of values obtained through an image processing operation (e.g., morphologic operations).

Template matching can be defined as the process of searching the target image (i.e., current frame of the video sequence) to determine the image region that resembles the template, based on a similarity or distance measure. Essentially, the template region should undergo a geometric coordinate transformation that would "place" the template onto the target image in such a way as to minimize the distance measure used. The goal of a template matching algorithm then becomes to estimate the parameters of such a transformation. More formally, let $\Upsilon(t)$ be the image region corresponding to the object being tracked at time step t. For a rigid object, $\Upsilon(t)$ can be obtained from the template region, denoted by Υ_0, by using a coordinate transformation

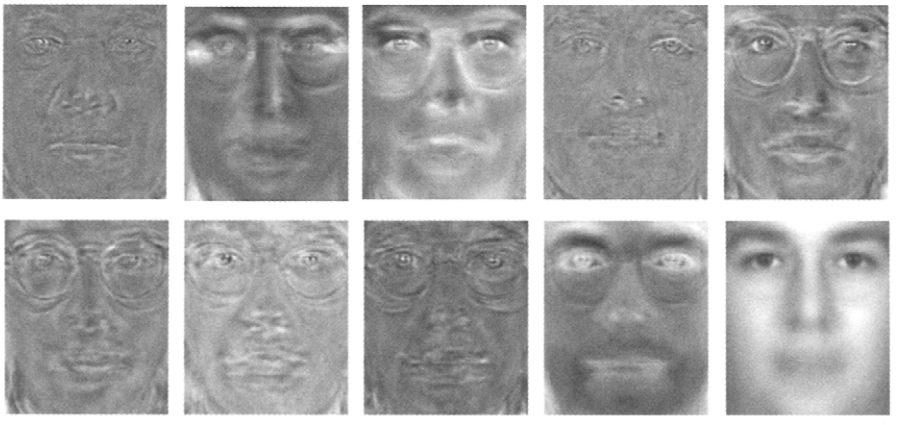

FIGURE 4 Sample eigenfaces computed from images of the M2VTS database [75] and the corresponding "average" face (lower right corner).

$\theta(\Upsilon_0) \Rightarrow \Upsilon(t)$ the parameters of which should be estimated by the algorithm. Affine (rotation, translation, scaling) [76, 77] or quadratic transformations can be used. Therefore, every point $\mathbf{m}(x, y)$ in the target region is obtained from a corresponding point $\xi(\xi_x, \xi_y)$ in the template region:

$$\mathbf{p} = \theta(\xi; \mathbf{a}(t)), \tag{18}$$

where $\mathbf{a}(t)$ denotes the transformation parameter vector associated with $\Upsilon(t)$. The object location in the current frame is determined by the vector $\mathbf{a}(t)$. Estimation of the transformation parameters is performed by identifying the image region that best matches the template. To avoid exhaustive search for the best match in the frame under examination, various techniques can be used. Background subtraction can be employed to determine image regions where motion activity appears and limit the search in these regions. Alternatively, prediction schemes like Kalman filters can be used to estimate the location of the object being tracked in the next frame and use it as the center of a limited-size search region. Finally, prior knowledge of the scene and other constraints (e.g., constraints on the expected object displacement between consecutive frames) can restrict the search area.

Many similarity/distance metrics have been used in the template matching step. More specifically, if T_i is the brightness of the ith pixel (x_i, y_i) in the template, $I_{i,v}$ is the brightness of the ith pixel (x_i, y_i) in the image region v and M is the number of pixels in the template, this can be performed by finding the image region v that minimizes one of the following distance metrics:

- The Hamming distance (i.e., the number of "different" pixels in the template and the image region),
- The SAD,

$$\sum_{i=1}^{M} |(I_{i,v} - T_i|, \tag{19}$$

- The SSD,

$$\sum_{i=1}^{M} (I_{i,v} - T_i)^2, \tag{20}$$

or maximizes one of the following similarity criteria:

- The normalized correlation,

$$\frac{\sum_{i=1}^{M} (T_i - \bar{T})(I_{i,v} - \bar{I}_v)}{\sqrt{\sum_{i=1}^{M} (T_i - \bar{T})^2 \sum_{i=1}^{M} (I_{i,v} - \bar{I}_v)^2}}, \tag{21}$$

where $\bar{T}$ is the mean brightness of the template, and $\bar{I}_v$ is the mean brightness of the image region v.

- The joint entropy,

$$-\sum_{i=1}^{M} I_{i,v} \ln\left(\frac{I_{i,v}}{T_i}\right), \tag{22}$$

- The mutual information, which can be expressed as:

$$H(I_v) + H(T) - H(I_v, T), \tag{23}$$

where

$$H(I_v) = -\sum_{i=1}^{M} I_{i,v} \ln(I_{i,v})$$

is the entropy of the image region I_v, $H(T)$ is the entropy of the template T and $H(I_v, T)$ is the joint entropy (22),

- The maximum likelihood [78].

Up to this point, it was assumed that template tracking is performed solely on rigid objects. However, due to the nonrigid nature of many natural objects or due to viewpoint changes, template tracking, as described so far, fails to provide satisfactory results in a number of real world scenarios. For that reason, deformable template tracking methods have been introduced [79–81]. In these methods, prior knowledge of the object shape is used in an energy minimization scheme. Deformable templates are specified by a set of parameters which enable a priori knowledge about the expected shape of the object to guide the template matching process. The deformable templates interact with the image in a dynamic manner. An energy function is defined for the template, consisting of terms attracting the template to salient features (intensity, edges). The template parameters are obtained by a minimization of the energy function. In essence, the deformable template is obtained by allowing an original template to deform using any appropriate deformation function. The result should cover the various instances of the deformable object as much as possible, with minimum computational overhead while maintaining the attributes of the template (smoothness, connectivity, etc.).

An easily implemented template-based face tracking technique is described in [82]. The face template is acquired in a training step where the user is asked to look at the camera, rendering the method useful only for tracking the face of the person it was trained for. Multiple people cannot be tracked. The template is a facial image covering the eyebrows, the outer left and right edges of the eyes, and the bottom of the mouth. A match function that finds the minimum sum of absolute differences between the value of the red color component of each pixel in the image region and the face template is used for template matching. To improve the performance of the technique, a simple facial model is used along with some assumptions about the environment.

A template-based object-tracking algorithm that uses color invariant features (independent of the viewpoint, surface orientation, illumination direction, illumination intensity and highlights) at each pixel position is presented in [77]. To account for sudden changes in illumination, the template is dynamically updated by means of robust filters like Kalman or particle filters. The filters are also used to estimate the position of the pixels comprising the object being tracked, as well as to handle short-time and partial occlusions. The distance metric used to compare the template and the candidate image regions is the Mahalanobis distance.

Template correlation is used in [83]. There are two template sizes available but both sizes can be magnified by a scale factor in the xth and yth direction independently. The template is matched within specified search windows using the mean absolute error between the image regions and the template.

A deformable template-based tracking technique is introduced in [84]. A hand-drawn prototype template describing the representative contour/edges of the object to be tracked is used. The original template undergoes deformation transformations to obtain a deformable template. The template matching process consists of comparing the template with the candidate image regions with respect to shape similarity (an object should have a similar shape in two successive frames), region similarity (properties like color and texture of a region should remain constant throughout a video sequence), interframe motion (the object outline should be attracted to pixels exhibiting large motion) and image gradients (the object outline pixels should have large image gradients). The template is dynamically updated in shape and size from the newly detected object.

2.1.5 Bayesian Object Tracking

Many object-tracking techniques perform tracking within a Bayesian framework. The latter belongs to the class of state space approaches that attempt to estimate the state of a system over discrete time steps, assuming that noisy measurements are available at these time steps. The state vector contains all data required to describe the system. For example, when tracking a moving object in two dimensions, the state vector would typically consist of the object position (x and y coordinates), as well as its velocity and acceleration along each coordinate (i.e., it would be a 6-dimensional vector). The measurement vector contains observations corrupted by noise that are related to the state vector. Its dimension is usually smaller than that of the state vector. In the previous example, the measurement vector would contain "noisy" object positions (x and y coordinates, as measured in the image).

To perform the estimation, one needs a system model that describes the evolution of the object state over time, for example:

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \tag{24}$$

and a measurement model that links the noisy measurements to the state vector:

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \mathbf{v}_k), \tag{25}$$

where $f_k : \Re^{n_x} \times \Re^{n_w} \to \Re^{n_x}$ and $h_k : \Re^{n_x} \times \Re^{n_v} \to \Re^{n_z}$ are possibly nonlinear functions, $\mathbf{w}_k$ and $\mathbf{v}_k$ are sequences that represent the independantly and identically distributed (i.i.d.) process noise and measurement noise respectively, n_x, n_w, n_z and n_v denote the size of the state, process noise, measurement and measurement noise vectors respectively.

To be able to perform object tracking in a Bayesian framework, two remarks should be taken into consideration. First, the system and measurement models should be available in a probabilistic form. Second, in object tracking, an estimate of the object position is required every time a new measurement becomes available. Hence, estimation can be performed recursively. Bayesian object tracking belongs to the class of online methods, i.e., it produces an estimate at each time step k based only on all past measurements $\mathbf{z}_k$ up to time k. Assuming, that the initial pdf of the state vector $p(\mathbf{x}_0|\mathbf{z}_0)$ is known ($\mathbf{z}_0$ is the set containing no measurements), the goal is to obtain the posterior pdf $p(\mathbf{x}_k|\mathbf{z}_k)$ at time step k. More specifically, let $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ be the system model defined by (24) and the process noise $\mathbf{w}_k$ statistics in probabilistic form and $p(\mathbf{z}_k|\mathbf{x}_k)$ the measurement model (also known as the likelihood function) defined by (25) and the measurement noise $\mathbf{v}_k$ statistics. The estimation process comprises two steps. During the first step (prediction), the posterior pdf at time step $k-1$, i.e., $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})$, is propagated forward in time, using the system model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ [85]:

$$p(\mathbf{x}_k|\mathbf{z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})d\mathbf{x}_{k-1}, \tag{26}$$

thus, obtaining the prior pdf $p(\mathbf{x}_k|\mathbf{z}_{k-1})$ at time step k. The second step (update) modifies the propagated pdf by exploiting the latest measurement available. Thus, the desired posterior probability density function (pdf,) $p(\mathbf{x}_k|\mathbf{z}_k)$, can be obtained by using Bayes theorem:

$$p(\mathbf{x}_k|\mathbf{z}_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1})}{p(\mathbf{z}_k|\mathbf{z}_{k-1})}, \tag{27}$$

where $p(\mathbf{z}_k|\mathbf{z}_{k-1})$ is used for normalization and calculated as follows [85]:

$$p(\mathbf{z}_k|\mathbf{z}_{k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1})d\mathbf{x}_k. \tag{28}$$

The optimum solution in the Bayesian sense can be obtained based on (26) and (27) [85]. Analytic forms of the solution can be obtained either when certain assumptions

hold, as in the case of standard Kalman filters or by approximations, as in the case of extended Kalman filters (EKFs) and particle filters (PFs) [17, 86, 87].

2.1.5.1 Kalman Filters and Extended Kalman Filters

The Kalman filter is a special case of the Bayesian filters mentioned earlier and is the best possible estimator, if the posterior pdf is Gaussian and the following conditions hold:

- Functions f and h in (24) and (25) are linear and known.
- The distributions of the process and measurement noises are again Gaussian.

If we wish to provide a formal definition, the Kalman filter is a set of mathematic equations that provides a computationally efficient, recursive solution to the least-squares method [14–16]. Let us assume that the process and measurement noise, denoted by $\mathbf{w}_k$ and $\mathbf{v}_k$, respectively, are independent, white ones with normal probability distributions:

$$p(\mathbf{w}_k) \sim N(0, \mathbf{Q}_k), \tag{29}$$

$$p(\mathbf{v}_k) \sim N(0, \mathbf{R}_k), \tag{30}$$

where $\mathbf{Q}_k$ and $\mathbf{R}_k$ are the process and measurement noise covariance matrices respectively, which can be constant or dynamically changing. Linear stochastic difference equations describe both the system model, i.e., the evolution of the object state over time, and the measurement model $\mathbf{z} \in \Re^m$:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_{k-1} + \mathbf{w}_{k-1}, \tag{31}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k, \tag{32}$$

where $\mathbf{u}_k \in \Re^l$ is an optional control input. We can observe that to propagate the state forward in time, the $n \times n$ matrix $\mathbf{A}_k$ must be defined. If there is an optional control input, then, the $n \times 1$ matrix $\mathbf{B}_k$ must also be defined to relate it to the state $\mathbf{x}_k$. Finally, the state and the measurement vectors are linked with the $m \times n$ matrix $\mathbf{H}_k$ in the measurement equation (32). All three matrices can be either constant or dynamically changing.

The Kalman filter operates in a two-step predictor-corrector manner. During the first step, the current estimate along with an estimate of the error covariance are propagated forward in time. The second stage incorporates a new measurement to modify the propagated current state and error covariance estimates. Let $\hat{\mathbf{x}}_k^- \in \Re^{n_x}$ denote the a priori state estimate at step k (based on all past measurements prior to step k) and $\hat{\mathbf{x}}_k \in \Re^{n_x}$ denote the a posteriori state estimate at step k (as soon as measurement $\mathbf{z}_k$ becomes available). The errors of the two state estimates can be respectively defined as [15]:

$$\mathbf{e}_k^- \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k^-$$

and

$$\mathbf{e}_k \equiv \mathbf{x}_k - \hat{\mathbf{x}}_k.$$

Then, the a priori and a posteriori error covariances can be respectively defined as:

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] \tag{33}$$

and

$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T]. \tag{34}$$

The equations of the first step (prediction) are [14, 15]:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_k \hat{\mathbf{x}_{k-1}} + \mathbf{B}_k \mathbf{u}_{k-1}, \tag{35}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^T + \mathbf{Q}_{k-1}. \tag{36}$$

The update stage begins by computing the so-called "gain" of the Kalman filter, denoted by $\mathbf{K}_k$. It is chosen to minimize the a posteriori error covariance (34). One popular form that performs this minimization is [15]:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}. \tag{37}$$

We can observe from (37) that, as $\mathbf{R}_k$ approaches zero, the actual measurement $\mathbf{z}_k$ becomes more trustworthy, whereas the predicted measurement $\mathbf{H}_k \hat{\mathbf{x}}_k^-$ becomes less trustworthy. The opposite occurs when the a priori estimate error covariance $\mathbf{P}_k^-$ approaches zero.

The gain is used, along with a measurement $\mathbf{z}_k$ (when it becomes available) to modify the a priori estimate $\hat{\mathbf{x}}_k^-$, so that the a posteriori state estimate $\hat{\mathbf{x}}_k$ can be computed:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-). \tag{38}$$

The difference in parentheses in (38) is called the measurement innovation, or the residual and it reflects the error between the predicted measurement $\mathbf{H}_k \hat{\mathbf{x}}_k^-$ and the actual measurement $\mathbf{z}_k$. The gain is also used to modify the a priori error covariance and obtain an estimate of the a posteriori error covariance:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-. \tag{39}$$

The two steps are repeated using the previous a posteriori estimates to predict the new a priori estimates. In the following, we will provide an example of applying Kalman filtering for object tracking by providing the description of an existing method.

In [88], a Kalman filter is used to perform pupil tracking for subsequently monitoring eyelid movements, determining gaze, and estimating face orientation. The state vector at time k is defined as $\mathbf{x}_k = [x_k, y_k, v_{x_k}, v_{y_k}]^T$, where x_k and y_k denote the coordinates of the pupil's centroid pixel position and v_{x_k} and v_{y_k} denote its velocity in the x and y directions respectively, at time step k. The system model equation assumes no optional control input $\mathbf{u}_k$:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{w}_{k-1}.$$

Assuming that the interframe pupil movements are small, the state transition matrix $\mathbf{A}_k$ can be parameterized as:

$$\mathbf{A}_k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A pupil detector is used in each time step to provide measurements, i.e., $\mathbf{z}_k = [\hat{x}_k \quad \hat{y}_k]^T$ at time step k. Since $\mathbf{z}_k$ refers to position only and for simplicity, matrix $\mathbf{H}_k$ in (32) is chosen as:

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Initialization of the Kalman filter is performed by means of pupil detection. If the latter is successful for two consecutive frames c and $c+1$, the state vector assumes values from the last successful detection, i.e., $\mathbf{x}_0 = [x_{c+1}, y_{c+1}, v_{x_{c+1}}, v_{y_{c+1}}]^T$. The prediction phase of the Kalman filter involves the propagation of the covariance (36), hence an initial covariance matrix must be defined. Since this is updated iteratively when more images are acquired, it can be simply initialized to large values. Assuming that the error of the predicted position and velocity of the pupil's centroid is $\pm d_{max}$ and $\pm v_{max}$ in both directions, a suitable initial covariance matrix could then be:

$$\mathbf{P}_0 = \begin{bmatrix} d_{max}^2 & 0 & 0 & 0 \\ 0 & d_{max}^2 & 0 & 0 \\ 0 & 0 & v_{max}^2 & 0 \\ 0 & 0 & 0 & v_{max}^2 \end{bmatrix}.$$

Additionally, the two covariance matrices associated with the process noise and measurement noise must be defined. These can be constant or changing over time. In [88], they are assumed constant and are empirically determined.

In a number of computer vision problems where Kalman filtering is used (including object tracking), the state and the measurement models might not be linear. In such cases, the standard Kalman filter cannot be used, unless some kind of linearization is performed. Indeed, a Kalman filter that linearizes about the current mean and covariance is known as the EKF [16] and has been used in the context of object tracking. Let $\mathbf{w}_k$ and $\mathbf{v}_k$ denote the process and measurement noise respectively, as described previously. The system model that describes the evolution of the object state over time:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \tag{40}$$

and the measurement model $\mathbf{z} \in \Re^m$:

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k) \tag{41}$$

are nonlinear. In other words, the formulation remains the same with the standard Kalman filter, the only difference being that the functions f and h are considered nonlinear.

Although the values of the process and measurement noise are not known, we can still approximate the state and measurement vector without them [15], for example,

$$\tilde{\mathbf{x}}_k = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0), \tag{42}$$

$$\tilde{\mathbf{z}}_k = h(\tilde{\mathbf{x}}_k, 0), \tag{43}$$

where $\hat{\mathbf{x}}_{k-1}$ denotes the a posteriori estimate of the state from a previous time step. Using the same notation with the standard Kalman filter and following the linearization process and the derivation in [15], the equations of the first step (prediction) are:

$$\tilde{\mathbf{x}}_k = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, 0), \tag{44}$$

$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{W}_{k-1}\mathbf{Q}_{k-1}\mathbf{W}_{k-1}^T, \tag{45}$$

where $\mathbf{A}_k$ is the Jacobian matrix of partial derivatives of $f(\cdot)$ with respect to $\mathbf{x}_k$, $\mathbf{W}_k$ is the Jacobian matrix of partial derivatives of $f(\cdot)$ with respect to $\mathbf{w}_k$ and $\mathbf{Q}_k$ is the process noise covariance matrix.

The second step (update) equations are:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^T)^{-1}, \tag{46}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\tilde{\mathbf{x}}_k, 0)), \tag{47}$$

$$\mathbf{P}_k = (I - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-, \tag{48}$$

where $\mathbf{H}_k$ is the Jacobian matrix of partial derivatives of $h(\cdot)$ with respect to $\mathbf{x}_k$, $\mathbf{V}_k$ is the Jacobian matrix of partial derivatives of $h(\cdot)$ with respect to $\mathbf{v}_k$ and $\mathbf{R}_k$ is the measurement noise covariance matrix.

Other variations of Kalman filters have been devised to improve its performance with respect to its application to computer vision problems. These include the unscented Kalman filter (UKF), [89], which is an improvement over the EKF. While the EKF uses only the first-order terms of the Taylor expansion and, consequently, introduces errors, UKF avoids these errors by using the third- and higher-order terms of the Taylor expansion.

2.1.5.2 Particle Filters

If the posterior pdf is not Gaussian, Kalman filters will not perform adequately. In such a case, particle filters can be used. They are sequential Monte Carlo methods that can be used for object tracking within a Bayesian framework. They come in a variety of names, such as conditional density propagation (or the Condensation algorithm) [17], survival of the fittest [86], interacting particle approximations [87], and so forth, and they have been extensively used in tracking objects. The main concept behind particle filters is to represent the probability distribution of alternative solutions as a set of samples (i.e., particles), each of which carries a weight. Estimates of the posterior distribution are calculated on the basis of these samples and their associated weights. As the number of samples grows, the filter approaches the optimal Bayesian estimate [85]. The ideal scenario would involve sampling directly from the posterior distribution. However, this is hardly the case. Solutions to this would be to use sampling techniques, such as factored sampling or importance sampling. If we cannot sample directly from the posterior pdf, because it is too complex, but we can sample from the prior pdf, a random sampling technique called factored sampling can be used. Each random sample is assigned a weight, and the weighted set can be the approximation to the posterior density. To improve the results of random sampling, alternative sampling techniques can be used. For example, importance sampling does not sample from the prior pdf, but from another density function that can "drive" the selection of samples toward areas of the posterior pdf that contain the most information (i.e., the resulting sample set will describe the posterior pdf more efficiently). Other sampling methods have also been devised [90]. A more thorough description of Kalman filters and particle filters are provided in [14–16] and [85, 91], respectively.

2.1.6 Occlusion Handling

An inevitable problem in object tracking is the occlusion or self-occlusion of the target object. In most video sequences, parts or even the entire tracked object are not visible in all the frames of the sequence, either due to the existence of static objects (e.g., walls, trees, obstacles) that occlude it or due to the existence of more than one moving objects (e.g., two people walking and crossing each other). A large number of algorithms in the tracking literature have ignored occlusions and treated them simply as noise in the matching process. However, a number of approaches try to deal with occlusion in more advanced ways.

The simplest way to handle occlusions is to reinitialize the tracker in a frame where the detected level of occlusion is high enough to lead to tracking drift or failure. For example, in a face tracking algorithm, this would use a face detection scheme similar to the one used to initialize the tracking process for redetecting a face that has been "lost." Obviously, this comes at the expense of computational burden. Based on the observation that the occlusion can be treated as a local effect, another more clever approach could be to segment the object of interest into N parts and perform tracking on the whole set of these parts [92]. This is performed by estimating N possible transformations (e.g., affine ones) and using a voting scheme to select a single transformation representative of the entire object, while the performance of all the transformations is evaluated at regular intervals to enhance the tracking process. This can enable tracking even in the presence of substantial occlusion, since a relatively low number of the object parts needs to be visible to successfully track the object of interest under partial occlusion.

In a similar manner, Fu et al. [48] track the boundary of an object by using an occlusion adaptive motion snake, similar to the ones described in Section 2.1.2. The algorithm provides two predicted locations for each of many nonoverlapping contour segments, one lying in the interior side of the snake and the other in the exterior side. Occlusions are handled by selecting the best of the two predicted locations (interior or exterior), exploiting a combination of intraframe energy functions (based on edges, color, curvature, and area) and interframe energy functions (based on optical flow, motion smoothness, and color consistency). This is performed in a coarse resolution first. Then, a hierarchic search strategy is used to finalize the tracking results.

Finally, prediction schemes, such as Kalman filters can be used to handle occlusions. For example, in [93], a Kalman filter is used to track occluded features of a feature-based 2D object-tracking algorithm. In case of partial occlusion, their coordinates are updated by using the Kalman filter to predict the movement of the upper left and the lower right point of the object bounding rectangle. In case of total occlusion, the Kalman filter predicts the position of the occluded region on the basis of the velocity estimates of the region corners, obtained from the measurements prior to total occlusion. In [77], the algorithm also uses a Kalman filter as an occlusion handling mechanism. It is reported to be robust against short-time partial occlusions. The method cannot handle efficiently severe, complete, and long-time occlusions. A similar approach was used in [94] to track moving people in video

sequences. Additionally, algorithms that can handle multiple hypotheses, such as the particle filters (or the Condensation algorithm) described in Section 2.1.5 can be used for the occlusion period until the image measurements can disambiguate with respect to the actual object position in the frame under examination.

All of the above occlusion handling mechanisms mainly refer to monocular object tracking. In multiple camera systems, self-occlusions and occlusions between moving objects or between moving and static objects can be treated more efficiently, since an object that is occluded in one view might be fully visible in another. Many promising multiple camera methods can cope with occlusion. In [95–97], multiple camera systems were used to track people successfully. These methods combine the information from all cameras to determine the "best" view. When a camera loses the target due to occlusion, information is obtained from other cameras that are also tracking the object. A probabilistic map function aiming at detecting occlusions calculates the probability that a particular location in the image is visible from a specific camera.

A different approach was presented in [98], where a probabilistic weighting scheme for spatial data integration through a simple Bayesian network was employed. The presented tracking scheme first extracts a set of measurements (observations) for the estimation of the state vector. Then, a predictor–corrector filter is exploited to filter the results. Finally, a Bayesian network performs spatial data integration using triangulation, perspective projections, and Bayesian inference. The input to the network is the set of measurements from the previous time step and the states from the views of the other cameras. The output constitutes the input to a Kalman filter whose goal is to maintain a temporal smoothing on the vector of the 3D trajectories.

Handling occlusion of rigid parts of an articulated object is presented in [99, 100]. Occlusion relationships are defined and handled using the relative motion of the articulated object parts. Simple occlusion relationships between two objects A and B are defined using visibility order: A occludes B, B occludes A, A and B are not occluded. Occlusion graphs can describe more complex occlusion relations when more than two objects are involved. Occlusion between two objects is considered unambiguous, if the convex hull of the union of all possible moves of one object can be partitioned from the other object by using a separating plane. Ambiguous occlusion cases are also defined. Matching is performed by registering layered templates which represent different occlusion situations. Since the paper aims at finger tracking, templates depict fingers in a number of occlusion configurations. The minimization of the sum of squared differences is used for template registration.

2.2 Three-Dimensional Rigid Object Tracking

Three-dimentional rigid object tracking can be defined as the estimation of the position and orientation of a rigid object in 3D space from one or more video sequences. The location of a rigid object in the 3D space is determined by the position of its center of mass with respect to a world coordinate system, and the relative orientation of a coordinate system attached to its center of mass with respect to the world coordinate system. Thus, 3D rigid object tracking has to estimate a total of six parameters although in certain applications determining only the position (i.e., considering the object as a point mass) or only the orientation parameters might suffice. One of the most important applications of 3D rigid object tracking is 3D head tracking (often referred to as head pose estimation), which is being used as a preprocessing step or a building block in face recognition and verification, facial expression analysis, avatar animation, human computer interaction, and model-based coding systems. Although the human head is not a rigid object, considering (and tracking) it as such (i.e., considering only the global head motion) is sufficient in a number of cases. Alternatively, head deformations can be taken into account in the tracking algorithm [101]. Other methods focus on tracking the facial features and expressions in two or three dimensions [102–104] but these fall outside the scope of this section. Three-dimensional vehicle tracking is another application of 3D rigid object tracking [105]. Apart from its obvious importance as a stand-alone task, 3D rigid object tracking often constitutes a basic building block of 3D-articulated object tracking methods, where one needs to estimate the position of rigid objects (links) that make up the articulated structure.

Certain methods (e.g., [106]) use 2D tracking techniques to derive the 2D image-plane motion of the object of interest and then a Kalman filter for deriving the 3D motion parameters. Another approach for head pose estimation [61] uses tracking in 2D of salient facial features (eye corners and nose). Projective invariance of the cross-ratios of the eye corners and anthropometric statistics are subsequently used to compute orientation relative to the camera plane. A similar approach in [55] locates robust facial features (eyebrows, eyes, nostrils, mouth, cheeks, and chin) and uses the symmetric properties of certain facial features and rules of projective geometry to determine the direction of gaze.

However most techniques are model-based [i.e., they involve a 3D geometry model (usually enriched with texture information) of the object of interest in order to derive 3D information from the 2D projections of the object on the video sequence]. A typical model-based 3D rigid object tracking algorithm can be roughly outlined as follows: A 3D geometric model of the object is initialized and enriched with texture information. Then, for each video frame the model parameter vector that best describes the object in the new frame is evaluated. The model parameters derived in the previous frame are used to provide a rough estimate of the parameters in the current frame, which is subsequently refined so as to maximize the similarity between the projection of the model on the image plane and the underlying image content.

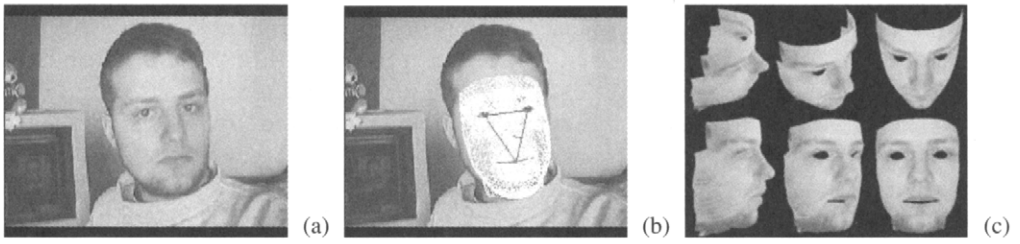

FIGURE 5 (a): A sample frame from a head and shoulders video sequence, (b) a generic face polygon model fitted on the video frame, and (c) the geometric model enriched with texture information. (See color insert.)

The rendering capabilities of current graphics hardware can be utilized for the fast projection (rendering) of the 3D model.

The 3D model representation can be either surface-based (usually a triangular mesh) or volumetric (spheres, cylinders, superquadrics, etc.) [1]. The number of parameters used depends on the accuracy level that is to be obtained and the overall system purpose and architecture. Surface models usually require more parameters than the volume models. A more detailed account of the various types of models that are used to provide a 'flesh' representation of the human body parts (considered as rigid bodies) will be given in Section 3.1. In what concerns the human head, different surface models exist [103, 107–109], some of which were obtained using 3D scanning technologies. The use of the cylindrical representation of the human head is very common because of its simplicity, especially when used in conjunction with texture information [110].

Tracking initialization is not trivial in most cases. During the initialization step, the initial geometric model configuration has to be estimated. Furthermore the texture of the model (reference texture) should be obtained from the video sequence (Fig. 5). Automatic approaches usually involve a step for detecting the object in the first video frame and a subsequent step for registering the 3D model with this projection. In [110] for example, a 2D face detector is used and then the model position is obtained by assuming that the object to be tracked is a head facing upright toward the camera. If the geometric model is a generic one and does not correspond to the actual object that is to be tracked (i.e., the object depicted in the video sequence), an additional step that deforms it to achieve individualization to the tracked object might be applied. The fitting of a generic polygonal face model to a human face is presented in [103]. Although the paper deals with facial feature and expression tracking, the fitting procedure is characteristic of this type of initialization for face tracking techniques. Fitting is performed by localizing important facial features, such as eyes, forehead line, jaw line, mouth, and so forth, on a pair of frontal and profile images. This is achieved through edge detection followed by snake-like techniques. Enhancement of some edges is also performed. In [108], the automatic adaptation of the "CANDIDE" face model [111] to video data is presented. Matching is performed by finding the main facial features (eyes, mouth) using deformable templates. Then, the model is fit using global and local adaptation. The global adaptation step estimates 3D eye and mouth center positions of the face model for scaling, rotation, and translation of the face model in the 3D model world. The system presented in [107] also involves finding the main facial features and performing global and local adaptation. Geometric considerations are used to perform the adaptation. The method involves a geometric model that is more detailed than CANDIDE and does not restrict the initialization images to depict the subject in a certain facial expression (e.g., with mouth closed). The two methods presented above are applied to head and shoulders image sequences since they aim at the initialization of coding algorithms for videophone sequences.

Enriching the geometric model with texture information during initialization involves finding a mapping function between the 2D image plane coordinate system (u, v) and the model surface, represented in parametric form (s, t) in a 2D parametric space (also known as texture map coordinate system). Finding this mapping requires that the 3D location of the model and the projection matrix associated with the camera are known, which is true for the initialization phase.

Since the projection of the 3D model on the image plane (u, v) is a 2D image, its matching with the actual image content can be performed with the techniques and similarity metrics used in 2D rigid object tracking like color similarity, sum of squared differences, joint entropy etc. (see Section 2.1.4). Matching using edge information or other features is also used. Especially in the case of head pose estimation, facial feature (eyes, noise, mouth) localization on the video frames followed by registration of the resulting features with the corresponding features of the projected model can be used.

Estimating the model parameters in a certain frame so that the residual error is minimized can be done using an appropriate search strategy. The trivial solution would be to test all possible combinations of model positions and orientations, but this is computationally prohibitive. Therefore, clever ways of reducing the possible solutions are required. Prior knowledge about the tracking environment or the intended application can provide such constraints. For example, in a videophone application, it would be safe to assume that the user's head would stay relatively close to the camera at all times and that it would not rotate too much

with respect to the camera. Constraints can also be introduced through appropriate motion models. Similar constraints for articulated object tracking are described in more detail in Section 3.1. In some cases (e.g., [101]), and due to the relatively high dimensionality of the matching problem, iterative approaches that aim at minimizing the similarity function over the model parameter space (e.g., standard nonlinear function optimization approaches like the conjugate direction method) are used. Alternatively, a prediction-correction scheme (e.g., the extended Kalman filter) having as error signal the difference between the model projection and the actual image content can be used to update the initial estimate of the model state (i.e., the model position and orientation parameters).

An alternative to projecting the 3D textured model onto the image plane (u, v), and performing matching there, is to perform matching in the texture map coordinate system (s, t) [110]. Indeed, representing the image-derived texture information in the 2D (s, t) coordinate system corresponds to a warped (flattened) image of this texture (Fig. 6). For each model position that has to be tested during the matching procedure, the video frame is warped to the (s, t) coordinate space and compared to the reference texture (i.e., the model texture) represented in the same space. Again, the matching procedure is a 2D one, therefore the similarity metrics outlined above and in Section 2.1.4 are still applicable.

Texture mapping is used in the 3D head tracking system presented in [110]. The head is modelled as a cylinder. Tracking is initialized by using a face detector in the first frame and finding the model position and orientation that fits best with the detected face. The detected face region is mapped as a texture on the cylinder. Tracking is subsequently formulated as an image registration problem, where the previous estimate of the model position and orientation is updated to minimize the sum of squared differences between the reference texture (i.e., the texture of the model) and the warped target texture (texture obtained from the current frame). No iterative optimization is involved in the process making the system capable of performing at 15 frames per second. To improve the performance, illumination regularization is applied using illumination templates. The illumination templates are superimposed on the target region by using a rough registration procedure. The templates do not depend on the specific person being tracked and are obtained by using a training procedure. Geometric head models generated by surface scans have been also used but were found to provide no advantage over the simple cylindric model.

In [64], a generalized 3D model of an average human face is built using a database of range images. An automatic detection step is employed to detect the position of a face, as well as the position of certain facial features (eyes, nose, mouth) in the first frame of a video sequence. The position of the detected facial features is used to properly align the general 3D model to the 2D image data. Subsequently, texture is attached to the 3D mesh using the underlying image intensity values. The 3D coordinates of the feature points are expressed in terms of their obtained 2D coordinates and they are provided as input (along with other parameters describing the camera) to an extended Kalman filter. SSD trackers are used to track the feature points in the frames of the video sequence. The output of these trackers becomes the observation vector (see Section 2.1.5.1) of the Kalman filter. The final output of the filter is the estimate of the 3D position of the head.

FIGURE 6 Head and shoulders image obtained from the Massachusetts Institute of Technology Vision and Modelling Group face database [73], along with the corresponding texture map.

3 Articulated Object Tracking

A number of physical entities in real-world environments can only be represented using articulated structures, [i.e., structures composed of rigid parts (links) connected by joints and typically represented through a tree-like kinematic hierarchy] (Fig. 7). Living beings, such as people or animals, exhibit such attributes. To be able to extract higher level information about the behavior of such entities (e.g., gesture recognition, understanding animal behavior, etc.), precise tracking of the corresponding articulated structures in 3D is necessary. Therefore, most of the articulated tracking algorithms attempt

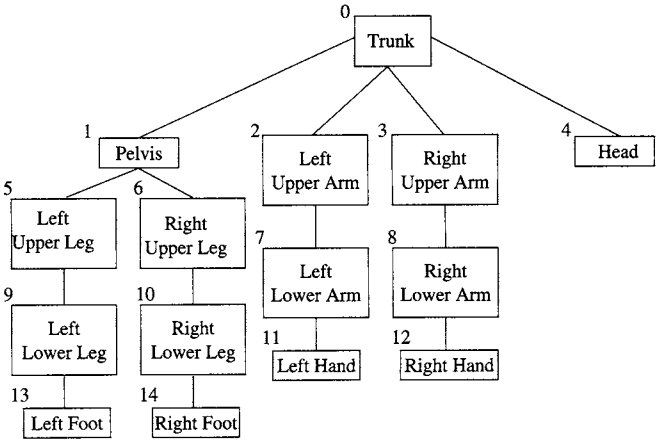

FIGURE 7 A treelike hierarchic representation of the human body.

object tracking in a 3D space. Furthermore, even if the goal is to track an articulated object in 2D (i.e., in the frames of a video sequence), the methodology is similar to that of 3D-articulated tracking algorithms. Moreover, many 2D-articulated object tracking algorithms use a 3D model of the articulated object. For all these reasons, most of this section covers 3D-articulated object tracking, followed by a brief discussion on 2D-articulated object tracking.

3.1 Three-Dimensional Articulated Object Tracking

Three-dimensional articulated object tracking approaches can be model-free or model-based. In the former case, no model of the articulated structure is used. Instead, a bottom-up approach is used to combine image information extracted locally (edges, corners, etc.), to create coherent structures, such as the limbs of the human body. Obviously, this approach requires that the structures reconstructed are constantly visible in the images. In the model-based approaches, which are the most used in 3D-articulated object tracking, a model of the articulated object is used. The complexity of the model depends on the accuracy required in a specific application. The human body, for instance, is represented by rigid parts (resembling limbs) connected to each other at joints. Even such a minimal representation has around 30 *df*.

Therefore, the first step of a model-based 3D-articulated object tracking method is to select an appropriate model. The second step involves finding the model instance that best describes the object under study, when compared with the image data of the video frame in question. One approach uses inverse kinematics [112, 113], as in tracking manipulators in robotics. The motion of the object to be tracked is estimated in the frames of a video sequence and an inverse kinematics framework is used to recover the corresponding motion of the 3D model. In other words, the 2D motion parameters derived for the object parts are used to estimate the pose and joint angles of the 3D model parts. To do this, a nonlinear equation that relates model data to image data is linearized using the Jacobian matrix and is inverted.

A different and widely used approach tries to predict the model configuration that, when projected onto the image, minimizes the error between the model projection and the actual image data. The 3D model is initialized either manually or automatically, using information extracted from the first frame of a video sequence. The ideal case would then be to perform an exhaustive search of all possible model configurations, attempting to match their projections against the information extracted from the next frame in the sequence. Such an exhaustive search, however, would be possible only for a model of a very low dimensionality. For models having an order of 20 to 30 *df*, "clever" ways of reducing the high-dimensional search space are required. This can be performed by "pruning" the search space using kinematic and other constraints. Even then, it remains prohibitively large for an exhaustive match. Further reduction can be achieved if additional constraints in the model configurations that need to be searched are introduced. The selection of a motion model is a way of introducing such constraints. For instance, if the motion of the object of interest is known to be periodic or if we wish to track the motion of people performing specific actions such as running or walking [13], appropriate motion models can be used to significantly reduce the complexity of the tracking process and, hence, the computational power required.

If one model configuration is to be estimated, single hypothesis methods, such as Kalman filters [41, 114] and least squares [112–116] can be used. It is often the case that, due to the complex nature of 3D-articulated object modelling and tracking, a single hypothesis method will result in loss of tracking or severe tracking drift. Methods capable of tracking multiple hypotheses, such as particle filtering or the Condensation algorithm [13, 17, 117], described in Section 2.1 or others [118] can be used instead. Multiple hypotheses are maintained until the extracted image data can help pinpoint a single model configuration. Regardless of the single or multiple hypotheses maintained in each time step, the choice of the type of image data used to match against the projected model configurations and the method of their extraction plays a crucial role in accurately tracking the articulated object in 3D.

A number of decisions have to be made when devising an algorithm for tracking 3D-articulated objects. One such important decision is the number of cameras used to obtain the video sequence(s) in which objects will be tracked. More than one camera can be used to deal with the inherent depth ambiguities of monocular tracking. Whether one or more cameras are used, camera calibration might need to be performed prior to obtaining the sequences, as described in Section 1.

Although multiple cameras can help disambiguate depth problems, monocular tracking is quite important, obviously due to the technical simplicity of single camera systems. However, the number of systems that use 3D monocular articulated object tracking is still small, because of the difficulties that arise from the use of a single camera. First, depth information is lost when projecting the world into 2D images, classifying 3D articulated object tracking as an ill-posed problem that might produce more than one solutions unless additional constraints or prior knowledge are used. This problem is, for example, obvious in the case of symmetric positions of the arm in 3D, close to its neutral position, which result in the same projection. Additionally, self-occlusion between different parts of the articulated object (e.g., the human body), which is considerably more difficult if a single camera is available, often occurs and, hence, has to be tackled with.

Another difficulty in 3D-articulated tracking algorithms is associated with the deformable clothing of humans, where most of the attempts on articulated tracking are focused on. Attempts to provide a solution by cloth simulation and reconstruction techniques require a carefully controlled multi-camera and lighting environment [119], while resulting in severe computational overheads. Attention should also be paid to the fact that the object model is matched against noisy image measurements. Images often contain many distracting features that can be potentially mistaken for the object parts. For example, illumination changes create problems if image edges or intensities are the features used for tracking. Shadows can produce false edges and varying illumination conditions can cause texture changes that can not be accounted for by the model used. Also, during rapid motion, blurring occurs at the motion boundaries, which can severely affect methods that rely on static feature extraction, such as the ones that compute the optical flow. Solutions to this problem involve the fusion of image cues that provide complementary information.

In the rest of this section the different stages of devising a 3D-articulated object tracking algorithm will be presented, along with example techniques.

3.1.1 3D Modeling

Since various applications such as human computer interaction, surveillance, etc. focus on people, many efforts have been targeted on representing highly articulated structures, such as the human body, using 3D models that vary in complexity, based on the accuracy needed in each specific application. 3D articulated models are also used in motion capture applications, where the goal of the tracking process is to recover the full-body pose. Due to the complexity of these models, the computational overheads are increased.

Human body models generally consist of two components: the skeletal structure and a representation for the flesh surrounding it. The skeletal structure is a collection of segments and joints with various degrees of freedom at the joints, as illustrated in Fig. 8. In general, each joint is associated with 3 *df*, while additional degrees of freedom are required for the global position and orientation of the model. In many cases, joints are allocated lesser degrees of freedom than the actual ones in order to reduce model complexity to a tractable level. The parameters of the skeletal structure are the joint angles. If the individual segments comprising the articulated structure have been allowed to deform, the shape parameters of the segments are also included in the set of parameters that parameterize the model. Additionally, the articulated object model should be initialized in a consistent manner, e.g., for humans, by taking into account the standard humanoid dimensions, as specified in [120, 121].

The flesh can be represented using polygons (Fig. 9) or other surface primitives [103, 107]. Alternatively, volumetric primitives such as spheres, cylinders (Fig. 10) [13, 110, 112], cones [114, 117, 122], superquadrics (generalized ellipsoids with additional parameters along each axis, which encode the "squareness") [116, 118, 123] can be used.

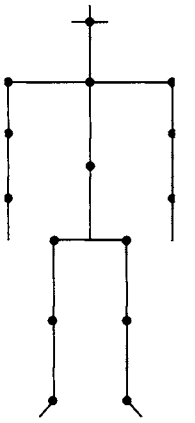

FIGURE 8 A example of a stick human body model. Each rigid part can be associated with 1 to 3 df.

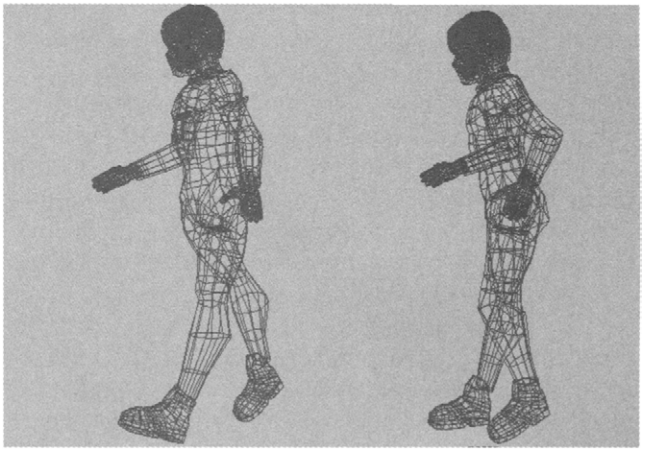

FIGURE 9 Polygonal representation of the human body, adapted from [124]. (See color insert.)

The accuracy of the representation achieved is proportional to the number of parameters used in the model and depends on the intended application, as mentioned in Section 1. In computer graphics, extremely accurate surface models (often obtained through body scans of actual people), that consist of thousands of polygons, are employed. In computer vision, however, the inverse problem we deal with, i.e., recovering a 3D model from images is much harder and less accurate. Consequently, coarse volumetric primitives have been preferred, because of the lower number of model parameters involved. These models are subsequently used for tracking human body parts in 3D space.

3.1.2 Kinematic and Motion Constraints

Kinematic constraints are distinguished in "hard" and "soft" constraints. The former are usually in the form of angular

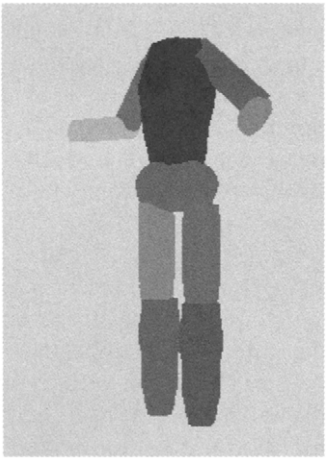

FIGURE 10 Cylinder-based volumetric representation of the human body. (See color insert.)

displacements, velocity and acceleration limits of the joints of an articulated structure, whereas the latter are probabilistic and associated to previous instances of the object (e.g., human) motion. For example, the range of angles where the human elbow can move is limited and these limits should be taken into account during the tracking process. Enforcing anatomic joint angle limits is relatively straightforward. They can be applied by specifying minimum and maximum allowed values for the joint degrees of freedom [114, 118]. Moreover, the speed attained by different body parts connected to the same joint is different. An example showing the need of imposing constraints on the motion of certain body parts is the human finger, which consists of three parts. The middle part moves to greater extent and more rapidly than the other two. Moreover, the human finger is not allowed to move to all directions. Finally, any physical noninterpenetration constraints between the different parts of the articulated structure have to be incorporated [118]. These constraints stem from the fact that two parts cannot occupy the same space simultaneously.

Prior information about the motion to be tracked can also lead to further constraints that can be used to improve the tracking process. For instance, several human activities are repetitive in nature (e.g., walking or running) and the corresponding motion can, therefore, be modelled using specific motion models that can be learned from training data [13]. Alternative methods include example-based motion models [125], where given a database of example human motions, the authors construct a low-dimensional model of the motion and project subsequences in the database onto this low-dimensional representation, obtaining a coefficient vector at each time step. Coefficients are then used to index the database. The same projection is applied to any new motion that is observed (and should be tracked). The problem of tracking then becomes a problem of searching the database efficiently and matching the observed motion with examples from the database. The examples matched are used as samples in a stochastic sampling (i.e., Condensation algorithm [17]) framework. Finally, motion can be modelled as a higher-order autoregressive process, as in [117]. Attempts have also been made to perform tracking by switching between multiple motion models [126]. Such an approach, however, severely increases the computational burden. One of the major problems is that the motion models used so far are not generic enough to be applied in various real-world conditions.

3.1.3 Image Cues

As soon as the model of the object has been properly defined, a method for matching the projected model configurations to the image data is required. Many different image features can be utilized for this purpose. They range from low-level (e.g., edges) to high-level cues (e.g., the locations of the joints). Joint locations are difficult to recover directly, because there is no characteristic intensity distribution around them. They are consequently inferred using the adjoining rigid parts of the articulated object, thus making the method very sensitive to the segmentation of the articulated object. These difficulties lead to the use of low-level features for the matching process. Edge information can be used, because edges are partially invariant to changes in viewpoint and lighting conditions [127], while being easy to detect, especially in humans. There are 3D motion patterns, however, that cannot be detected using edges. For instance, when the human limbs rotate around their 3D symmetry axes, the edge changes in the image are insufficient for the detection of such a motion [118].

To alleviate such problems, intensity can be used as a cue for image/model matching. This involves a step of acquiring a reference texture, either as part of an initialization stage [99] or in the previous time frame [13, 112–114, 118]. The reference texture is mapped onto the model surface. The textured model is projected onto the image and matched against the underlying image texture. This, however, requires that image texture can be reliably extracted at all times, otherwise tracking may deteriorate. Complex real-world environments, where intensity may vary significantly, the subjects may wear loose and deformable clothing, and texture information may not be easy to extract, can severely affect the performance of such algorithms. Silhouettes can additionally be employed to track objects (especially people). They can be acquired by motion segmentation, contour tracking or background segmentation [11, 122], as described earlier.

To increase the robustness of the incorporated algorithms, an object tracking system may use more than one image cues, in an effort to extract as much information as possible from the image. Several variations are possible and include a combination of the above, such as edges and intensity, silhouettes and edges [114, 117], and so forth.

3.1.4 Example Techniques

In this section, a number of works on 3D-articulated object tracking are briefly presented. In [128], a 3D tracking system for high degree of freedom articulated structures is presented. The human hand is modelled using an articulated structure consisting of 16 rigid bodies (15 rigid bodies for the fingers and one for the palm). The total hand pose is represented by a 28-dimensional vector. Each finger part is characterized by its central axis line. The finger tip coordinates are also used. A priori knowledge of the hand kinematics and geometry and exact hand localization is required. Matching is performed by using a prediction mechanism. A correction scheme, formulated as a linear least squares problem, is subsequently applied.

In [123], the authors perform tracking in a hierarchic manner. They use an articulated 3D model. The human body skeletal structure is modelled by a 22-*df* "stick figure" model, while for the flesh-shape representation, the class of tapered superquadrics that includes shapes such as cylinders, spheres, ellipsoids, and hyperrectangles is utilized. They use edges as image cues and they use chamfer matching as a similarity measure between projected model configurations and actual image data. To deal with depth ambiguities, they make use of a ring of four inward-looking calibrated cameras, whereas to increase the accuracy of the method, they use subjects that wear tight clothing with sleeves of contrasting colors. Their hierarchic approach produces solutions for the different parts of the human body in a top-down manner using a kinematic chain, i.e., the position of the torso is found first and then the other body parts lower down in the hierarchy, such as the individual limbs, are recursively estimated. The proposed technique does not handle occlusion well.

In [112], twists are used to model the kinematic chain, while the individual body parts are modelled using cylinders. The authors perform monocular tracking of subjects walking either parallel to the image plane or diagonally towards the camera. Tracking is also performed on a number of the classic Muybridge images recorded in 1884, where images from three different viewpoints are available at each time instance. To reduce the complexity, only half the human body skeleton (19 *df*), corresponding to the visible side of the body, is used.

In [122], a volumetric model comprising of parallelepipeds, spheres, and truncated cones is used. The authors employ three calibrated cameras and derive 2D springlike forces between the predicted model and the extracted silhouette. The derived 2D forces from each camera are combined to obtain 3D forces to be applied to the 3D model and align it with the data. The dynamic simulation is embedded in a Kalman filtering framework. Results are reported on subjects running in an indoor environment, wearing unconstrained clothing.

In [114], a shape model built from truncated cones is used to estimate motion in a monocular sequence using articulated kinematics. Edges and intensity are used as image cues in an EKF. In the Kalman filter prediction step, anatomic joint parameter limits are enforced. The results reported involve tracking in an unconstrained environment of a subject moving parallel to the image plane.

Sidenbladh et al. [13] use an articulated 3D model based on cylinders. They perform tracking in monocular cluttered sequences, using intensity as the image cue. They use the Condensation algorithm [17] for tracking. To constrain the tracking process, they use prior cyclic motion models for tracking humans walking. In a more recent work of the same authors, instead of using prior motion models, they use models learned from training data [125]. Results are reported for tracking the planar motion of an arm and the motion of a walking person.

3.2 Two-Dimensional Articulated Object Tracking

As already mentioned in Section 1, articulated objects can be defined as objects composed of more than one rigid parts (links) connected by joints allowing rotational or translational motion in 1, 2, or 3 *df*. Two-dimensional articulated object tracking refers to recovering the position and size in 2D, i.e., on the image plane, of the rigid parts comprising the articulated structure. It follows that 2D-articulated object tracking shares characteristics and principles with 2D rigid object tracking, described in Section 2.1, as well as 3D-articulated object tracking, described in Section 3.1.

Similarly to 3D algorithms, 2D-articulated object tracking can be model-free or model-based. Model-free methods proceed by exploiting image information (edges, intensity, etc.) in order to create coherent structures that correspond to the rigid parts of the articulated structure (e.g., the upper and lower segments of the human arm). Occlusion can create problems in such methods, due to the lack of visibility or partial visibility of one or more of the rigid parts comprising the articulated object. Alternatively, model-based approaches use a 2D or 3D model of the articulated object, depending on the application and the precision required. The rigid parts of a 2D model can be represented using geometric primitives, such as sticks (i.e., lines), circles, rectangles, and ellipses or by curves and snakes if the object parts are allowed to deform. These can be used to represent the shape of the object. Additionally, the appearance of the object can be modelled by including texture on the above mentioned geometric primitives. If a 3D model is used, this can be similar to the ones described earlier in Section 3.1.

In an early work, Lowe [129] uses a 3D model to track an object in video sequences. The Marr-Hildreth edge detector [130] is used to identify edges in the frames of the video sequence. The edges are grouped based on local connectivity.

The matching process finds the best match between the extracted image edges and the projected contours of the 3D model. The matches are ranked and the best ones are chosen for minimization. The model is then rotated and translated and the above procedure is iteratively repeated until the best view is found. The computational time depends on the complexity of the model used. The algorithm is robust but rather slow.

A model-based 2D-articulated object tracking algorithm is presented in [131]. The proposed system uses three different models: a 2D-articulated model of the object being tracked, a dynamic model, and an appearance model. The 2D-articulated model approximates the shape of the object in the image (in this case the human body) and consists of rigid parts that are connected with each other. The displacement vectors and the joint angles are the parameters that describe the motion of the articulated object. The dynamic model is a stochastic linear equation that rules these parameters and is used to predict the model configuration. It consists of a trained linear stochastic equation and a look-up table that contains a list of exceptions (nonlinear configurations). The appearance model is a set of profiles of the object centered at specific feature points on the object. These feature points are detected during the tracking process using template matching. The tracking of the object is achieved through the following steps: first, the dynamic model predicts potential configurations of the model and consequently a template matching technique is used to detect the specific feature points for each configuration. After filtering the results by EKFs and evaluating them by measuring the color matchings between the results and the appearance model, the configuration with the largest matching score is selected as the final result for this frame. Other model-based articulated object tracking methods are presented in [132–135].

A model-free articulated object tracking system is presented in [136], avoiding the use of prior models by exploiting the pyramid-based Kanade-Lucas-Tomasi feature tracking algorithm [137] and a foreground color distribution obtained through training. Feature points are selected on the object to be tracked by using the algorithm presented in [57], which is described in Section 2.1.3. The feature points are considered to be a flock of features, i.e., a global constraint is enforced on the feature locations that keeps them spatially combined. During the tracking process, the feature points must be close to each other and not far from the median feature point. Two thresholds define the allowable distances in each case. The algorithm is tested on human hands. The color of the human hand is learned by a hand detection scheme which provides a normalized-RGB histogram. The CamShift algorithm [36] operates on this modality. Therefore, the color distribution is used as a probability map during the selection and tracking of the feature points. In each case, high skin color probability locations are preferable. Another model-free articulated object tracking technique is introduced in [94].

Acknowledgment

The authoring of this chapter has been supported by VISNET, a European Network of Excellence (visnet-noe.org) funded under the European commission IST FP6 program.

References

[1] D. M. Gavrila, "The visual analysis of human movement: A survey," *Computer Vision and Image Understanding* **73**, 82–98 (1999).

[2] Y. Luo, T.D. Wu, and J. N. Hwang, "Object-based analysis and interpretation of human motion in sports video sequences by dynamic bayesian networks", *Computer Vision and Image Understanding* **92**, 196–216 (2003).

[3] P. Figueroa, N. Leite, R. M. L. Barros, I. Cohen, and G. Medioni, "Tracking soccer players using the graph representation," in *International Conference on Pattern Recognition (ICPR2004)*, **IV** (Cambridge, England, August 2004), 787–790.

[4] M. Köhle, D. Merkl, and J. Kastner, "Clinical gait analysis by neural networks: issues and experiences," in *10th IEEE Symposium on Computer-Based Medical Systems (CBMS97)* (Maribor, Slovenia, March 1997), 138–143.

[5] D. Meyer, J. Denzler, and H. Niemann, "Model based extraction of articulated objects in image sequences," in *IEEE International Conference on Image Processing (ICIP97)*, **III** (Washington, DC, October 1997), 78–81.

[6] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Comput. Visi. Image Understanding* **81**, 231–268 (2001).

[7] T. B. Moeslund, *Interacting with a virtual world through motion capture, in Interaction in Virtual Inhabited 3D Worlds*, L. Qvortrup, ed. (Springer-Verlag, Berlin/New York, 2000), Chap. 11.

[8] G. H. Burdea and P. Coiffet, *Virtual Reality Technology, 2nd ed.* (Wiley-Interscience, New York, 2003).

[9] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Comput. Graph. Appl.* **22**, 24–38 (2002).

[10] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata, "Nonrigid motion analysis: Articulated and elastic motion," *Comput. Vis. Image Understanding* **70**, 142–156 (1998).

[11] I. Haritaoglu, R. Cutler, D. Harwoodc, and L. S. Davis, "Backpack: Detection of people carrying objects using silhouettes," *Comput. Visi. Image Understanding* **81**, 385–397 (2001).

[12] M. Isard and J. MacCormick, "BraMBLe: A bayesian multiple-blob tracker," in *IEEE International Conference on Computer Vision (ICCV2001)* (Vancouver, BC, Canada, July 2001), 34–41.

[13] H. Sidenbladh, M. Black, and D. Fleet, "Stochastic tracking of 3D human figures using 2D image motion," in *European Conference on Computer Vision, (ECCV2000)* (Dublin, Ireland, June 2000), 718–720.

[14] P. S. Maybeck, *Stochastic Models, Estimation and Control, Volume 1* (Academic, New York, 1979).

[15] G. Welch and G. Bishop, An introduction to the Kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, Department of Computer Science, 1995.

[16] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach* (American Institute of Aeronautics, 2001).

[17] M. Isard and A. Blake, "Condensation-conditional density propagation for visual tracking," *Int. J. Comput. Visi.* **29**, 5–28 (1998).

[18] O. Faugeras, L. Quan, and P. Strum, "Self-calibration of a 1D projective camera and its application to the self-calibration of a 2D projective camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, 1179–1185 (2000).

[19] E. Malis and R. Cipolla, "Camera self-calibration from unknown planar structures enforcing the multiview constraints between collineations," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**, 1268–1272 (2002).

[20] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition* **36**, 585–601 (2003).

[21] J. K. Aggarwal and Q. Cai, "Human motion analysis: A review," *Comput. Visi. Image Understanding* **73**, 428–440 (1999).

[22] J. J. Wang and S. Singh, "Video analysis of human dynamics-a survey," *Real Time Imaging J.* **9**, 320–345 (2003).

[23] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics, Part C* **34**, 334–352 (2004).

[24] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision.* (Prentice Hall, Upper Saddle River, NJ, 1998).

[25] T. Gevers and A. W. M. Smeulders, "Pictoseek: Combining color and shape invariant features for image retrieval," *IEEE Trans. Image Process.* **9**, 102–119 (2000).

[26] G. Buchsbaum, "A spatial processor model for object color perception," *J. Franklin Instit.* **310**, 1–26 (1980).

[27] G. Finlayson, S. Hordley, and P. Hubel, "Color by correlation: A simple, unifying framework for colour constancy," *IEEE Trans. Pattern Anal. Mach. Intell.* **23**, 1209–1221 (2001).

[28] B. D. Zarit, B. J. Super, and F. K. H. Quek, "Comparison of five color models in skin pixel classification," in *ICCV99 International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS99)* (Corfu, Greece, September 1999), 58–63.

[29] C. Terrillon, M. David, and S. Akamatsu, "Automatic detection of human faces in natural scene images by use of a skin color model and invariant moments," in *Third IEEE International Conference on Automatic Face and Gesture Recognition (AFGR98)* (Nara, Japan, April 1998), 112–117.

[30] A. Albiol, L. Torres, and E. Delp, "Optimum color spaces for skin detection," in *IEEE International Conference on Image Processing (ICIP2001)*, Vol. 1 (Thessaloniki, Greece, October 2001), 122–124.

[31] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. of Comput. Vis.* **7**, 11–32 (1991).

[32] M. Vezhnevets, "Face and facial feature tracking for natural Human-Computer Interface," in *International Conference on Computer Graphics between Europe and Asia (GraphiCon-2002)* (Nizhny Novgorod, Russia, September 2002), 86–90.

[33] K. Schwerdt and J. L. Crowley, "Robust face tracking using color," in *International Conference on Automatic Face and Gesture Recognition (AFGR2000)* (Grenoble, France, March 2000), 90–95.

[34] W. Lu, J. Yang, and A. Waibel, "Skin-color modeling and adaptation," in *Third Asian Conference on Computer Vision (ACCV98)* Vol. 2 (Hong Kong, China, January 1998), 687–694.

[35] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, Color-based probabilistic tracking," in *European Conference on Computer Vision (ECCV2002)*, Vol. 1 (Copenhagen, Denmark, May–June 2002), 661–675.

[36] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Workshop on Applications of Computer Vision (WACV98)* (Princeton, NJ, October 1998), 214–219.

[37] Y. Wu and T. S. Huang, "Color tracking by transductive learning," in *International Conference on Computer Vision and Pattern Recognition (CVPR2000)*, Vol. 1 (Hilton Head, SC, June 2000), 133–138.

[38] H. Sidenbladh and M. Black, "Learning the statistics of people in images and video," *Int. J. Comput. Vis.* **54**, 181–207 (2003).

[39] S. J. McKenna, Y. Raja, and S. Gong, "Tracking and segmenting people in varying lighting conditions using colour," in *International Conference on Automatic Face and Gesture Recognition (AFGR98)* (Nara, Japan, April 1998), 228–233.

[40] P. Fieguth and D. Terzopoulos, "Color-based tracking of heads and other mobile objects at video frame rates," in *International Conference on Computer Vision and Pattern Recognition (CVPR97)* (San Juan, PR, June 1997), 21–27.

[41] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "PFinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. and Mach. Intell.* **19**, 780–785 (1997).

[42] C. Smith, C. Richards, S. A. Brandt, and N. P. Papanikolopoulos, "Visual tracking for intelligent vehicle-highway systems," *IEEE Trans. Vehic. Tech.* **45**, 744–759 (1996).

[43] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 747–757 (2000).

[44] A. Baumberg and D. Hogg, "Learning flexible models from image sequences," in *European Conference on Computer Vision (ECCV94)*, Vol. 1 (Stockholm, Sweden, May 1994), 299–308.

[45] Q. Cai and J. Aggarwal, "Tracking human motion using multiple cameras," in *International Conference on Pattern Recognition (ICPR96)*, Vol. 3 (Vienna, Austria, August 1996), 68–72.

[46] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 809–830 (2000).

[47] M. Kass, M. Witkin, and A. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.* **1**, 321–331 (1988).

[48] Y. Fu, A. T. Erdem, and A. M. Tekalp, "Tracking visible boundary of objects using occlusion adaptive motion snake," *IEEE Trans. Image Process.* **9**, 2051–2060 (2000).

[49] H. Wang, J. Leng, and Z. M. Guo, "Adaptive dynamic contour for real-time object tracking," in *Image and Vision Computing New Zealand (IVCNZ2002)* (Auckland, New Zealand, December 2002).

[50] B. Bascle, P. Bouthemy, R. Deriche, and F. Meyer, "Tracking complex primitives in an image sequence," in *International Conference on Pattern Recognition (ICPR94)*, Vol. 1 (Jerusalem, Israel, October 1994), 426–431.

[51] N. Xu and N. Ahuja, "Object contour tracking using graph cuts based active contours," in *IEEE International Conference on Image Processing (ICIP2002)*, Vol. 3 (Rochester, NY, September 2002), 277–280.

[52] N. Xu, R. Bansal, and N. Ahuja, "Object segmentation using graph cuts based active contours," in *International Conference on Computer Vision and Pattern Recognition (CVPR2003)*, Vol. 2 (Madison, WI, June 2003), 46–53.

[53] K. Toyama, "Prolegomena for robust face tracking," Technical Report MSR-TR-98-65, Microsoft Research, 1998.

[54] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Trans. Pattern Anal. Mach. Intell.* **8**, 2–14 (1986).

[55] A. Nikolaidis and I. Pitas, "Probabilistic multiple face detection and tracking using entropy measures," *Pattern Recognition* **33**, 1783–1791 (2000).

[56] C. Tomasi and T. Kanade, "Detection and tracking of point features," Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991.

[57] J. Shi and C. Tomasi, "Good features to track," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR94)* (Seattle, WA, June 1994), 593–600.

[58] J. Crowley and F. Berard, "Multi-modal tracking of faces for video communication," in *International Conference on Computer Vision and Pattern Recognition (CVPR97)* (San Juan, June 1997), 640–645.

[59] E. Elagin, J. Steffens, and H. Neven, "Automatic pose estimation system for human faces based on bunch graph matching technology," in *International Conference on Automatic Face and Gesture Recognition (AFGR98)* (Nara, Japan, April 1998), 136–141.

[60] A. Gee and R. Cipolla, "Fast visual tracking by temporal consensus," *Image Vis. Comput.* **14** 105–114 (1996).

[61] T. Horprasert, Y. Yacoob, and L. S. Davis, "Computing 3-D head orientation from a monocular image sequence," in *International Conference on Automatic Face and Gesture Recognition (AFGR96)* (Killington, VT, October 1996), 242–247.

[62] T. Maurer and C. Von Der Malsburg, "Tracking and learning graphs and pose on image sequences of faces," in *International Conference on Automatic Face and Gesture Recognition (AFGR96)* (Killington, VT, October 1996), 176–181.

[63] N. Oliver, A. Pentland, and F. Berard, "LAFTER: Lips and face real time tracker," in *International Conference on Computer Vision and Pattern Recognition (CVPR97)* (San Juan, PR, June 1997), 123–129.

[64] T. S. Jebara and A. P. Pentland, "Parameterized structure from motion for 3D adaptive feedback tracking of faces," in *International Conference on Computer Vision and Pattern Recognition (CVPR97)* (San Juan, PR, June 1997), 144–150.

[65] P. Yao, G. Evans, and A. Calway, "Face tracking and pose estimation using affine motion parameters," in *12th Scandinavian Conference on Image Analysis* (Norwegian Society for Image Processing and Pattern Recognition, June 2001), 531–536.

[66] C. Luo, T. S. Chua, and T. K. Ng, "Face tracking in video with hybrid of Lucas-Kanade and Condensation algorithm," in *IEEE International Conference on Multimedia and Expo (ICME2003)* (Baltimore, MD, July 2003), 531–536.

[67] H. Chao, Y. F. Zheng, and S. C. Ahalt, "Object tracking using the Gabor wavelet transform and the golden section algorithm," *IEEE Transactions on Multimedia* **4**, 528–538 (2002).

[68] A. Shokurov, A. Khropov, and D. Ivanov, "Feature tracking in images and video," in *International Conference on Computer Graphics between Europe and Asia (GraphiCon-2003)* (Moscow, Russia, September 2003), 177–179.

[69] E. Loutas, I. Pitas, and C. Nikou, "Probabilistic multiple face detection and tracking using entropy measures," *IEEE Transactions on Circuits and Systems for Video Technology* **14**, 128–135 (2004).

[70] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence* (Vancouver, BC, Canada, August 1981), 674–679.

[71] P. Beardsley, P. H. S. Torr, and A. Zisserman, "3D model aquisition from extended image sequences," in *European Conference on Computer Vision (ECCV96)*, Vol. 2 (Cambridge, England, April 1996), 683–695.

[72] S. Marcelja, "Mathematical description of the responses of simple cortical cells," *J. Opt. Soc. Amer.* **70**, 1297–1300 (1980).

[73] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cog. Neurosci.* **3**, 71–96 (1991).

[74] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vis.* **26**, 63–84 (1998).

[75] S. Pigeon and L. Vandendorpe, "The M2VTS multimodal face database," in *International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA97)* (Crans Montana, Switzerland, March 1997), 403–409.

[76] H. T. Nguyen and A. W. M. Smeulders, "Template tracking using color invariant pixel features," in *IEEE International Conference on Image Processing (ICIP2000)* Vol. 1 (Rochester, NY, September 2000), 569–572.

[77] H. T. Nguyen and A. W. M. Smeulders, "Fast occluded object tracking by a robust appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.* **26**, 1099–1104 (2004).

[78] C. F. Olson, "Maximum-likelihood image matching," *IEEE Trans. Pattern Anal. Mach. Intell.* **24**, 853–857 (2002).

[79] L. V. Tsap, D. B. Goldgof, and S. Sarkar, "Fusion of physically-based registration and deformation modeling for nonrigid motion analysis," *IEEE Trans. Image Process.* **10**, 1659–1669 (2001).

[80] Y. Wang and S. Zhu, "Analysis and synthesis of textured motion, Particles and waves," *IEEE Trans. Pat. Anal. Mach. Intell.* **26**, 1348–1363 (2004).

[81] T. Schoepflin, V. Chalana, D. R. Haynor, and Y. Kim, "Video object tracking with a sequential hierarchy of template deformations," *IEEE Trans. Circuits Syst. Video Tech.*, **11**, 1171–1182 (2001).

[82] R. Kjeldsen and A. Aner, "Improving face tracking with 2D template warping," in *International Conference on Face and Gesture Recognition (AFGR2000)* (Grenoble, France, March 2000), 129–135.

[83] J. Heinzmann and A. Zelinsky, "Robust real-time face tracking and gesture recognition," in *International Joint Conference on Artificial Intelligence (IJCAI97)* (Nagoya, Aichi, Japan, August 1997), 1525–1530.

[84] Y. Zhong, A. K. Jain, and M. P. Dubuisson-Jolly, "Object tracking using deformable templates," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 544–549 (2000).

[85] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.* **50**, 174–188 (2002).

[86] K. Kanazawa, D. Koller, and S. Russell, "Stochastic simulation algorithms for dynamic probabilistic networks," in *Eleventh Annual Conference on Uncertainty in AI (UAI95)* (Montreal, Canada, August 1995), 346–351.

[87] P. Moral, "Non-linear filtering: Interacting particle solution," *Markov Processes Related Fields* **2**, 555–580 (1996).

[88] Ji, Q. Qiang and Yang, X. Xiaojie, "Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance," *Real Time Imaging* **8**, no. 5, Oct. 2002, 357–377.

[89] S. J. Julier, J. K. Ulmann, and H. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference* (Orlando, April 1995), 1628–1632.

[90] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian non-linear state space models," *J. Comput. Graph. Statist.* **5**, 1–25 (1996).

[91] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice* (Springer-Verlag, New York, 2001).

[92] C. Gentile, O. Camps, and M. Sznaier, "Segmentation for robust tracking in the presence of severe occlusion," *IEEE Trans. Image Process.* **13**, 166–178 (2004).

[93] E. Loutas, K. I. Diamantaras, and I. Pitas, "Occlusion resistant object tracking," in *IEEE International Conference on Image Processing (ICIP2001)*, Vol. II (Thessaloniki, Greece, October 2001), 65–68.

[94] Y. Ricquebourg and P. Bouthemy, "Real-time tracking of moving persons by exploiting spatio-temporal image slices," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 797–808 (2000).

[95] H. Tsutsui, J. Miura, and Y. Shirai, "Optical flow-based person tracking by multiple cameras," in *IAPR Workshop on Machine Vision Applications (MVA98)* (Chiba, Japan, November 1998), 418–421.

[96] A. Mittal and L. S. Davis, "M2Tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using regionbased stereo," *Int. J. Comput. Vis.* **51**, 189–203 (2003).

[97] A. Utsumi, H. Mori, J. Ohya, and M. Yachida, "Multiple-human tracking using multiple cameras," in *International Conference on Automatic Face and Gesture Recognition (AFGR98)* (Nara, Japan, April 1998), 498–503.

[98] S. L. Dockstader and A. M. Tekalp, "Multiple camera tracking of interacting and occluded human motion," in *Proceedings of the IEEE*, Vol. 89 (October 2001), 1441–1455.

[99] J. Rehg and T. Kanade, "Model-based tracking of self occluding articulated objects," in *IEEE International Conference on Computer Vision, (ICCV95)* (Cambridge, MA, June 1995), 612–617.

[100] D. D. Morris and J. Rehg, "Singularity analysis for articulated object tracking," in *International Conference on Computer Vision and Pattern Recognition (CVPR98)* (Santa Barbara, CA, June 1998), 289–296.

[101] J. Paterson and A. Fitzgibbon, "3D head tracking using non-linear optimization," in *British Machine Vision Conference*, Vol. 2 (Norwich, UK, September 2003), 609–618.

[102] J. Ahlberg and R. Forchheimer, "Face tracking for model-based coding and face animation," *Int. J. Imaging Syst. Technol.* **13**, 8–22 (2003).

[103] T. Goto, S. Kshirsagar, and N. Magnenat-Thalmann, "Automatic face cloning and animation," *IEEE Signal Process.* **18**, 17–25 (2001).

[104] H. Li, P. Roivainen, and R. Forcheimer, "3D motion estimation in model-based facial image coding," *Trans. Pattern Anal. Mach. Intell.* **15**, 545–555 (1993).

[105] W. Hu, X. Xiao, D. Xie, T. Tan, and S. Maybank, "Traffic accident prediction using 3D model-based vehicle tracking," *IEEE Trans. Vehic. Technol.* **53**, 677–694 (2004).

[106] A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland, "Visually controlled graphics," *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 602–605 (1993).

[107] M. Kampmann. "Automatic 3D face model adaptation for model-based coding of videophone sequences," *IEEE Trans. Circ. Syst. Video Tech.* **12**, 172–182 (2002).

[108] L. Zhang, "Automatic adaptation of a face model using action units for semantic coding of videophone sequences," *IEEE Trans. on Circ. Syst. Video Tech.* **8**, 781–795 (1998).

[109] S.-C. Pei, C.-W. Ko, and M.-S. Su, "Global motion estimation in model-based image coding by tracking three-dimensional contour feature points," *IEEE Trans. Circ. Syst. Video Tech.* **8**, 181–190 (1998).

[110] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3D models," *IEEE Trans. Med. Imag.* **22**, 322–336 (2000).

[111] R. Rydfalk, "CANDIDE, a parameterised face," Technical Report Lith-ISY-I-0866, Univesity of Linkping, Sweden, 1987.

[112] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR98)* (Santa Barbara, CA, June 1998), 8–15.

[113] S. X. Ju, M. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *International Conference on Automatic Face and Gesture Recognition (AFGR96)* (Killington, VT, October 1996), 38–44.

[114] S. Wachter and H. Nagel, "Tracking persons in monocular image sequences," *Comput. Vis. Image Understand.* **74**, 174–192 (1999).

[115] D. DiFranco, T. Cham, and J. Rehg, "Reconstruction of 3D figure motion from 2D correspondence," in *IEEE International Conference on Computer Vision (ICCV2001)*, Vol. 1, (Kauai, HI, December 2001), 307–314.

[116] R. Plankers and P. Fua, "Articulated soft objects for multiview shape and motion capture," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**, 1182–1187 (2003).

[117] J. Deutscher, A. Blake, and I. Reid, "Articulated body motion capture by annealed particle filtering," in *IEEE International Conference on Computer Vision and Pattern Recognition, (CVPR2000)*, Vol. 2 (Hilton Head, SC, June 2000), 126–133.

[118] C. Sminchisescu and B. Triggs, "Estimating articulated human motion with covariance scaled sampling," *Int. J. Robot. Res.* **22**, 371–393 (2003).

[119] R. L. Carceroni and K. N. Kutulakos, "Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance," *Int. J. Comput. Vis.* **49**, 175–214 (2002).

[120] Hanim-Humanoid Animation Working Group. Specifications for a standard humanoid. Available at http://www.h-anim.org/Specifications/HAnim1.1, Feb. 2, 2005, 2002.

[121] NASA. *Anthropometric Source Book. Vol. II.: A Handbook of Anthropometric Data.*

[122] Q. Delamarre and O. Faugeras, "3D articulated models and multi-view tracking with silhouettes," *Comput. Vis. Image Understanding* **74**, 174–192 (1999).

[123] D. M. Gavrila and L. S. Davis, "3D model-based tracking of humans in action: a multi-view approach," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR96)* (San Fransisco, CA, June 1996), 73–80.

[124] J. Kundert-Gibbs and P. Lee, *Mastering Maya 3* (Sybex Inc, 2001).

[125] H. Sidenbladh, M. Black, and L. Sigal, "Implicit probabilistic models of human motion for synthesis and tracking," in *European Conference on Computer Vision, (ECCV2002)*, Vol. 1 (Copenhagen, Denmark, May–June 2002), 784–800.

[126] V. Pavlovic, J. Rehg, T. Cham, and K. Murphy, "A dynamic bayesian approach to figure tracking using learned dynamical models," in *IEEE International Conference on Computer Vision, (ICCV99)* (Corfu, Greece, September 1999), 94–101.

[127] I. Biederman, "Recognition-by-components: A theory of human image understanding," *Psychol. Rev.* **94**, 115–147 (1987).

[128] J. Rehg and T. Kanade, "Visual tracking of high DOF articulated structures: an application to human hand tracking," in *European Conference on Computer Vision (ECCV94)*, Vol. 2 (Stockholm, Sweden, May 1994), 35–46.

[129] D. G. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Trans. Pattern Anal. Mach. Intell.* **13**, 441–450, (1991).

[130] E. Hildreth and D. Marr, "Theory of edge detection," in *Proceedings of Royal Society of London*, Vol. 207, 187–217 (1980).

[131] M. Jesus, A. J. Abrantes, and J. S. Marques, "Tracking the human body using multiple predictors," in *Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects* (London, UK, 2002), 155–164.

[132] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people," in *European Conference on Computer Vision (ECCV2002)*, Vol. 4 (Copenhagen, Denmark, May–June 2002), 700–714.

[133] G. McAllister, S. J. McKenna, and I. W. Ricketts, "MLESAC tracking with 2D revolute-prismatic articulated models," in *International Conference on Pattern Recognition (ICPR2002)* Vol. 2 (Quebec City, Canada, August 2002), 725–728.

[134] T. J. Cham and J. M. Rehg, "A multiple hypothesis approach to figure tracking," in *International Conference on Computer Vision and Pattern Recognition (CVPR99)*, Vol. 2 (Fort Collins, CO, June 1999), 239–245.

[135] E. Loutas, N. Nikolaidis, and I. Pitas, "A mutual information approach to articulated object tracking," in *IEEE International Symposium on Circuits and Systems (ISCAS2003)*, Vol. II (Bangkok, Thailand, May 2003), 672–675.

[136] M. Kolsch and M. Turk, "Fast 2D hand tracking with flocks of features and multi-cue integration," in *IEEE Workshop on Real-Time Vision for Human Computer Interaction (RTV4HCI 2004)* (Washington DC, June 2004).

[137] J. Y. Bouguet, Pyramidal implementation of the Lucas Kanade feature tracker, Technical report, Intel Corporation, Microprocessor Research Labs, OpenCV Documents (1999).

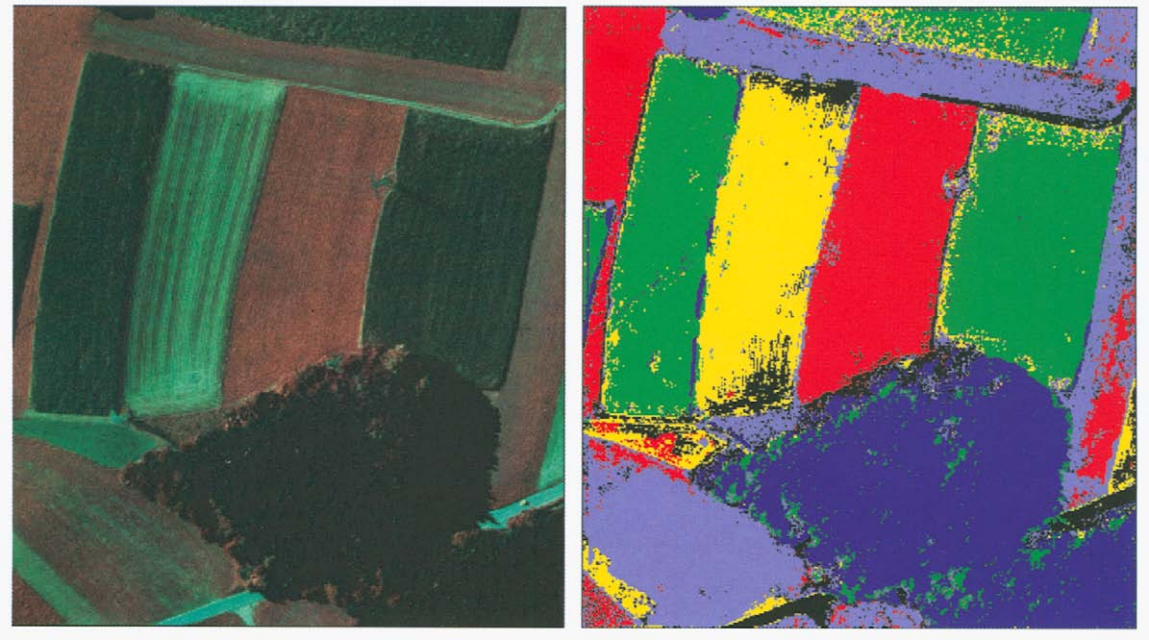

FIGURE 4.8.8 Segmentation of another aerial image, this time of a rural crop field area, using the same texture-based maximum likelihood procedure employed in Fig. 7.

FIGURE 4.11.1 Skin detection based on simple thresholding in the Hue-Saturation components of the HSV color space.

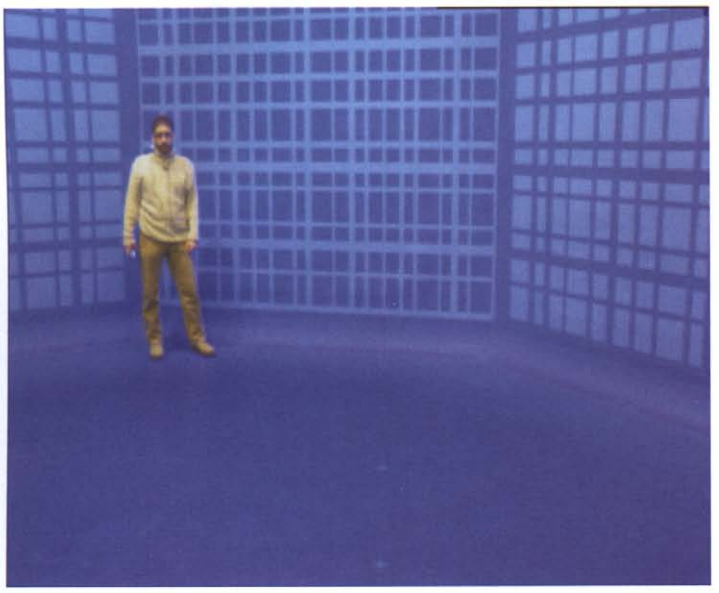

FIGURE 4.11.2 A typical environment for color-based object tracking using chroma keying.

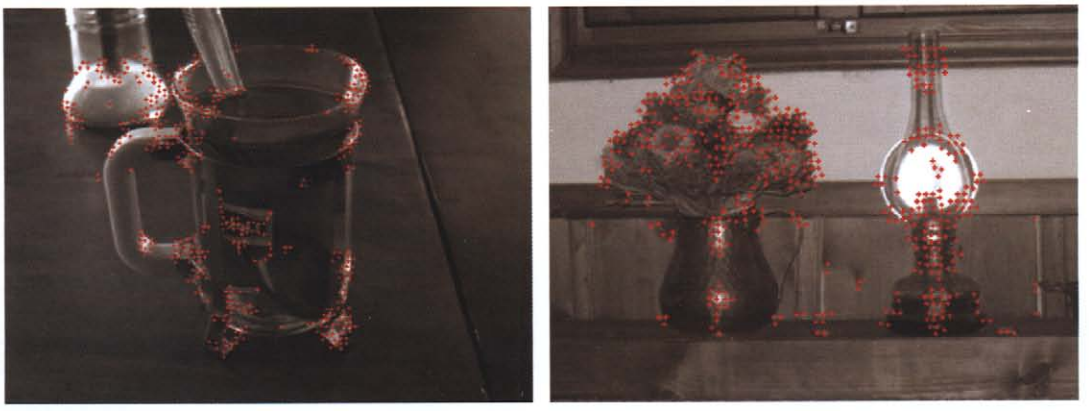

FIGURE 4.11.3 Feature selection based on [57].

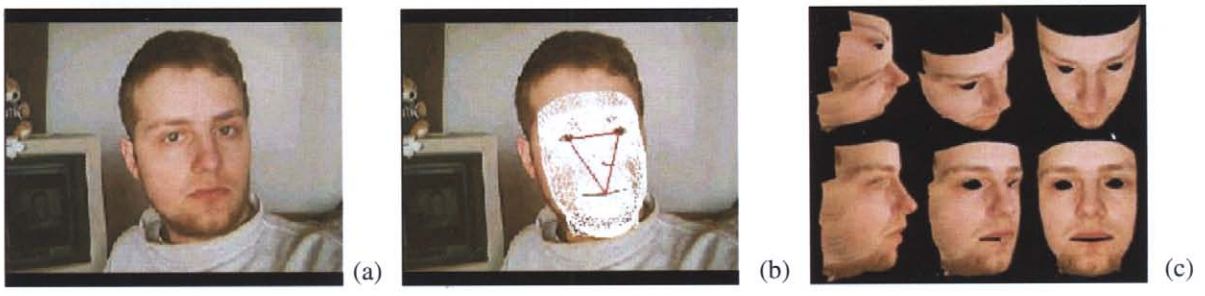

FIGURE 4.11.5 (A): A sample frame from a head and shoulders video sequence, (B) a generic face polygon model fitted on the video frame, and (C) the geometric model enriched with texture information.

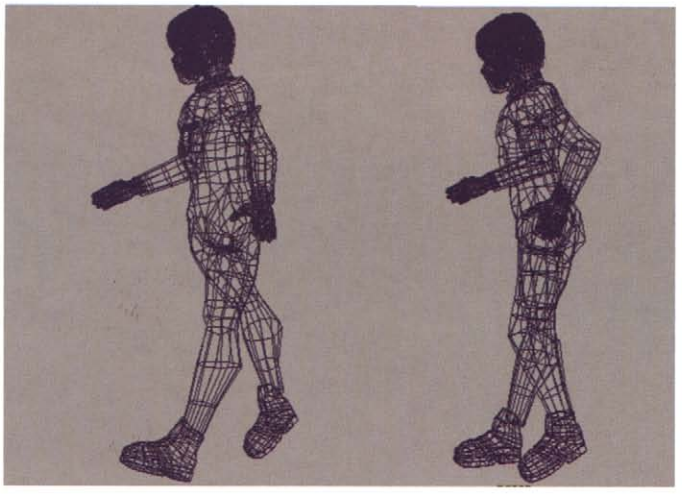

FIGURE 4.11.9 Polygonal representation of the human body, adapted from [124].

FIGURE 4.11.10 Cylinder-based volumetric representation of the human body.

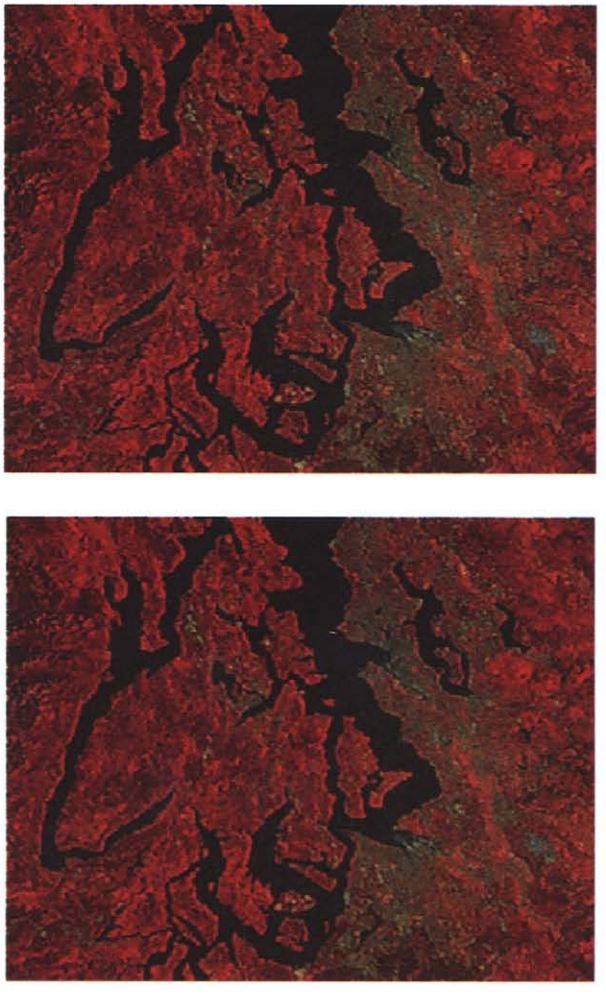

FIGURE 4.14.4 (top) SPOT multispectral image of the Seattle area, with additive Gaussian-distributed noise, $\sigma = 10$. (bottom) Vector distance dissimilarity diffusion result, using diffusion coefficient in (9).