1

WHAT IS VIDEO TRACKING?

1.1 INTRODUCTION

Capturing video is becoming increasingly easy. Machines that see and understand their environment already exist, and their development is accelerated by advances both in micro-electronics and in video analysis algorithms. Now, many opportunities have opened for the development of richer applications in various areas such as video surveillance, content creation, personal communications, robotics and natural human–machine interaction.

One fundamental feature essential for machines to see, understand and react to the environment is their capability to detect and track objects of interest. The process of estimating over time the location of one or more objects using a camera is referred to as *video tracking*. The rapid improvement both in quality and resolution of imaging sensors, and the dramatic increase in computational power in the past decade have favoured the creation of new algorithms and applications using video tracking.

Video Tracking: Theory and Practice. Emilio Maggio and Andrea Cavallaro

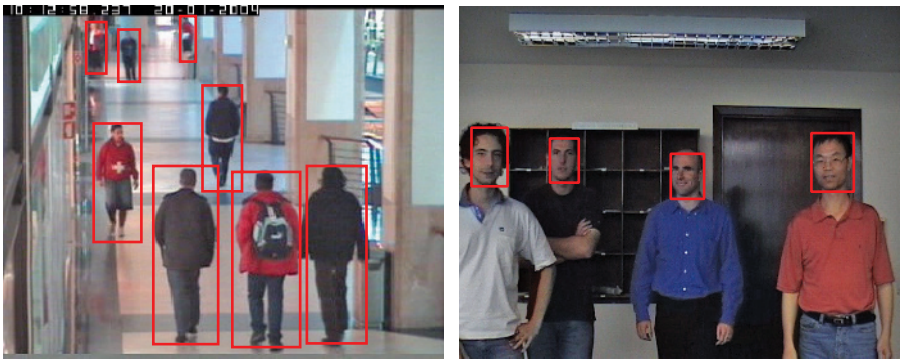

Figure 1.1 Examples of targets for video tracking: (left) people, (right) faces.

The definition of object of interest[1] depends on the specific application at hand. For example, in a building surveillance application, targets may be people (Figure 1.1 (left)[2]), whereas in an interactive gaming application, targets may be the hands or the face of a person (Figure 1.1 (right)).

This chapter covers the fundamental steps for the design of a tracker and provides the mathematical formulation for the video tracking problem.

1.2 THE DESIGN OF A VIDEO TRACKER

Video cameras capture information about objects of interest in the form of sets of image pixels. By modelling the relationship between the appearance of the target and its corresponding pixel values, a video tracker estimates the location of the object over time.

The relationship between an object and its image projection is very complex and may depend on more factors than just the position of the object itself, thus making video tracking a difficult task. In this section, we first discuss the main challenges in video tracking and then we review the main components into which a video-tracking algorithm can be decomposed.

1.2.1 Challenges

The main challenges that have to be taken into account when designing and operating a tracker are related to the similarity of appearance between the target and other objects in the scene, and to appearance variations of the target itself.

[1] Note that the terms *target* and *object of interest* will be used interchangeably in this book.
[2] Unmarked image from the CAVIAR dataset (project IST-2001-37540).

Figure 1.2 Examples of clutter in video tracking. Objects in the background (red boxes) may share similar colour (left) or shape (right) properties with the target and therefore distract the tracker from the desired object of interest (green boxes). Left: image from the Birchfield head-tracking dataset. Right: Surveillance scenario from PETS-2001 dataset.

The appearance of other objects and of the background may be similar to the appearance of the target and therefore may interfere with its observation. In such a case, image features extracted from non-target image areas may be difficult to discriminate from the features that we expect the target to generate. This phenomenon is known as *clutter*. Figure 1.2 shows an example of colour ambiguity that can distract a tracker from the real target. This challenge can be dealt with by using multiple features weighted by their reliability (see Chapter 6).

In addition to the tracking challenge due to clutter, video tracking is made difficult by changes of the target appearance in the image plane that are due to one or more of the following factors:

- *Changes in pose*. A moving target varies its appearance when projected onto the image plane, for example when rotating (Figure 1.3(a)–(b)).
- *Ambient illumination*. The direction, intensity and colour of the ambient light influence the appearance of the target. Moreover, changes in global illumination are often a challenge in outdoor scenes. For example, ambient light changes when clouds obscure the sun. Also, the angles between the light direction and the normals to the object surface vary with the object pose, thus affecting how we see the object through the camera lens.
- *Noise*. The image acquisition process introduces into the image signal a certain degree of noise, which depends on the quality of the sensor. Observations of the target may be corrupted and therefore affect the performance of the tracker.

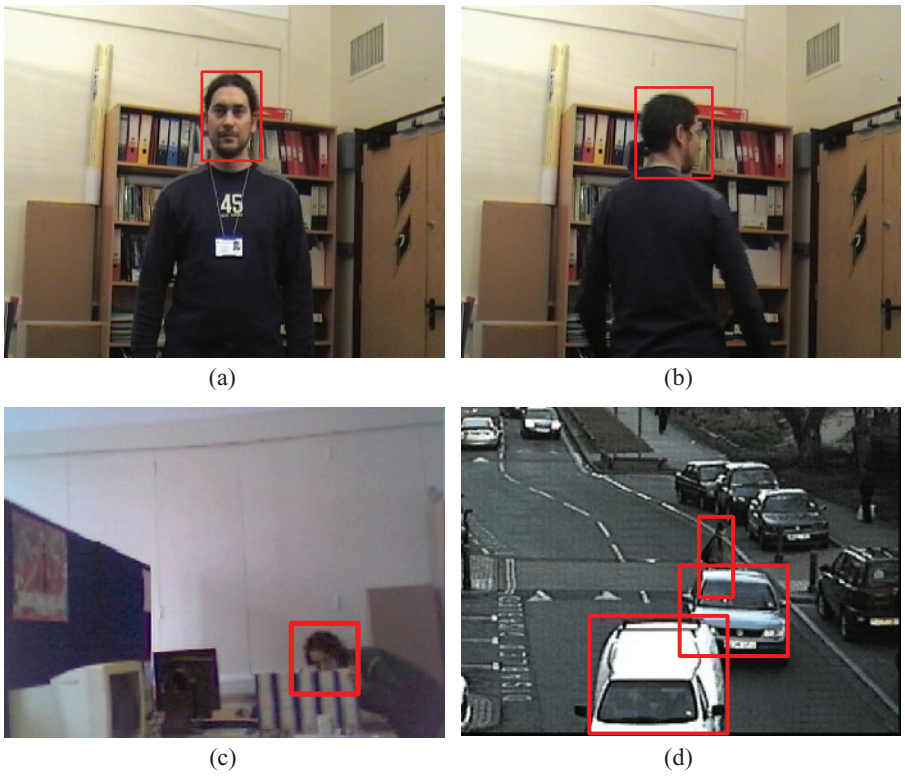

Figure 1.3 Examples of target appearance changes that make video tracking difficult. (a)–(b) A target (the head) changes its pose and therefore its appearance as seen by the camera. Bottom row: Two examples of target occlusions. (c) The view of the target is occluded by static objects in the scene. (d) The view of the target is occluded by another moving object in the scene; reproduced with permission of HOSDB.

- *Occlusions*. A target may fail to be observed when partially or totally occluded by other objects in the scene. Occlusions are usually due to:
  - a target moving behind a static object, such as a column, a wall, or a desk (Figure 1.3(c)), or
  - other moving objects obscuring the view of a target (Figure 1.3(d)).

  To address this challenge, different approaches can be applied that depend on the expected level of occlusion:
  - *Partial occlusions* that affect only a small portion of the target area can be dealt with by the target appearance model or by the

target detection algorithm itself (see Section 4.3). The invariance properties of some global feature representation methods (e.g. the histogram) are appropriate to deal with occlusions. Also, the replacement of a global representation with multiple localised features that encode information for a small region of the target may increase the robustness of a video tracker.

– Information on the target appearance is not sufficient to cope with *total occlusions*. In this challenging scenario track continuity can be achieved via higher-level reasoning or through multi-hypothesis methods that keep propagating the tracking hypotheses over time (see Section 5.3). Information about typical motion behaviours and pre-existing occlusion patterns can also be used to propagate the target trajectory in the absence of valid measurements. When the target reappears from the occlusion, the propagation of multiple tracking hypotheses and appearance modelling can provide the necessary cues to reinitialise a track.

A summary of the main challenges in video tracking is presented in Figure 1.4.

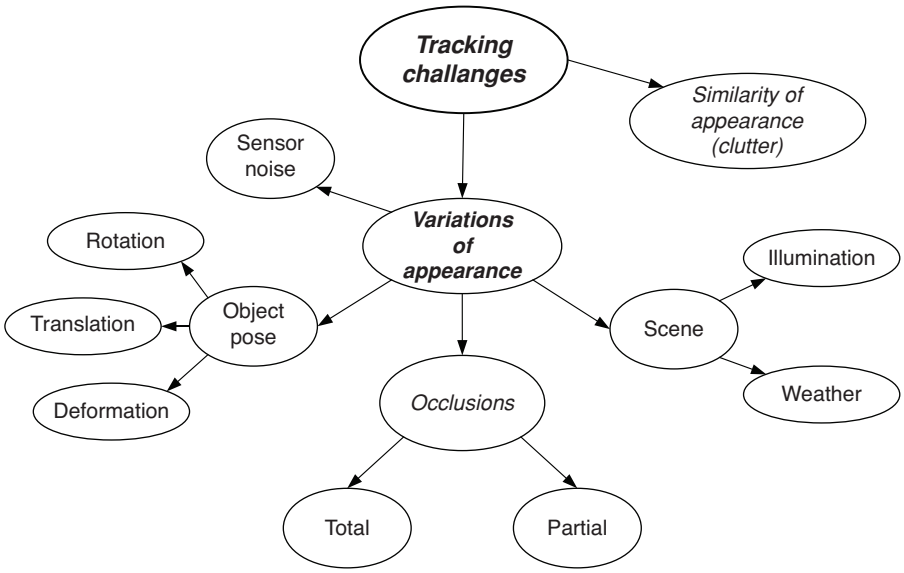

Figure 1.4 The main challenges in video tracking are due to temporal variations of the target appearance and to appearance similarity with other objects in the scene.

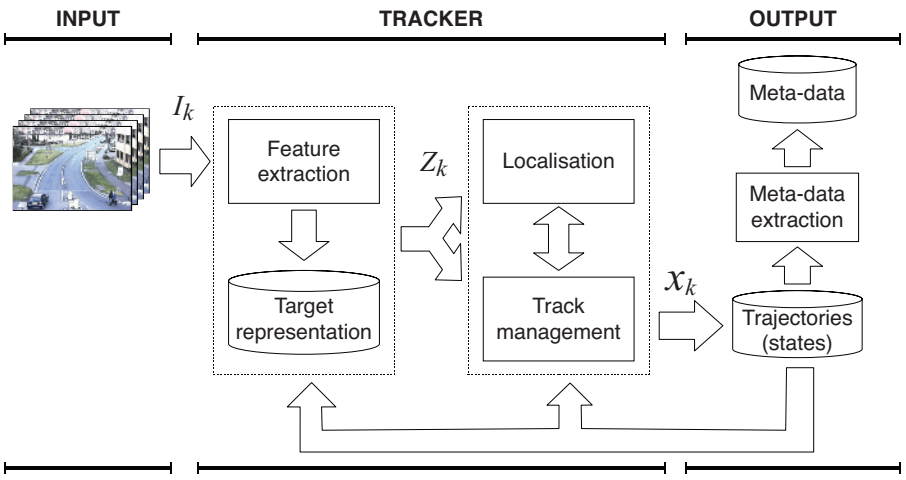

Figure 1.5 The video-tracking pipeline. The flow chart shows the main logical components of a tracking algorithm.

1.2.2 Main components

In order to address the challenges discussed in the previous section, we identify five main logical components of a video tracker (Figure 1.5):

1. The definition of a method to *extract* relevant information from an image area occupied by a target (Chapter 3). This method can be based on motion classification, change detection, object classification or simply on extracting low-level features such as colour or gradient (see Section 3.3), or mid-level features such as edges or interest points (see Section 3.4).
2. The definition of a representation for encoding the appearance and the shape of a target (the *state*). This representation defines the characteristics of the target to be used by the tracker (Chapter 4). In general, the representation is a trade-off between accuracy of the description (descriptiveness) and invariance: it should be descriptive enough to cope with clutter and to discriminate false targets, while allowing a certain degree of flexibility to cope with changes of target scale, pose, illumination and partial occlusions (see Sections 4.2 and 4.3).
3. The definition of a method to *propagate* the state of the target over time. This step recursively uses information from the feature extraction step or from the already available state estimates to form the trajectory (see Chapter 5). This task links different instances of the same object over time and has to compensate for occlusions, clutter, and local and global illumination changes.

4. The definition of a strategy to manage targets appearing and disappearing from the imaged scene. This step, also referred to as *track management*, initialises the track for an incoming object of interest and terminates the trajectory associated with a disappeared target (see Chapter 7). When a new target appears in the scene (*target birth*), the tracker must initialise a new trajectory. A target birth usually happens:
   - at the image boundaries (at the edge of the field of view of the camera),
   - at specific entry areas (e.g. doors),
   - in the far-field of the camera (when the size of the projection onto the image plane increases and the target becomes visible), or
   - when a target spawns from another target (e.g. a driver parking a car and then stepping out).

   Similarly, a trajectory must be terminated (*target death*) when the target:
   - leaves the field of view of the camera, or
   - disappears at a distance or inside another object (e.g. a building).

   In addition to the above, it is desirable to terminate a trajectory when the tracking performance is expected to degrade under a predefined level, thus generating a *track loss* condition (see Section 9.5.1).
5. The extraction of *meta-data* from the state in a compact and unambiguous form to be used by the specific application, such as video annotation, scene understanding and behaviour recognition. These applications will be described in Chapter 2.

In the next sections we will discuss in detail the first four components and specific solutions used in popular video trackers.

1.3 PROBLEM FORMULATION

This section introduces a formal general definition of the video-tracking problem that will be used throughout the book. We first formulate the single-target tracking problem and then extend the definition to multiple simultaneous target tracking.

1.3.1 Single-target tracking

Let $\mathbf{I} = \{I_k : k \in \mathbb{N}\}$ represent the frames of a video sequence, with $I_k \in E_I$ being the frame (image plane) at time k, defined in E_I, the space of all possible images.

Tracking a single target using monocular video can be formulated as the estimation of a time series

$$\mathbf{x} = \{x_k : \ k \in \mathbb{N}\} \tag{1.1}$$

over the set of discrete time instants indexed by k, based on the information in $\mathbf{I}$. The vectors $x_k \in E_s$ are the *states* of the target and E_s is the state space. The time series $\mathbf{x}$ is also known as the *trajectory* of the target in E_s. The information encoded in the state x_k depends on the application.

I_k may be mapped onto a feature (or observation) space E_o that highlights information relevant to the tracking problem. The observation generated by a target is encoded in $z_k \in E_o$. In general, E_o has a lower dimensionality than that of original image space, E_I (Figure 1.6).

The operations that are necessary to transform the image space E_I to the observation space E_o are referred to as *feature extraction* (see Chapter 3). Video trackers propagate the information in the state x_k over time using the extracted features. A localisation strategy defines how to use the image features to produce an estimate of the target state x_k (see Chapter 5).

We can group the information contained in x_k into three classes:

1. Information on the target *location* and *shape*. The positional and shape information depends on the type of object we want to track and on the amount (and quality) of the information we can extract from the images. We will discuss shape approximations for tracking in Section 4.2.

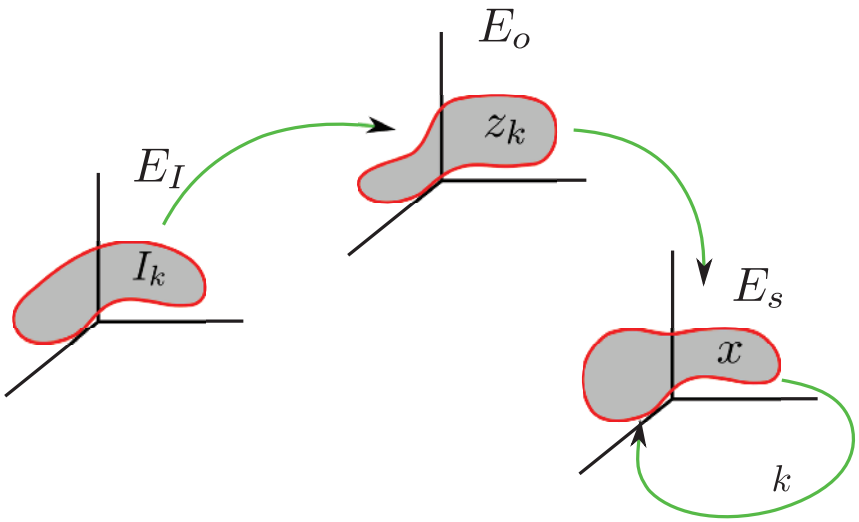

Figure 1.6 The flow of information between vector spaces in video tracking. The information extracted from the images is used to recursively estimate the state of the target (Key. E_I: the space of all possible images; E_o: feature or observation space; E_s: state space; k: time index).

2. Information on the target *appearance*. Encoding appearance information in the state helps in modelling appearance variations over time. We will cover target appearance representations in Section 4.3.

3. Information on the *temporal variation* of shape or appearance. The parameters of this third class are usually first- or higher-order derivatives of the other parameters, and are optional. The description of shape and appearance variations in the state will be discussed in Section 4.3.3.

Note that some elements of the state x_k may not be part of the final output required by the specific application. This extra information is used as it may be beneficial to the performance of the tracker itself. For example, tracking appearance variations through a set of state parameters may help in coping with out-of-plane rotations. Nevertheless, as adding parameters to the state increases the complexity of the estimator, it is usually advisable to keep the dimensionality of x_k as low as possible.

Figure 1.7 shows examples of states describing location and an approximation of the shape of a target. When the goal is tracking an object on the

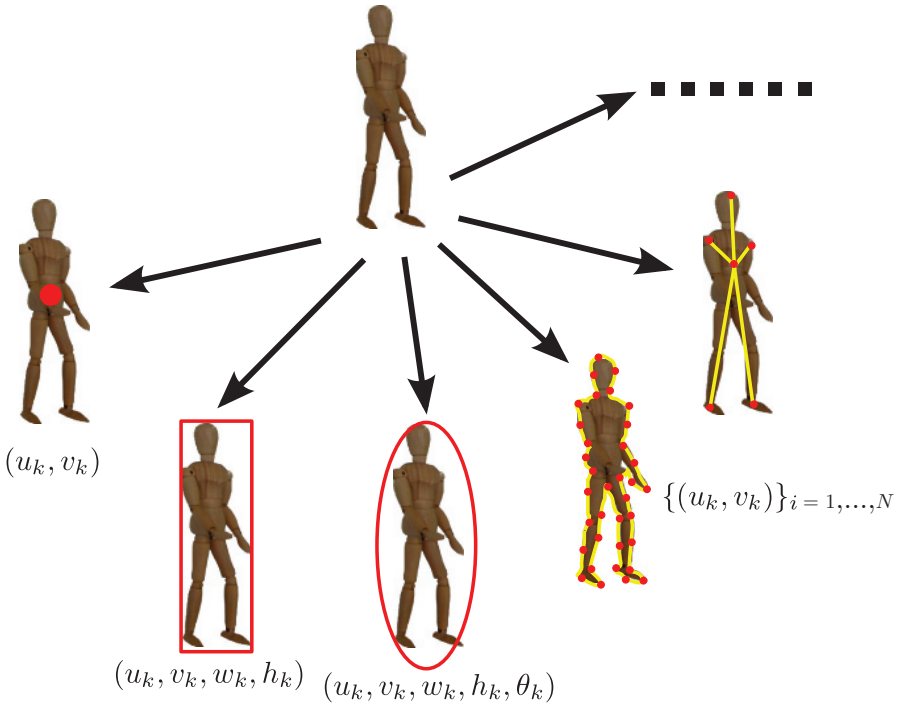

Figure 1.7 Example of state definitions for different video-tracking tasks.

image plane, the minimal form of x_k will represent the position of a point in I_k, described by its vertical and horizontal coordinates, that is

$$x_k = (u_k, v_k). \tag{1.2}$$

Similarly, one can bound the target area with a rectangle or ellipse, defining the state x_k as

$$x_k = (u_k, v_k, h_k, w_k, \theta_k), \tag{1.3}$$

where $y_k = (u_k, v_k)$ defines the centre, h_k the height, w_k the width and (optionally) θ_k the clockwise rotation. More complex representations such as chains of points on a contour can be used. Chapter 4 will provide a comprehensive overview of shape and appearance representations.

1.3.2 Multi-target tracking

When the goal of video tracking is to simultaneously track a time-varying number of targets, we can formulate the multi-target state X_k as the vector containing the concatenation of the parameters of the single-target states. If $M(k)$ is the number of targets in the scene at time k, and $\mathcal{F}(E)$ is the collection of all the finite subsets of E, then the multi-target state, X_k, is the set

$$X_k = \left\{x_{k,1}, \dots x_{k,M(k)}\right\} \in \mathcal{F}(E_s). \tag{1.4}$$

Some multi-target tracking algorithms solve the tracking problem as the association of multiple observations generated by an object detector over time (see Section 3.5). In this case the single-target observation vector $z_k \in E_o$ is composed of parameters defining a single detection. Typically these parameters represent the position and size of a bounding box. Similarly to Eq. (1.4) we can extend the definition of observation to multiple targets by defining the multi-target observation (or measurement) Z_k as the finite collection of the single target observations, that is

$$Z_k = \left\{z_{k,1}, \dots z_{k,N(k)}\right\} \in \mathcal{F}(E_o), \tag{1.5}$$

formed by the $N(k)$ observations. For simplicity, let us define the set of active trajectories at frame k as

$$\mathbf{X}_k = \left\{\mathbf{x}_{k,1}, \dots \mathbf{x}_{k,M(k)}\right\}, \tag{1.6}$$

and

$$\mathbf{Z}_k = \left\{\mathbf{z}_{k,1}, \dots \mathbf{z}_{k,M(k)}\right\}, \tag{1.7}$$

where each $\mathbf{z}_{k,i}$ is the set of observations assigned to the trajectory i up to time step k. Multi-target tracking algorithms will be covered extensively in Chapter 7.

1.3.3 Definitions

In order to evaluate the quality of the state estimation, we will quantify the accuracy and the precision of tracking results:

- The *accuracy* measures the closeness of the estimates to the real trajectory of the target.
- The *precision* measures the amount of variation in the trajectory estimates across repeated estimations that are obtained under the same conditions. In other words, the precision measures the degree of repeatability of a video-tracking result.

We will discuss the formal evaluation of the state estimation results in Chapter 9. The *state* definitions that we will use throughout the book are listed below. Similar definitions and notations are valid for representing the *measurements* extracted from the images:

- x: a vector in the state space
- x_k: state of a target at time index k
- $x_{a:b}$: the collection of states between time indexes a and b
- $\mathbf{x}$: the collection of states (i.e. the time series) forming the trajectory
- $\mathbf{x}_k$: the collection of all the states forming the trajectory of a target up to time index k
- $x_{k,j}$: the state of the jth target at time index k
- $x_{a:b,j}$: the collection of states of the jth target between time indexes a and b
- $\mathbf{x}_{k,j}$: collection of all the states forming the trajectory of the jth target up to time index k
- X: a set of vectors in the single-target state space
- $X_k = \{x_{k,1}, x_{k,2}, \ldots, x_{k,M(k)}\}$: the multi-target state at time k
- $X_{a:b} = \{x_{a:b,1}, x_{a:b,2}, \ldots\}$: the set of trajectory states between time indexes a and b
- $\mathbf{X}_k = \{\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots\}$: the set of trajectory states up to time k.

Figure 1.8 shows a pictorial representation of the symbols defined above and used for a multi-target state.

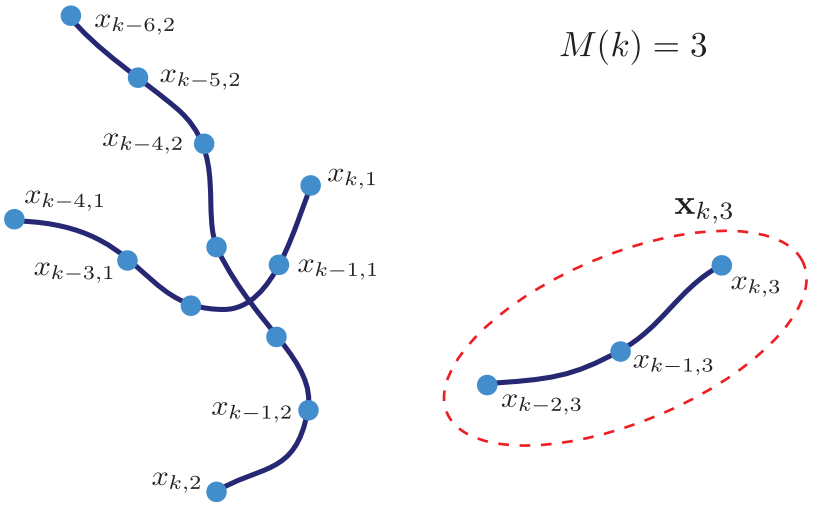

Figure 1.8 Pictorial representation of three trajectories: definitions of the symbols used for a multi-target state.

1.4 INTERACTIVE VERSUS AUTOMATED TRACKING

Based on the amount of interaction with the users to identify the object(s) of interest, video-tracking algorithms can be grouped into three classes, namely manual, interactive (or supervised) and automated (or unsupervised). These three classes are discussed below:

- Video tracking may be performed directly by the user. *Manual* tracking is used when a high accuracy, for example in the definition of the boundaries of the targets, is necessary. One example of application that requires manual tracking is in film production, when the contour of a character is selected and its evolution over time is defined by an operator, on each frame. Although this procedure allows for a good definition of the target position (or boundaries), it is very time consuming and cannot be extensively used for large volumes of visual data.
- *Automated* tracking uses a priori information about targets that is encoded in an algorithm. For example, an object detector can be used to initialise the tracker and/or to support the state estimation over time. Examples of automated tracking methods are those based on face detection and moving-object segmentation (see Section 3.5). Fully automated video-tracking techniques for real-world applications are still in their infancy, because translating the properties defining a generic target into algorithmic criteria is a difficult task in non-constrained scenes.
- *Interactive* (semi-automated) strategies are used as a trade-off between a fully automated tracker and a manual tracker. The principle at the

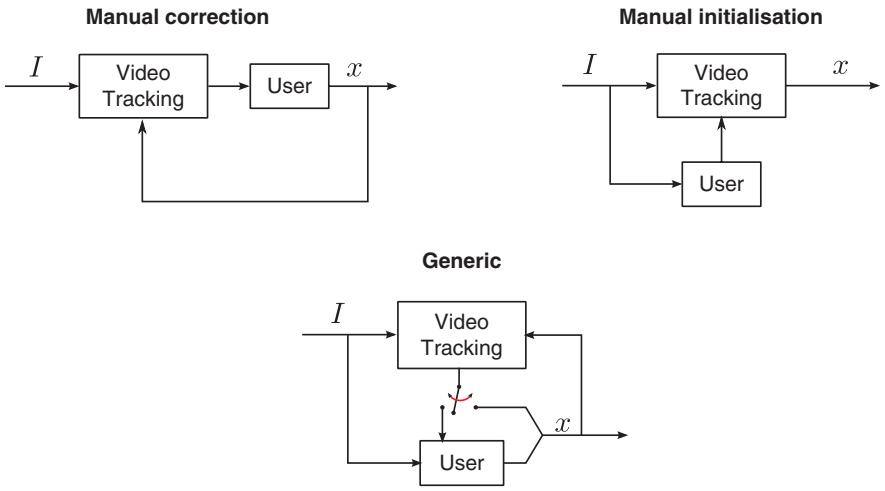

Figure 1.9 Automated and interactive tracking. Block diagrams representing different configurations enabling user interaction in video tracking.

basis of semi-automated techniques is the interaction of the user during some stages of the tracking process, where the information about the target area is provided directly by the user. Interactive tracking is used in *tag-and-track* applications (e.g. video editing and surveillance) when an operator manually initialises (selects) a target of interest that is then followed by the tracking algorithm. After the user provides the initial definition of the target, the tracker follows its temporal evolution in the subsequent frames thus propagating the initialisation information. This second phase can be either unsupervised, or the operator can still verify the quality of the tracking results and correct them if needed. This allows, for example, video editors or annotators to select the appearance of an object in a single frame and then to propagate these changes over time, thus saving the time-consuming task of manually editing each frame. Another example of an application is camera control for surveillance, when a person is selected by an operator and then automatically followed with a pan tilt and zoom (PTZ) camera.

Schematic representations of different types of interactive procedure are shown in Figure 1.9.

1.5 SUMMARY

In this chapter we introduced the concept of video tracking and discussed the major steps in the design of a tracker. We defined a tracking taxonomy based

on five building blocks that can be commonly identified in video-tracking algorithms. In this context, we highlighted the factors that make video tracking a difficult task. To cope with these factors, a tracker should have an invariant representation of the target or else adapt the representation over time.

Next, we introduced the definition and a generic problem formulation that is based on the concept of the state of a target. Moreover, we discussed a classification of tracking algorithms that is based on the amount of interaction with a user. Other classifications are possible as, for example, marker-based and markerless tracking or causal and non-causal tracking.

In the remainder of the book we will discuss the fundamental aspects of video-tracking algorithm design and the various implementation choices and trade-off decisions that are necessary to define accurate video trackers.