Chapter 9

Object Tracking – an Overview

The output of the background subtraction algorithm, described in the previous chapter, is a classification (segmentation) of pixels into foreground pixels (those belonging to moving objects) and background pixels. This is performed independently for each image of the video stream. The next step is to group the foreground pixels into actual objects, followed by the process of tracking these objects from one image frame to the next – that is, establishing object identity over space and time.

This chapter gives a brief review of existing tracking methods – some of which are not suitable for the non-linear geometry of catadioptric cameras. Then, the methods that were used for the OmniTracking application are introduced.

9.1 Object Tracking

Real-time object tracking can be described as a *correspondence problem*, and involves finding which object in an image frame corresponds to which object in the next frame[1] of a video stream [TRUC98 §8.1.2]. (For now, it will be assumed that the 'object' that appears in the image has a one-to-one correspondence with the real world object – failure of this assumption will be examined later on). Within the context of tracking, objects are sometimes also referred to as *targets*.

Normally, the time difference between two successive frames is very small, meaning that, likewise, the changes visible from one frame to the other should be limited. This

[1] The word 'next' frame here refers to the processing rate used by the program. For example, the original PETS datasets have a frame rate of 25 frames per second (fps), but the OmniTracking application reads every 5th frame, therefore having a processing rate of 5 fps. The 'next' frame is in this case the one at time t+5, that is, 5 frames apart.

allows the use of certain *temporal constraints* that simplify the correspondence problem. For example, an object is expected to move only slightly from one frame to the next, so at time *t+1*, it should remain spatially near it's position at time *t*. Other things that should change slowly are: the object's apparent velocity and direction of motion (*smoothness of motion* constraint), and the appearance of the object (*similarity* constraint). These can be used when matching objects from one frame to the other [JAIN95 §14.6; SONK §14.2].

In the case where the objects to be tracked are known beforehand (for example, using a known geometric model), then the matching problem becomes one of generating a hypothesis about the object, based on its model and information gathered at time *t*, and testing this hypothesis against the next image frame *t+1*. This approach to object tracking is called the *top-down* approach. And the other one mentioned before is called the *bottom-up* approach because it starts with the images segmented into a number of objects and uses only the (low-level) information available in those images [SONK93 §8.1.2].

For the bottom-up approach, where the model and exact type of objects to be tracked may not be known, it may still be possible to place some limits on what class of objects are expected to be present or what the tracking program is interested in (for example, a program used for tracking cars and ignoring the rest). This information can be used to restrict the number of potential possibilities during the matching process.

Normally, the process of tracking an object involves building some 'representation' of an object, which can then act as a description for that object. Two ways of doing this is through the use of statistical modelling or by extracting some 'features' directly from the image [ELGA02]. The latter case is used often in bottom-up tracking, and the set of features representing an object is usually called a *feature vector*. Object tracking then consists of matching the features of an object in an image frame with the features of potential matches in the next frame.

It follows that the choice of features should allow the tracker to discriminate between different objects – that is, a feature vector should have some measure of difference associated with it that can be used to compare and identify objects [OWEN02]. Ideally, the features chosen should also remain relatively constant throughout the object's lifetime and be robust to changes in the appearance of the object.

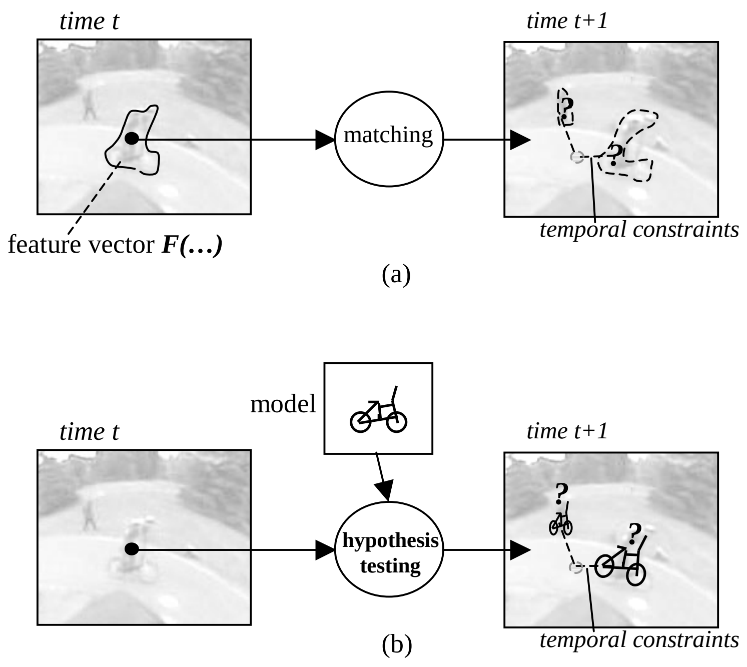

Figure 9.1: (a) Bottom-Up and (b) Top-Down Object Tracking.

Several features could be used, for example the object's boundary, colour, motion, etc. These features could be global (such as the object's colour[2]) or local sub-features could be used that act as representative points for that object. Examples of sub-features are *corner points* [SHI94]. The use of sub-features may require some form of grouping into objects if the matching is done on the feature-level rather than on the object-level.

Another fact that influences the choice of features is whether *rigid* or *non-rigid* objects are being tracked. A rigid object in the world (for example a car) is one in which all its sub-parts move in an identical way and its motion in an image sequence can be modelled by an affine transformation [FORS02 §14.5]. Non-rigid objects, such as people, move in more complex ways, with different parts moving at different rates and in different directions. For rigid objects, if using sub-features, then motion-based constraints can be applied, since the feature points are expected to move rigidly together and have a common motion. On the other hand for tracking non-rigid objects, flexible object models may need to be used.

[2] Colour may not always be a global feature, in the case of multi-coloured objects which consist of sub-regions having different colours.

9.2 Tracking Problems

As mentioned at the beginning of §9.1, no distinction has been made yet between the object that exists in the real world and the object as it appears in the image – it was assumed so far that there's a direct relationship between the two.

To distinguish between them, one can talk of a 'world object', that is, the physical entity (example, a bicycle) that the tracking program is interested in, and the 'image object' for the appearance of the object as seen in the image. A world object is projected onto the image plane as a spatially contiguous set of pixels (see Figure 9.2). And in the ideal case, there is a one-to-one correspondence between a world object and the image object. But this is not always the case, due to factors like occlusion and merging, and due to errors generated during the motion detection phase[3].

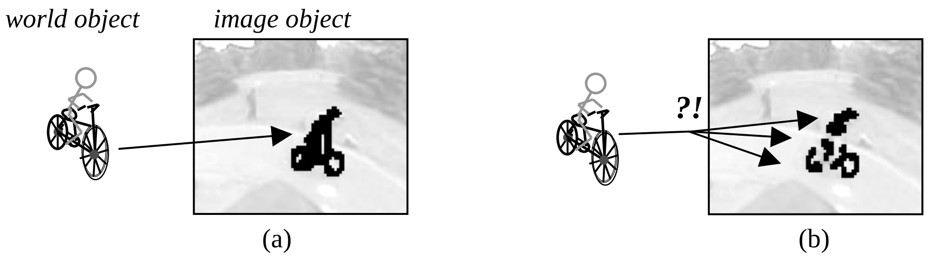

Figure 9.2: World Objects and Image Objects. (a) A one-to-one relationship exists between the world object and the image object; (b) In the presence of detection errors or factors like occlusion, there is no longer a one-to-one relationship (in this case, due to object fragmentation).

These problems affect the tracking algorithm, since errors in the image tend to propagate to the features used for matching during object tracking. And occlusion and object merging can cause the tracker to lose the object completely. Therefore, a very important characteristic of any object tracker is that it is robust to these problems and that it is able to track objects through occlusion, merging and the features used to represent objects must be relatively unaffected by motion detection errors.

[3] In the rest of this document, most often the word 'object' will be used without explicitly specifying which of the two meanings it is being used for, and its meaning can be assumed from the context in which it is used.

For example, tracking methods that use sub-features or colour-based trackers are normally immune to partial occlusion [GROV98]. Other trackers do not use the motion detection results directly, but only use them to serve as a 'guide' for the initial estimates (example [LIPT98]) – these are less sensitive to motion detection errors such as object fragmentation.

Other factors that can make life complicated for object trackers depend on the scene-camera geometry. For example, changes in the appearance of objects due to variations in the intensity and orientation of incident light, when objects move into areas covered with shadow, objects that may rotate so exposing different textured sides into view, etc. [RASM96]. If the matching features are not robust enough to these changes, then the matching process may fail.

Tracking of non-rigid objects (for example, people) can also be problematic, because their visual appearance can be hard to define, it can change very rapidly from one frame to the next and may change in complex ways [INTI95]. This can be seen in Figure 9.3, where the outline of the person undergoes quite rapid changes in shape and size – the boundary is not a good feature to track with, in this case[4].

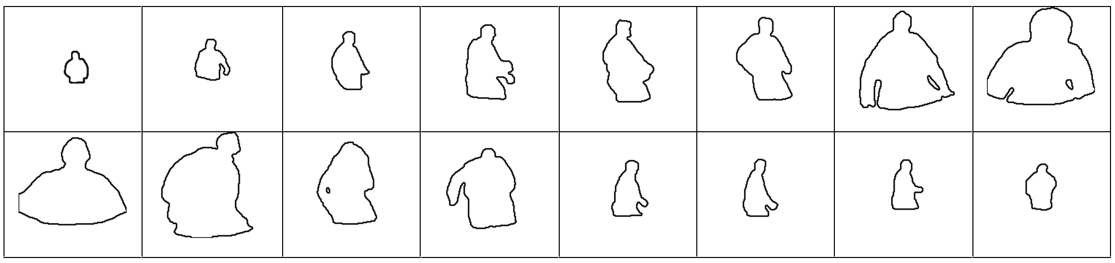
Figure 9.3: Non-rigid Object Boundary Changes. (Appearance of a person in the PETS-ICVS dataset).

Trackers that don't use pre-defined object models or knowledge about the objects to expect, assume that any new moving region that appears in the image is a new single 'object' and then try to maintain and track that object in the initial form it first appeared in. This is a reasonable assumption, as generally objects tend to be distinct from each other when they first appear. But it can happen, that two world objects appear in the camera's field-of-view close to each other and keep moving in a similar

[4] Also, in general using the boundary is not a good idea to track with, as this requires the motion detection module to detect and locate boundaries accurately.

way – the trackers will most probably track them as one object until they diverge from each other or disappear from view.

9.3 Different Tracking Methods

There are several different tracking methods in computer vision, and this section mentions briefly some of the most common ones.

- **Model-based Tracking** methods start with a 2D or 3D geometric model of known objects and uses hypothesis testing to check if any of these objects are present in the image or not. These are mostly applied for tracking rigid objects. An example of model-based tracking applied to vehicle tracking is given in [FORS02 §19.4.1], while [FIEG97] models objects as a set of 2D sub-regions (rectangles) with a known layout for head tracking.
- **Template Matching (Region Correlation)** techniques solve the correspondence problem by using templates (2D image segments), containing the known brightness values of the object (or parts of it) and then comparing these templates against potential locations in the next frame. Comparison is done by placing the template at the position to be tested, and computing the dissimilarity using sum of squared errors or normalised cross-correlation [JAIN95 §15.6.1]. The template is normally updated to reflect the most recent state of the object. These methods are limited to tracking objects that undergo affine motion only and are very sensitive to size changes and brightness variations. Some tracking applications using adaptive template matching are those by [LIPT98] and [INTI95].
- **Optical Flow** methods use the apparent motion of the image brightness pattern of an object to determine its motion. They use object features consisting of some points of interest, and compute their optical flow to determine where the object will be located in the next frame. These methods tend to be computationally expensive and require that the object is continually moving [TRUC98 §8.3]. (See also §8.1 for the use of optical flow as a motion detector).
- **Kalman Filter** based methods are estimator-based methods, where for some given object point features in one frame, their probable positions in the next frame are estimated, using a prediction-correction estimation cycle [TRUC98

§8.4]. These methods are limited in the type of motion that can be modelled (for example, Kalman filters are not useful for non-rigid human motion). [FORS02 §19.4.1] mentions some applications that use Kalman filters for vehicle tracking.

- **Particle Filters (Condensation)** are also estimator-based techniques that, unlike the Kalman filter, can be used to model and track more complex motion such as that of people which are modelled as a collection of body segments connected with rigid transformations [FORS02 §20.3]. This technique is also known as the *Condensation* method (*con*ditional *dens*ity propag*ation*) [BLAK96]. Particle filter based trackers require the use of a known (motion) model for the objects. [NUMM03] is an application example that use particle filters together with colour features to track people.

- **Active Contour (Snake)** based methods can be used for tracking non-rigid objects with a complex varying outline. They model the boundary of objects as an energy-minimising parametric curve (snake) that is controlled by its internal energy (the snake's configuration) and external energy (outside forces affecting the snake), and then try to find the location that minimises the energy [SONK93 §14]. For tracking purposes, the external forces could be defined in such a way that the snake is attracted to the boundary of objects, making the method useful for tracking deformable objects, but with the requirement that an accurate boundary has been found by the motion detection algorithm.

- **Feature Tracking** methods normally do the tracking process on the level of sub-features extracted from the image, that is, at a local level. These sub-features are grouped into objects using some constraints or similarity measures and the object's motion is then the global result of the sub-features' motions. For example, [SHI94] defines a tracking method that uses corner point features extracted from the image, which are then individually tracked using an affine motion model. The advantage of using sub-features is that these are usually robust to partial occlusion.

- **Colour Tracking** methods use the colour of objects as the feature to track with, because generally colour offers robustness to partial occlusion and object distortions. Colour trackers can be used to track both single-coloured objects and also for multi-coloured objects. Colour trackers normally use of two types of models: The first model the colour of objects non-parametrically, usually by using a colour histogram and using some similarity measure for comparing two objects. The second type models the object's colour parametrically, for example, by using a probability mixture mode. Colour trackers can suffer from object loss if an object has a similar colour as the background. Examples of colour-based tracking applications are [RASM96; GROV98; RAJA98].

- **Blob Tracking** methods work directly with the output obtained from motion detection, that is, rely on the pixel segmentation result of motion detection. The pixels belonging to moving objects are grouped together into *blobs* using connected components and these blobs are then matched from one frame to the next using some features extracted from the image and using temporal constraints. Blob-based tracking methods have the advantage of being very fast, but they suffer from object loss and mismatches in the presence of problems like occlusion and object merging. Examples of blob-based tracking applications are [INTI95; BOUL98b; COL99].

9.4 Tracking for Omnidirectional Applications

Omnidirectional-based applications place some constraints on what type of tracking methods can be used. As mentioned in §4.4, catadioptric cameras generate non-linear image distortions, which means that tracking methods that are based on linear or affine motion models (such as the Kalman filter, and optical flow[5]) cannot be used. Also, methods like template matching use a rectilinear template based on the assumption that the local neighbourhood of a point is linear – this is not the case in omnidirectional images.

For example, Figure 9.4 shows an application using a corner-based tracking method, called KLT[6], being used to track vehicles. The KLT tracker [SHI94] locates feature points (corners) in the image that have a high degree of texture content within some small window centred on that point (and so termed "good features to track"). Tracking

[5] Although optical flow is still used in some applications like robot navigation, where the wide field-of-view outweighs the inaccuracy due to the non-linear image geometry. The advantages being the reduction of the aperture problem for optical flow, the focus of expansion and focus of contraction are always visible, and the image obeys the total intensity conservation constraint (as objects don't suddenly appear or disappear when they go out of the field-of-view, as happens in conventional cameras).

[6] Kanade-Lucas-Tomasi (KLT) tracker.

of the corners then uses a combination of estimating the affine deformation matrix needed to measure the window similarity of the corner from the first frame to the current one and a linear translation model to slide the window from the current frame to the next. Figure 9.4(b) shows the window of one corner over 2 successive frames, with a roughly-drawn overlay showing the affine deformation of the edges. This method will fail for omnidirectional images due to the assumptions of local areas undergoing affine deformations and the use of a linear translation model. And Figure 9.4(c) gives an approximate idea of how the local neighbourhood of an omnidirectional image looks like.

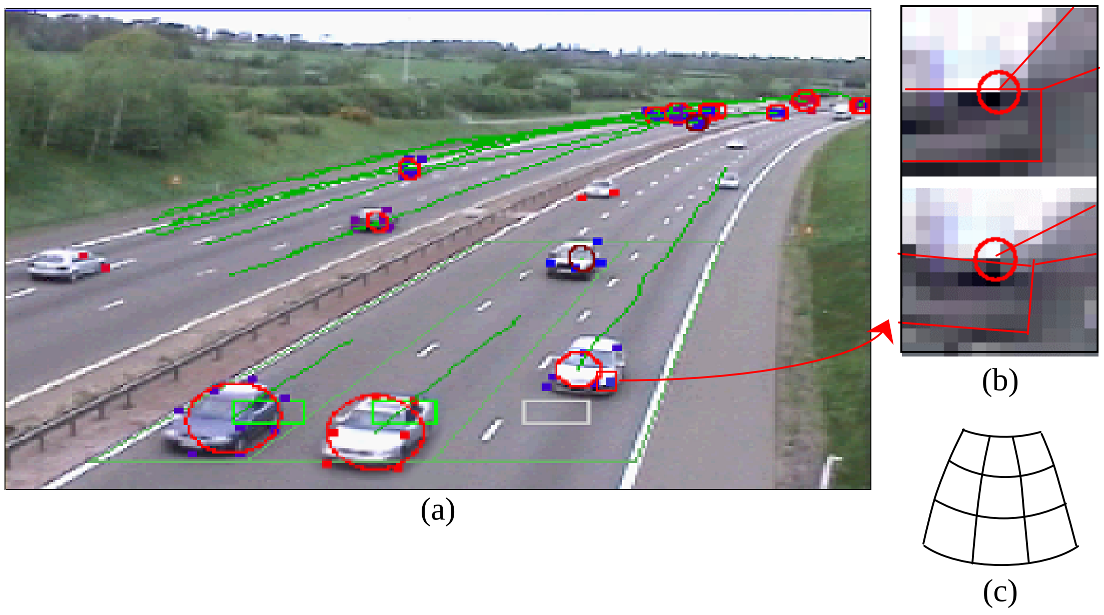

Figure 9.4: Corner Tracking and its affine and linear translation model. The picture shown in (a) is a screenshot from a traffic analysis system, a project that the author worked on while writing this thesis.

In addition, certain assumptions like objects appearing in an “upright” position cannot be used in omnidirectional images, meaning that object orientation and shape-related features like those used by [FIEG97], cannot be applied in this case.

One can dewarp the omnidirectional image and then apply the tracking procedure on this geometrically correct version. But this introduces errors due to interpolation, and there’s the additional problem of tracking objects across the artificially-introduced boundaries of the dewarped image.

9.5 Chosen Methods – an Introduction

As in most generic visual surveillance systems, the OmniTracking application does not use any pre-defined models or information about which objects can appear within its field-of-view. In the datasets used by this program, the targets in the indoor sequence are the people attending the meeting. But for the outdoor dataset, there are mainly two different types of objects – persons and cars. This means that a bottom-up approach must be adopted with object information extracted directly from the images.

Two different tracking methods were attempted: blob tracking and colour-based tracking. These are discussed separately in the next two chapters, followed by a comparison between their results. In general, colour-based tracking methods give better results than blob-based tracking and are more robust to partial occlusion and variations in an object's appearance. But blob-tracking methods are simpler and usually faster.

The OmniTracking program can be configured to run with either of the two methods – by default it is set to use colour tracking, but this can be changed in the initialisation file. The reason for including both, apart from being able to compare the two methods, is that one can envisage situations where speed may be more important than tracking accuracy. Alternatively, the blob-tracking method may be used in conjunction with other higher-level algorithms or more accurate tracking algorithms, by providing rough estimates of where the likely objects can be found.