Chapter 12

3D Localisation and High-Level Processing

This chapter describes how the results obtained from the moving object tracking phase are used for estimating the 3D location of objects, based on the assumption of motion along the ground plane. The object tracking results are also used for creating 'virtual cameras' that automatically track objects as they move across the omnidirectional camera's field-of-view. Finally, the program performs some rudimentary summarisation of the tracking results and saves the output to disk.

12.1 3D Localisation

Determining the 3D position of objects from images is the main purpose of the field of *stereo vision*, where the environment is observed from two (or more) different viewpoints, and from the acquired images, the 3D structure of the surrounding environment can be inferred [TRUC98 §7.1]. When only one camera is available, a limited amount of 3D information can be extracted from the image if some domain constraints are known. A commonly used constraint is the assumption that motion takes place on some planar surface, normally called the *ground-plane*. This assumption is not unreasonable in many situations because most man-made environments consist of planar terrains (for example, roads, rooms).

For linear cameras, estimating the position by using the ground plane is normally achieved through the simple use of a *projective transformation* (known also as a *homography*) [FORS02 §15.1.2]. This defines a mapping from the 2D ground-plane (in the 3D world) to the 2D appearance of the ground-plane in the image. Creating this

transformation, requires the knowledge of at least 4 world points and their corresponding points in the image.

The geometry of catadioptric cameras is more complicated than that of linear cameras, and 2D planar surfaces in the world are not projected into a planar (linear) representation by a catadioptric camera (see §6.4) – therefore projective transformations cannot be used. But the problem of 3D localisation can be simplified for omnidirectional images if another constraint is used in addition to that of the ground-plane. This constraint is based on the assumption that the omnidirectional camera's axis is oriented perpendicularly to the ground-plane. This is a reasonable assumption because catadioptric cameras have a 360° 'horizontal' field-of-view, which in most set-ups, is made to coincide with the horizon.

Distance is estimated by finding the 'lowest' point of an object in the omnidirectional image (which by the ground-plane constraint, should be the point of contact of the object with the ground). This image point is then back-projected from the image to the ground-plane using the single viewpoint constraint of the catadioptric camera (§2.4) and the mirror equation – this is illustrated in Figure 12.1 below. Knowing the position of one world point and its equivalent image position is then sufficient for 3D localisation[1]. The back-projection calculation is performed through a pre-calculated look-up table, created during the calibration phase of the program (§7.4.4).

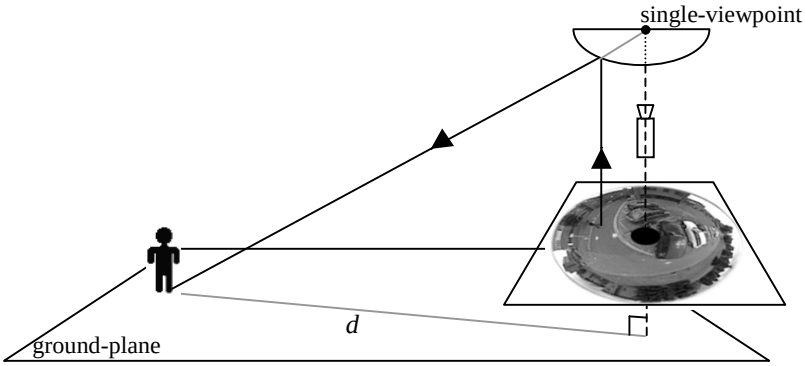
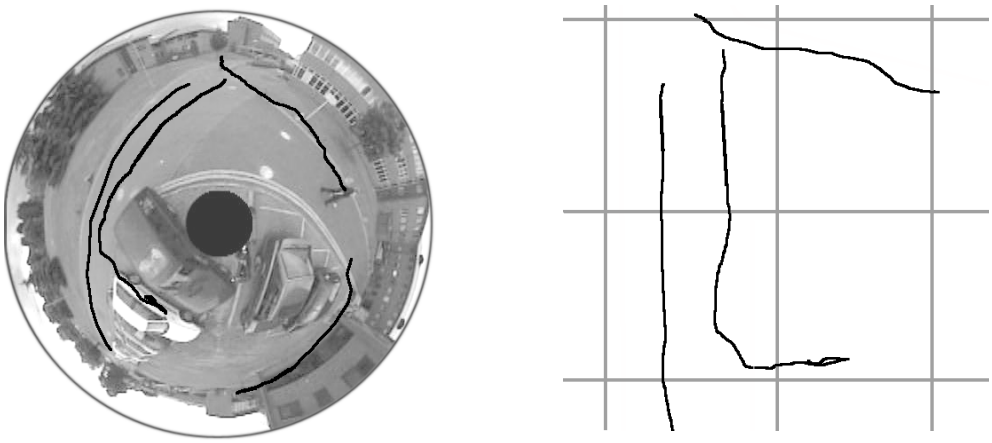


Figure 12.1: 3D localisation by means of back-projection from omnidirectional images.

Figure 12.2(a) shows the estimated 3D path of several objects as they move across the field-of-view of the camera in the PETS2001 dataset. The equivalent path as seen in the omnidirectional image is also given in Figure 12.2(b). Inaccuracies in the 3D

[1] Without knowledge of this world point, relative distance estimates (depth) can still be obtained by back-projection.

positions arise whenever the ground-plane constraint is violated, either because the object's contact point with the ground is not visible (occluded by background elements or other objects), or because of noise from the background subtraction and tracking algorithms.



(a) (b)

Figure 12.2: Results of 3D Localisation for the PETS2001 dataset

3D localisation was also attempted for the PETS-ICVS dataset. But in this dataset, the ground-plane is not visible and there is no useful 'contact point'. An attempt was made based on detecting the height of the persons as they move across the room and using an assumption of an average height, but this proved to be inaccurate because the persons are sometimes standing up and sometimes sitting, so the height is not a reliable 3D estimator. Unfortunately, no other method could be found for extracting accurate depth information without resorting to high-level domain knowledge.

12.2 Automatic Target Tracking with Virtual Cameras

One of the advantages of omnidirectional cameras is their ability to observe large fields-of-view and track objects simultaneously and in different parts of the image. This is an important property for monitoring and surveillance systems. In many of these systems, the end result is intended to be viewed by human beings and therefore it would be a useful feature if the system is able to display the targets as they are being tracked in a form best suited for human viewing. §6.5 describes how perspective views can be generated from omnidirectional images and how this was

implemented for the OmniTracking application. An observation was made in §2.4 that the process of generating perspective views is similar to placing a 'virtual' perspective camera at the omnidirectional camera's single viewpoint, and a control model was defined in §6.6.1 that allows the user to manually change the viewpoint of the virtual camera at run-time. With the addition of moving object detection and tracking algorithms in §§8-11, it is now possible to allow these virtual cameras to be controlled automatically by the program itself – a process that will be called *automatic target tracking* in this chapter.

The OmniTracking application supports two different modes of automatic target tracking – one is completely automatic and the other is manually-initiated automatic tracking. The algorithm that implements these two tracking methods is called the *camera controller*.

12.2.1 Automatic Target Tracking

The behaviour of the camera controller is governed by the idea that there could be many objects to track at any one moment in time, while on the other hand, the virtual cameras constitute a limited resource. The camera controller keeps an internal queue of objects that can potentially be tracked and when a virtual camera becomes available, the first object on the queue is allocated to this camera.

When the OmniTracking program starts, a pre-defined number (user-configurable) of virtual cameras are automatically opened and made available for use by the camera controller algorithm. In addition, while the program is running, the user can open new virtual cameras and these will also be added to the pool of virtual cameras that the camera controller can make use of. The user can also decide to stop a virtual camera from being used by the camera controller (by 'locking' the window) or return a locked window back to the virtual camera pool ('unlocking' the window).

When assigning an object to a free virtual camera, the camera controller selects the first object it finds in the queue. Currently no attempt is made at sorting the objects on the queue according to some significance measure – this is a limitation of the current implementation.

12.2.2 Manually-Initiated Automatic Target Tracking

When this method is chosen, the camera controller waits until the user indicates that a particular object is to be tracked. Then the camera controller finds the first free virtual camera and assigns it to track the object. The user selects objects to be tracked, by clicking with the mouse on the omnidirectional image.

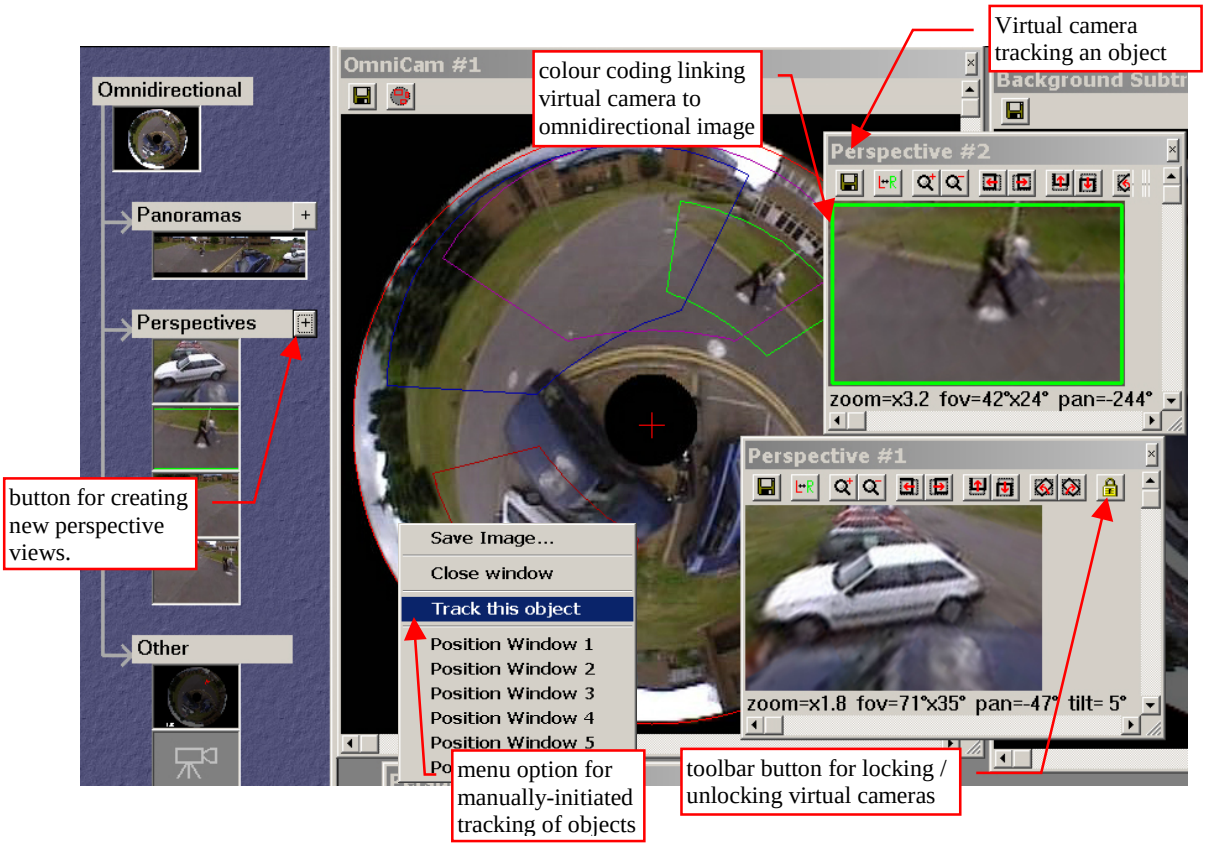


Figure 12.3: Different ways of automatically tracking objects

12.2.3 Automatic Camera Control

The camera controller uses the object's image centroid and spatial extent (bounding box) to set the control parameters of the virtual perspective camera, that is, to set the pan angle, tilt and zoom (see §6.6.1). It tries to keep the virtual camera centred on the object and adjusts the zoom factor so that the object fills a reasonable part of the view (vertical field-of-view between 1.5× to 2.5× object's height).

Because of the presence of noise and errors generated by motion detection, relying only on an object's centroid and size creates jittery camera motion that irritates the

user. For this reason, the camera controller uses a method for virtual camera control based on simulating the *inertia* of physical cameras. Each of the virtual camera controls is described in terms of an acceleration factor that is continually updated. For example, when an object starts moving, the camera takes some time to catch-up with the object, thus creating a smooth transition from a stationary state to a moving state.

12.3 Tracking History

The final node in the program's processing pipeline is the history node (see Figure 5.4), which receives the tracking results generated by the blob or colour tracker nodes and saves the output to disk. The output is in the form of a set of HTML files[2], with one history file (page) generated for each detected object and a global history file acting like the main index page. HTML was chosen because of its simple format, the ease of linking pages together and support for embedding images. Information about short-lived objects (which most probably are due to noise) is not saved to disk – currently, the program uses a simple threshold to identify these objects. For the other objects, the node saves a sequence of 'snapshots' of the object, with a new image added to the history file whenever the object moves a large enough distance. The image in which the object had the largest size, is used for the object's image in the main page – the assumption being that this image most probably provides the best representation of the object. A sample history from the PETS-ICVS dataset is shown in Figure 12.4.

[2] Hyper-Text Markup Language (HTML).

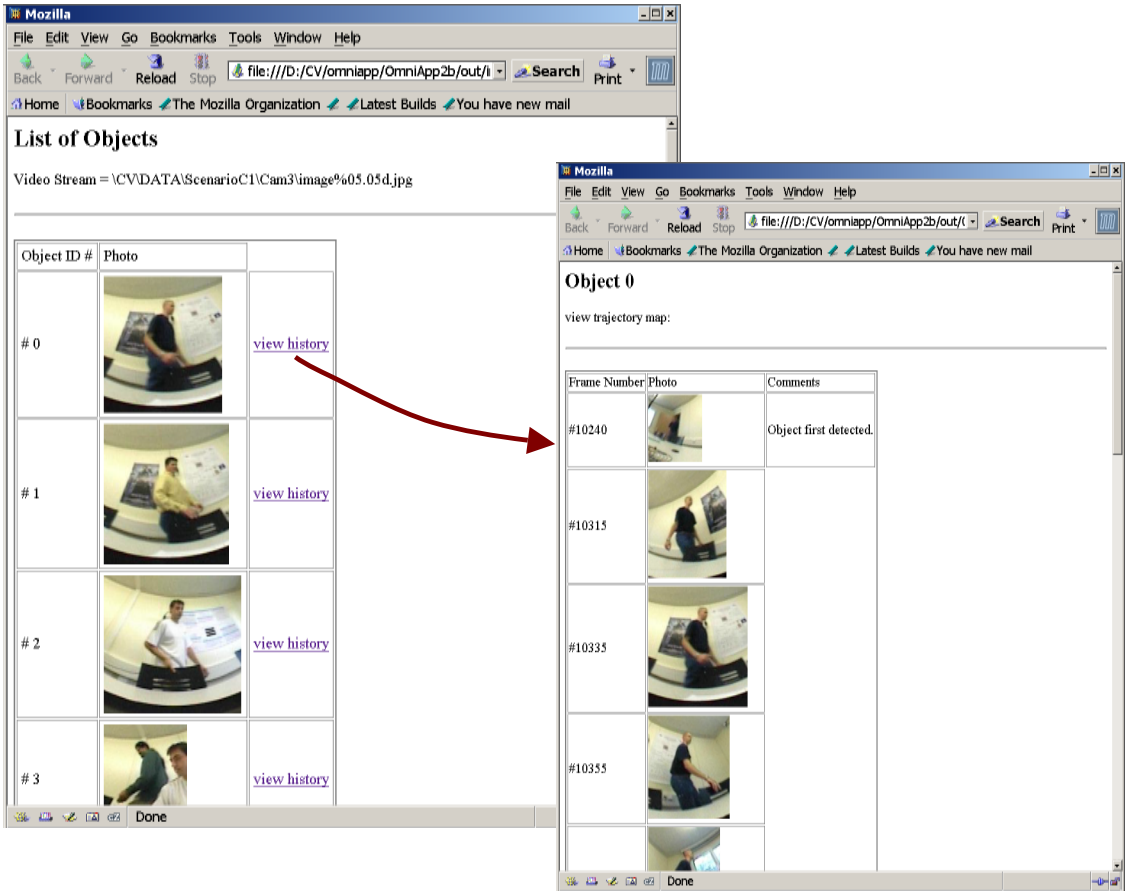


Figure 12.4: Tracking History results generated by the application and viewed in a web browser.

12.4 Conclusion

This chapter described how the OmniTracking application uses the results from the moving object tracking phase for 3D object localisation, assuming movement along the ground-plane. Objects can also be automatically tracked using virtual perspective cameras and this chapter described the two different modes of operation implemented by the program. The chapter then ended with a description of how the tracking history of objects is saved to disk. All three of them offer a great potential for further improvements and development, for example, domain information could be used by the history node to list events such as ‘person walked towards whiteboard’. In the case of automatic tracking of objects, the virtual camera controller could assign tracking priorities to objects based on some measure of significance, regions of interest define within the omnidirectional image or by using rules to dynamically switch a virtual camera from one object to another.