Real-Time Imaging **8**, 73–93 (2002)
doi:10.1006/rtim.2001.0236, available online at http://www.idealibrary.com on IDEAL®

A New Approach to Pose Detection using a Trinocular Stereovision System

Object location and tracking is a major issue in computer vision. This problem is normally solved through the extraction of representative features of the object, and the two-dimensional coordinates of these image features are used to compute the position of the object. When more than one camera is used, a certain similarity measure between the image features extracted from both stereoscopic images helps to match the correspondences. In this way, three-dimensional measurements can be recovered from the 2D coordinates of the features extracted from different cameras. In this paper the use of a trinocular system is considered to estimate both the position and velocity of known objects by using their apparent area, and with no use of the image-plane coordinates of the object's features. A high precision low-level image processor has been developed for performing object labeling and noise filtering of the images at video rate. Then, a position measurement tool uses the apparent area captured by every camera to locate the object. This enables us to estimate the position of the object. Finally, a prediction tool refines the estimation in locating the object. We show the performance of the trinocular system with a real implementation. This system has been designed to process the images provided by any conventional of high-speed cameras at video rate.

Rafael Garcia, Joan Batlle and Joaquim Salvi

Computer Vision and Robotics Group, Institute of Informatics and Applications, Dep. of Electronics, Informatics and Automation, University of Girona, Av. Lluis Santalo, s/n, E-17003 Girona (Spain)
E-mail: {rafa.jbatlle,qsalvi}@eia.udg.es

Introduction

A lot of effort has been carried out to reconstruct the spatial geometry of a scene using binocular stereovision systems. Most of the algorithms used by binocular systems use a certain similarity measure between both stereoscopic images in order to match the correspondences. Unfortunately, matching homologous points between images is not always possible, and false matches may appear. Thus, given a point in one image, prediction algorithms must be used to find the position of the homologous point in the second image. A computationally effective solution to overcome these difficulties relies on the use of trinocular stereovision systems, reducing the amount of false matches, by using the epipolar geometry to predict correspondences and, therefore speeding up the processing [1,2]. However, calibration represents a great deal of effort [3], since the focal length of the camera, f, and the two-dimensional coordinates of the principal point, x_c and y_c, are unknown unless they have been determined by some calibration procedure. Moreover, the coordinates of a point in the image plane are measured in terms of pixels. Converting these pixels into units of length (e.g. meters) also requires calibration. Our aim is to perform measurements minimizing the problem of calibration,

1077-2014/02/$35.00

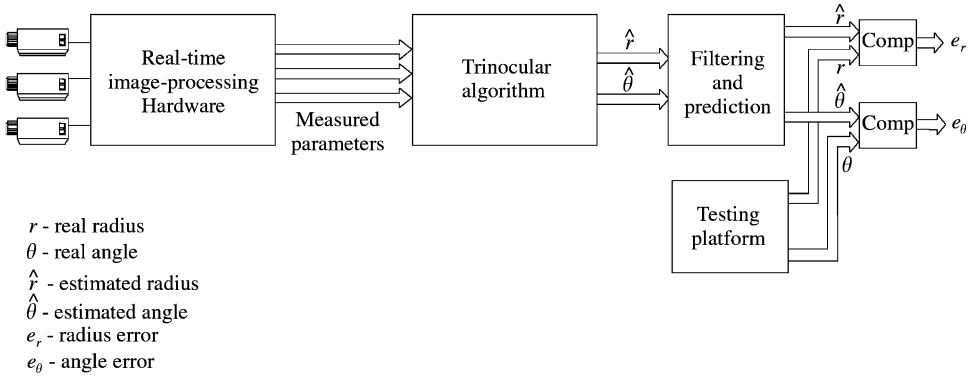

Figure 1. Block diagram of the system.

taking advantage of some additional knowledge of both the scene and the object to be tracked. On the other hand, it is well known that the precision of the three-dimensional scene reconstruction depends much on the distance between the cameras: the larger this distance is, the better the precision will be [4]. Increasing this distance also augments the differences between images, complicating the matching task.

In the literature, one can distinguish between two sorts of stereo algorithms. Firstly, those that use calibrated stereo algorithms, which perform measurements by means of the calibration matrix, and secondly, those using uncalibrated stereo systems, which involve the computation of invariants [5]. It should be noticed that our work is far away from such approaches, since our system takes advantage of the size, and, in some way, of the shape of the objects. Focusing on this idea, very few authors have used the apparent area of the objects for measuring distances. Weldon *et al.* [6] determined depth by using the apparent size of a planar patch, while Huber *et al.* [7] could settle the basis of a collision avoidance system in measuring the apparent separation between two corners of a rigid obstacle. In both these cases, the camera motion was known, thus the problem could be solved without knowing the intrinsic parameters of the camera. Pedersini *et al.* [8,9] have more recently proposed a calibrated trinocular system which takes profit of the apparent area for removing possible matching ambiguities. On the contrary, we propose a trinocular system which does not use the image coordinates where the extracted features of the objects are located in order to estimate depth. However, the system is restricted to those situations where the apparent shape of the object to track keeps invariant with respect to its position. For instance, let us take a racing car in a circuit, always passing in the same position and orientation in relation to a static camera. Its apparent area will provide distance information when approaching the static cameras at every turn. In a more general way, we can assign a parameter to the object, say ρ, which will help us to locate it. This parameter could be the length or height of the object, which will vary smoothly as the object moves. The simplest case of study would be a sphere, where the parameter can be defined by its diameter. In the case of the racing car, parameter ρ can be defined by the apparent width or height of the car.

Starting from the measure of these parameters, we will be able to find the position of the object by means of a specific algorithm as shown in Figure 1. First, an image processing hardware has been developed for processing the information provided by every camera. This hardware provides the parameters that are used by the trinocular system for determining object position. Then, the trinocular algorithm gives a first estimation of the position of the objects, and passes this data to the filtering module. Next, this module estimates recursively the state of the system, giving us the objects' position and velocity. Finally, this estimation is compared to the data supplied by the testing platform for obtaining a measure of the accuracy of the algorithm.

All these algorithms run in real-time granted that: 1) The image-processing tool computes the image algorithms at video rate, and 2) The trinocular algorithm, as well as the filtering and prediction can be computed in the host computer accomplishing the timing requirements of the system.

Summarizing, in the next section we introduce the basic geometry of our trinocular system. Following this, we discuss two different approaches to compute the

position of a known object depending on the apparent area viewed from every camera. Then, the next section shows the implementation of these algorithms in a computer vision frame. Finally, some results are presented, describing the advantages of our system.

Architecture for Real-Time Pose Detection

One of the aims of our system is to measure the position of the object at video rate. Thus, the development of a hardware, which could cope with the information provided by each camera, is absolutely essential.

From a system viewpoint, we consider the existence of three different ways of implementing specific image processing algorithms [10]. The first one consists of programming a general-purpose computer to perform the processing sequentially (software approach). In this case, the algorithm is mapped onto a fixed hardware machine. Another option is to design a specific circuit implementing the image-processing algorithm (hardware approach). Here, the machine structure is tailored to the application. The third alternative is to build a machine to obtain the versatility of the first approach, and the performance of the second one. We have applied the methodology of this last approach merging specific components for image processing (A/D–D/A video converters, sync. extractors, analog filters, etc.) with multiple purpose programmable devices (FPGAs). In this way, we obtain a real-time image-processing tool, which can be reconfigured to implement different algorithms. However, from an algorithmic viewpoint, the tracking problem is usually divided into three subproblems: segmentation, correspondence and motion estimation, usually in this order. Typically, correspondence is established between primitives, i.e. interest points [11], line segments [12] or planar patches [13], and it becomes a difficult problem to solve when several moving objects are present in the scenario. Our approach tackles the segmentation and the motion estimation problems. Lighting conditions have to guarantee that the cameras perceive the same object with similar chromatic characteristics. With the aim of providing the system with a higher degree of robustness, we have implemented an algorithm for correcting the effects of saturation in the CCD, whenever the lighting conditions generate this problem. In this way, color attributes are used to overcome the correspondence problem, assuming the chromaticity of the objects does not vary under the different projections. No further attempt has been made in this work to solve correspondences, although it would be a very interesting subject. This implementation upgrades our previous attempt of an image processing system presented in [14]. Substantial modifications have been introduced, which help to provide the system with a higher flexibility for performing more complex algorithms.

In order to compute the apparent area of the object, the first step consists of its extraction from the scene. In order to segment the object, we use the discriminatory properties of two color attributes: hue and saturation. Next, low-pass filtering is performed in order to reduce the inherent noise in the system. The third step consists of the computation of the apparent area of the object in the image plane by using the enhanced image from the segmentation module. The processing hardware presented in this section provides the sequence of apparent areas at video rate. As a last step, the measured areas are filtered in a sequence of images by means of an extended Kalman filter. Figure 2 shows a block diagram of the implemented algorithm. The low-level image processing operations are performed in hardware, while filtering and prediction is implemented in software.

Since the aim of the system is to estimate motion, we use regions because they provide more robust estimations than contours [15]. Like this, the proposed architecture uses a region-based tracking algorithm. Figure 3 illustrates the relationship between the algorithm presented in Figure 2 and its hardware

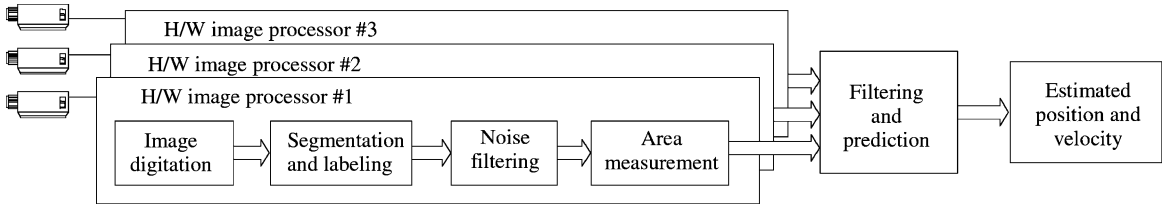

Figure 2. The image-processing algorithm for real-time tracking.

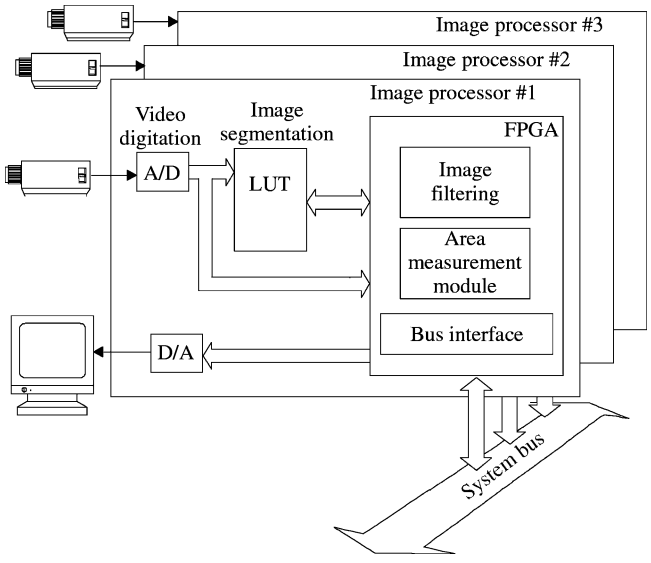

Figure 3. Block diagram of the 2D tracking system.

implementation. It can be seen how three hardware processors work in parallel in the images provided by their respective cameras.

In the following sections a detailed description of this hardware is given. As shown in Figure 3, it is divided in three modules: the segmentation module, the filtering module and the tracking module.

Segmentation module

This module performs a pixel labeling of the image, assigning a different label to pixels belonging to different objects. As we want to classify the objects by using their chromatic features, a real-time color conversion is performed. The system allows the programming of any mathematical conversion from RGB to a perceptual color model [16, 17]. In this case, we have chosen the model shown in Eqns (1) and (2) because of their scale-invariance properties. However, depending on the application, the most adequate model can be pre-programmed.

$$H(r,g,b) = tan^{-1}\frac{\sqrt{3}(g-b)}{(r-g)+(r-b)} \tag{1}$$

$$S(r,g,b) = 1 - \frac{3\,min(r,g,b)}{r+g+b} \tag{2}$$

From the previous equations it can be seen that the conversion from *RGB* to *HS* has the property of scale-invariance, that is, $H(r,g,b) = H(\alpha r, \alpha g, \alpha b)$ and $S(r,g,b) = S(\alpha r, \alpha g, \alpha b)$. Thereon, hue and saturation are stable under variations on the intensity of the illuminant [18,19]. Nevertheless, this linear behavior does not hold due to the non-linear response of the camera CCD, especially when one of the RGB components reaches its maximum value (know as "color clipping" in the literature). Moreover, when different sources of light illuminate the scenario non-uniformly, this linear relationship may not hold true either. We attenuate the clipping problem by applying a specific algorithm to minimize the problems of the CCD saturation, which increases virtually the dynamic range of the camera. This algorithm is fully described in [20]. Its main idea is to characterize the color of every object, finding the equations for R, G and B; and then re-scaling these three equations when one of the components reaches its maximum value. In this way, the segmentation module is much more robust against changes in the lighting conditions, giving the same value of H or S even when one of the channels of the camera is saturated.

Therefore, the next step consists of determining the chromatic characteristics of the objects to be tracked. These characteristics are defined in the HS color space. Since we are tracking known targets, minimum and maximum values of the hue must be defined for every object. The same operation has to be carried out for the saturation component. So, every object to be tracked is defined by a region on the HS space, depending on its color attributes. Figure 4 shows two different areas defining the color regions for two green patches with different saturation. In this figure, the angle of orientation represents the hue and the radius corresponds to the saturation (the further from the center, the more saturated is the color).

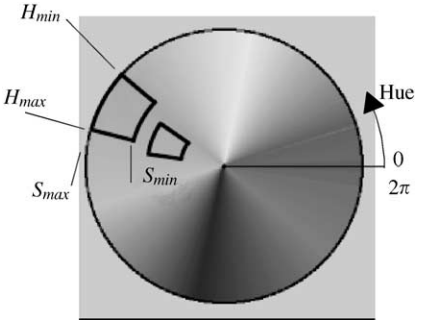

Figure 4. Color spaces belonging to two different green objects with different levels of saturation.

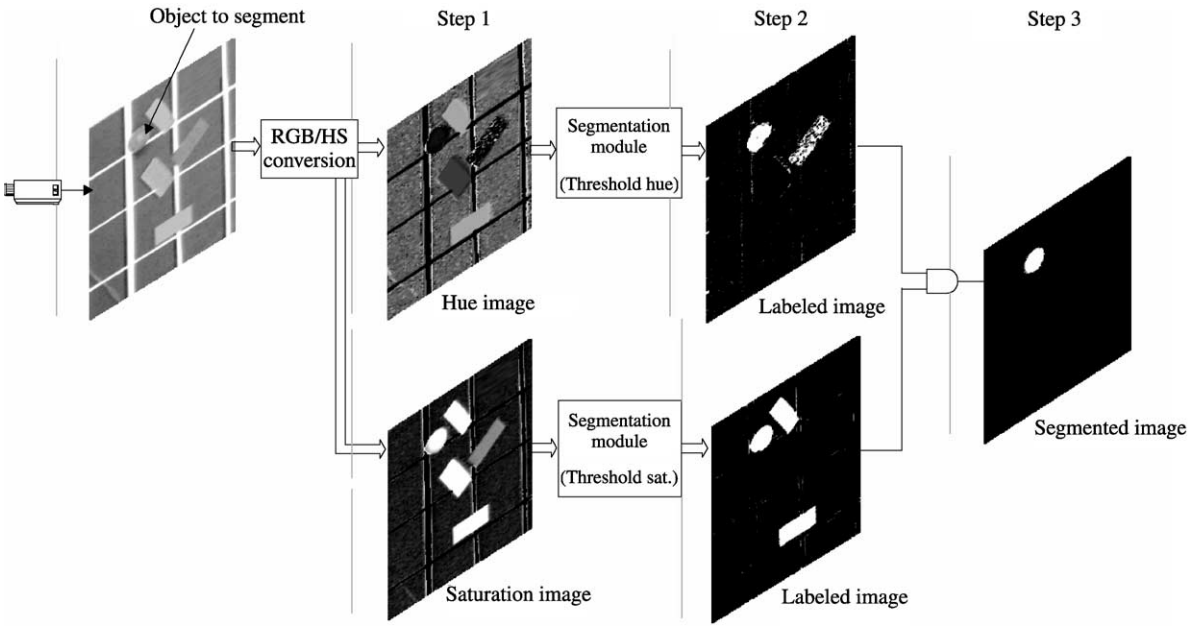

Figure 5. Scheme of the segmentation algorithm.

Figure 5 shows an example of the procedure we will follow. As it can be seen, the process starts with a conversion of the original RGB image to the HS space (step 1). Next, the hue image is thresholded by using the minimum and maximum parameters of the object to be segmented. As Figure 5 shows, more than one object may have a hue similar to the object to segment (step 2). Then, the saturation image is also thresholded. Finally, a Boolean AND operation is applied to both images, resulting in the segmented object (step 3).

By using these minimum and maximum thresholds, a "segmentation table" is computed. This table consists of 2^{18} positions of four bits, that is, a four-bit word for every RGB combination. We will call this four-bit word the "segmentation word" or "pixel label", and its value codifies every object in the scene. Every one of the 2^{18} positions is associated to an RGB combination, that is, six bits for every color component. Then, for every RGB value, the system validates to which object it belongs. If it does not belong to any object, the segmentation word stores "0000". Otherwise, it stores the label of its corresponding object, codified between one and 15. In this way, the objects are labeled depending on their color attributes. Figure 6 shows the segmentation table. In practice, the board incorporates a $128\,k \times 8$ bit memory, with 17 address bits. The least significant bit of the blue component is used to distinguish between the higher or lower address.

The computation of the segmentation table is performed off-line, in the set-up process. Once it has been calculated and transferred to the memory of the board, it will perform segmentation and labeling at video-rate. Therefore, the A/D converters for signals R, G and B generate a 18-bit address for the segmentation table, providing a new pixel-label to the FPGA every 100 ns (the converter clock runs at 10 MHz). The resulting segmented image is shown in Figure 7.

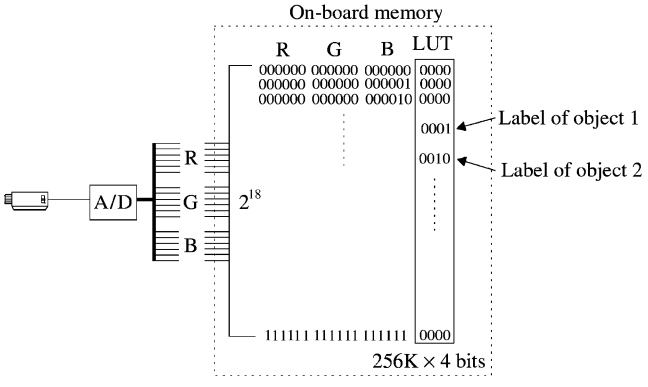

Figure 6. The labeling/segmentation table (LUT).

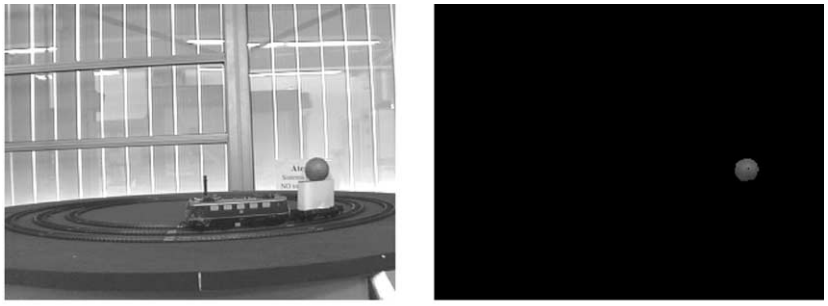

Figure 7. (a) Original image; (b) segmented image.

Filtering module

It is well known that all the vision systems are affected by an inherent noise that has different sources. Low-pass filtering usually eliminates this noise, but sometimes it has a double effect: blurring edges and other sharp details. An additional problem is that image filtering in itself requires considerable computation. Since the objective is to achieve noise reduction rather than blurring, a valid approach is the use of a mode filter. This filter counts the number of times that a certain datum appears in a 3×3 neighborhood of the processed pixel. The mode filter is well suited for our architecture, allowing noise reduction without loosing the contours. The filtering module takes as input an image containing the four-bit labels from the segmentation module, and it carries out two iterations of the mode filter over the labeled image. Figure 8 shows how a second filter is applied in order to decrease the rate of noise of the image. This two-iteration mode filter is computed at video-rate in the FPGA using 5% of its capacity.

Tracking module

At this stage, the tracking module computes the apparent area of the objects to be tracked from the

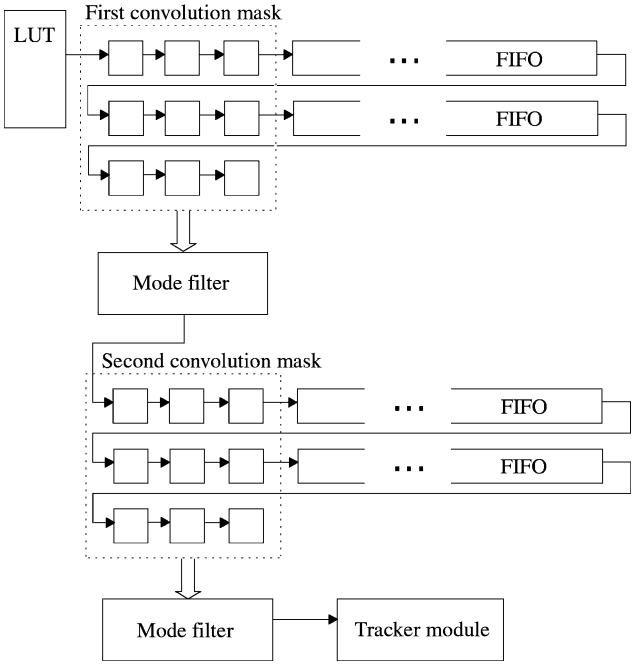

Figure 8. Two-step mode filter.

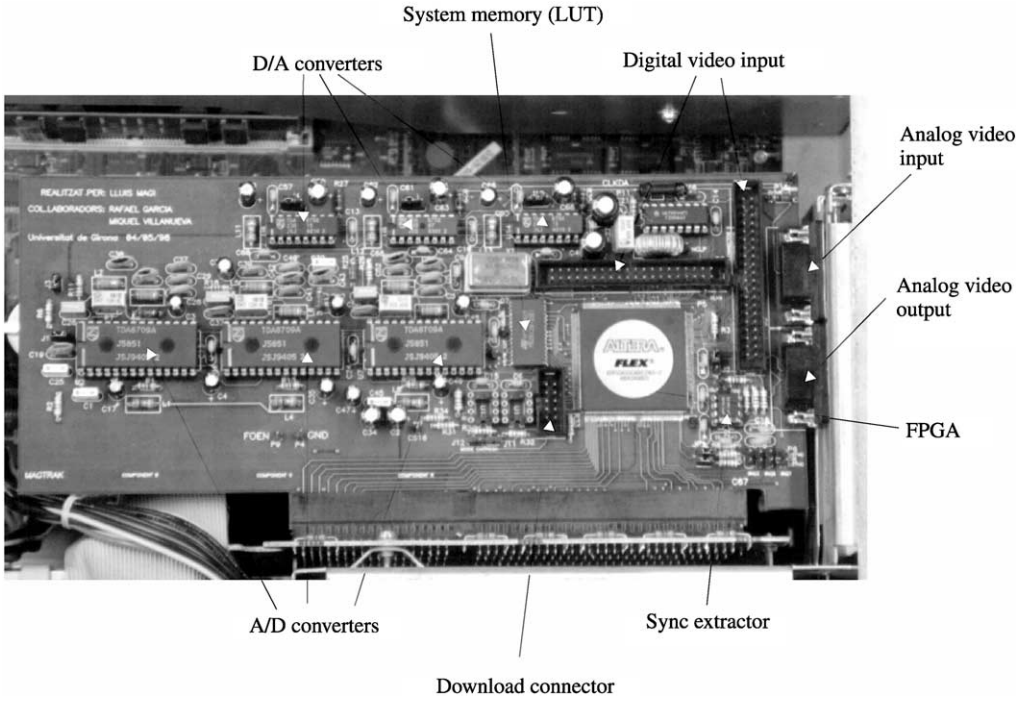

Figure 9. Photograph of the tracking device.

image provided by the filtering module. The first phase consists of a blob classification, detecting how many objects of interest are present in the scene. After the processes of color labeling and image filtering, some noise could still appear in the image. For this reason, some further steps should be taken to achieve the most accurate results. First, those blobs which have a size smaller than a certain threshold are discarded. This technique filters off most of the noise due to the existence of objects with a color similar to the objects of interest. Normally, when more than one object is present in the scene, different objects are classified with different labels. However, if two identically colored objects were present in the scene, they would be assigned the same label. As detailed before, the labeling process classifies the objects depending only on their chromatic features. For this reason, a connected blobs algorithm checks whether two objects classified with the same label are present in the image. If this happens, it assigns every object a different label. Next, the apparent area of the objects is computed accumulating the number of pixels for every region classified with a certain label L_i. Finally, a set of apparent areas is provided to the host computer. A photograph of the hardware image processor is illustrated in Figure 9.

The modules described above have been implemented using Hardware Description Languages (VHDL), and synthesized on an Altera Flex-10 K100 Field Programmable Gate Array (FPGA). The image processor incorporates a connector for reprogramming the FPGA in-system, facilitating the fast prototyping and development of algorithms in hardware. Thus, future improvements of the system can easily be carried out. Moreover, digital cameras can be plugged into the processor through the digital video connectors, allowing to process images at a higher frame rate.

Temporal analysis

Most of the image processing systems use a two-step approach, where one image is being processed while the next image is being grabbed (stored in most sort of memory). However, our implementation reduces the latency of the system by starting the processing with the first pixel of the image. The use of a dedicated hardware allows the overlapping of different tasks through a pipeline technique, as illustrated in Figure 10.

Since we use a multitask operating system without real-time capabilities (Windows NT), it is difficult to predict the time needed by the host to carry out the

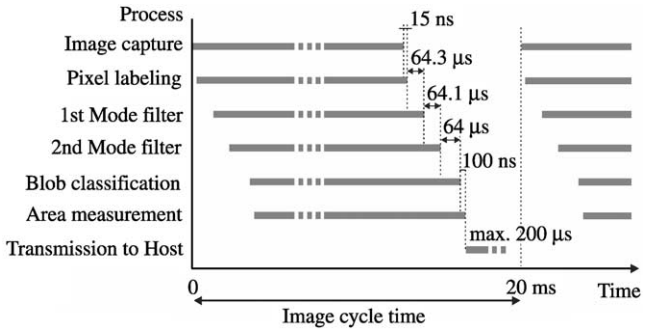

Figure 10. Timing diagram of the algorithm. The different data-intensive processes run in parallel.

reading of the data. The performed tests showed how this time ranged from 60 to 200 µs, with a typical value around 150 µs. This reading will depend not only on the speed of the CPU, but also on the scheduling of the threads, on their priority and on the load of the system. Further details on this matter will be exposed in the later section on temporal issues.

Geometry of the System

The trinocular stereovision system

We present a vision system composed of three cameras the distribution of which is known to us, and which cover the whole scene. As an example, Figure 11 shows three cameras shifted $2\pi/3$ radians among them. In order to locate the objects in the scenario, we define a world coordinate system centered at O_w, chosen as the approximate intersection of the optical axes of the three cameras. A simple computer program can help in the location of the cameras, settling O_W in the center of the image plane.

The plane defined by the optical centers (C_i) of the three cameras is parallel to the ground plane (X_w, Y_w).

So as to facilitate the comprehension of the system equations, the positions of the objects are expressed in polar coordinates (r, θ), taking $\theta = 0$ as the orientation towards the optical center of camera 1 (see Figure 11). Initially, we will assume the objects are located in the plane $Z_w = 0$.

A mathematical approach to parameterize the area of the objects

Our aim is to adjust the trajectory followed by the object through a mathematical function, which depends on the radius and the distance between the object and every camera. To start with, we analyze the function that governs how the apparent area (A), viewed from one of the cameras, changes as the object moves. Let us imagine that a radius r is fixed, and the object moves

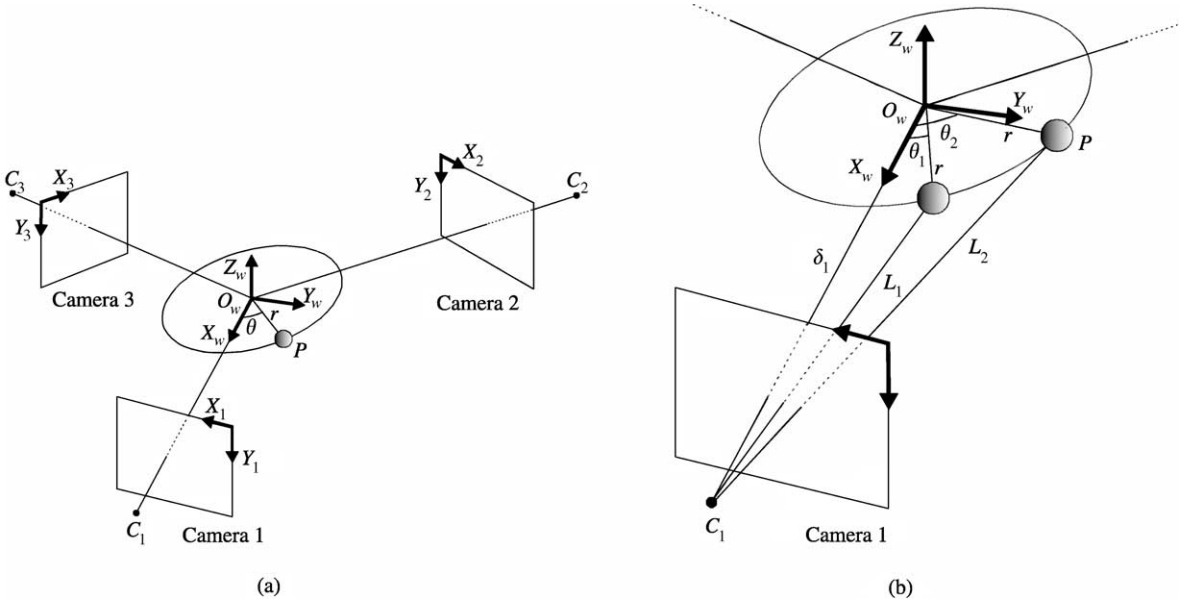

Figure 11. Geometry of the trinocular stereovision system.

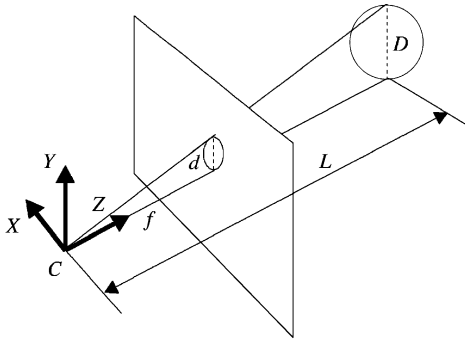

Figure 12. Perspective projection of an object on the image plane.

describing a circle centered at O_w. As shown in Figure 11(b), the distance L varies depending on the angle θ.

Eqn (3) is derived from the triangles shown in Figure 11(b), using the cosine theorem.

$$L_i^2 = \delta^2 + r^2 - 2\delta r \cos(\theta_i) \tag{3}$$

As the apparent size of an object is inversely proportional to the distance, it is possible to establish a relationship between the real size of the object D, and its apparent size d, depending on the focal length of the camera (see Figure 12).

Using the pinhole camera model, the origin of the three-dimensional system (X, Y, Z) is the center of projection C. The distance between the optical C and the image plane π is the focal length f. So, as the distance L from the object to the camera decreases, d augments. Therefore, applying the geometric law of the perspective projection [21], we obtain:

$$d_i = \frac{D \cdot f}{L_i}. \tag{4}$$

Fixing r, the apparent size d_i for every θ_i, can be found:

$$d_i = \frac{D \cdot f}{\sqrt{\delta^2 + r^2 - 2\delta r \cos(\theta i)}} \tag{5}$$

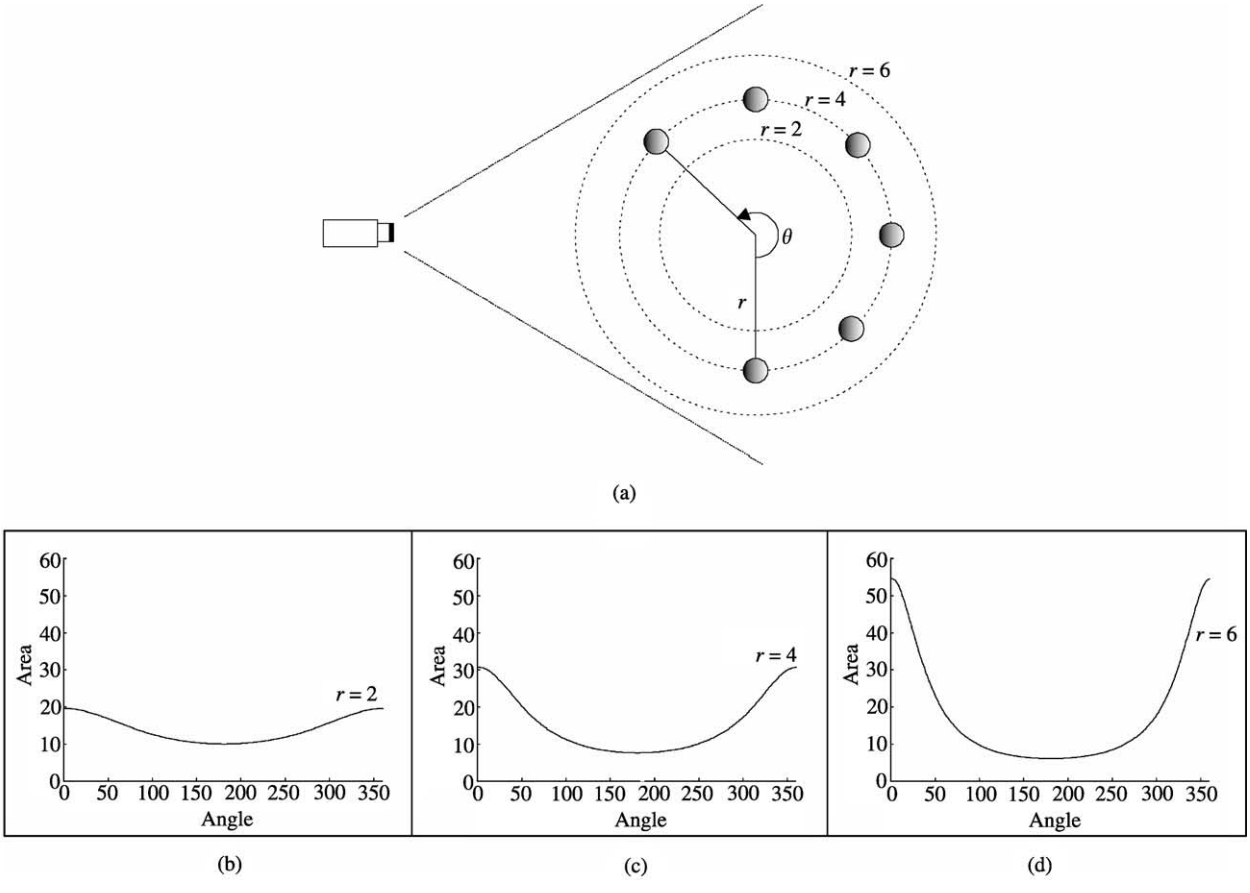

Figure 13. Variation of the apparent area of a spherical object when it performs a 360° rotation with a fixed radius r. (a) Scheme of the simulation; (b), (c) and (d) results with $r = 2$, $r = 4$ and $r = 6$, respectively. The angle is expressed in degrees.

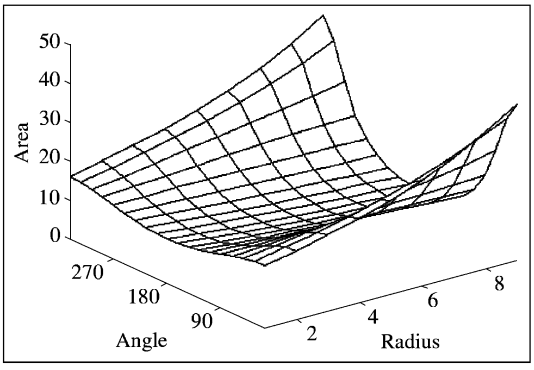

Figure 14. Variation of the apparent area of a spherical object with respect to its location. The position of the object is represented in polar coordinates (radius r and angle θ).

As our d_i measures are a direct function of Df, we do not mind about the shape, but only the apparent area. In Figure 13 we have normalized Eqn (5) by taking $Df = 1$. For every camera, we are computing the variation in the apparent area (A) for every angle (θ_i), obtaining the graphics shown in Figure 13.

Now, if this set of functions is pictured in a 3D representation, the function in Figure 14 is obtained. As it was expected, the apparent area of the object augments when the distance to the camera decreases.

As we know what the area function is like, depending on r and θ, and since we are dealing with circular trajectories, we can compute the area functions seen from the cameras 2 and 3, by shifting the function in Figure 15 in $2\pi/3$ and $4\pi/3$ radians, respectively. This is illustrated in Figure 15. Obviously, cameras can be placed at any different positions.

Placing the object at any position, we can measure then the apparent area viewed from every camera of the trinocular system, obtaining three different values (A_1, A_2, A_3). If we intersect these three values with its corresponding functions, we obtain the $[r, \theta]$ candidate values from every function, as shown in Figure 16.

Analyzing Figure 17, it can be seen that a set of (r, θ) values are possible candidates for every camera. As an example, let is analyze the function for camera 1. Figure 18 shows how there exists a direct relationship between the intersection points and the possible locations of the object.

Since the (r, θ) values we are looking for must be the same obtained from every camera, we have to be able to

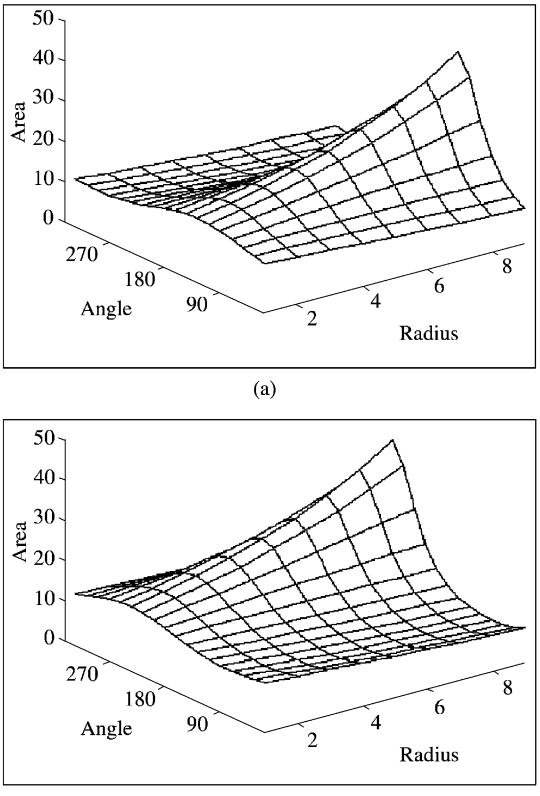

Figure 15. Apparent area function viewed from cameras 2 and 3, located at $2\pi/3$ and $4\pi/3$ radians.

infer this univaluated (r, θ) point starting from the intersection of these three curves. Figure 19 shows this intersection, illustrating how a single crossing point

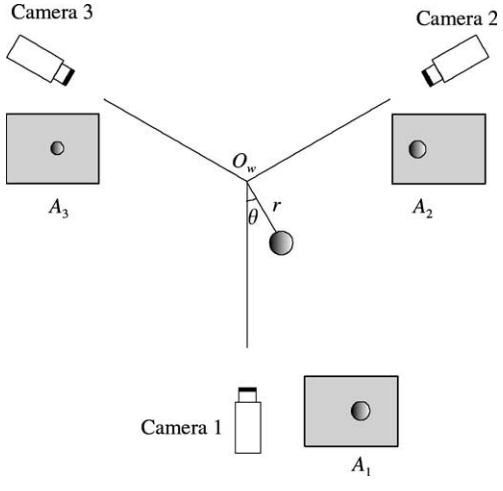

Figure 16. Apparent areas (A_1, A_2, A_3) measured from cameras 1, 2 and 3, respectively.

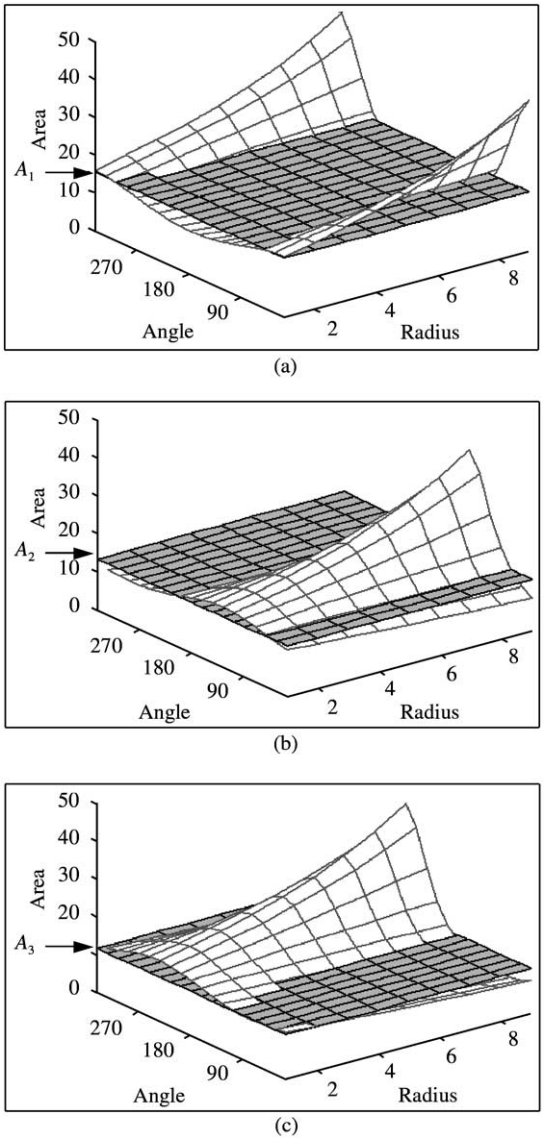

Figure 17. Intersection of the three area functions with planes A_1, A_2, A_3. Every plane corresponds to the measure of the apparent area of the object captured from its corresponding camera.

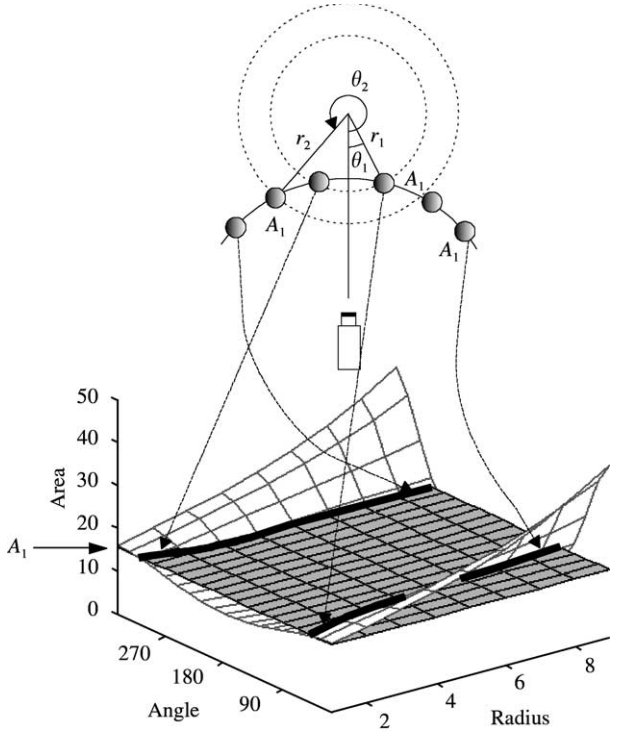

Figure 18. The thick solid line shows the intersection of the area function for camera 1 and the measured area (plane A_1). Dotted lines show the correspondence with the possible locations of the object. Depth cannot be computed from one image by using only the apparent area.

exists among these functions. It is possible to obtain a set of candidate points by finding which values of (r, θ) generate a value of the apparent area close enough to the area measured by every camera. These values are the intersection of the functions and planes showed in Figure 18. In this way, we obtain the three functions shown in Figure 19. These three functions intersect in a point, giving the adequate (r, θ) value. However, this method needs a high amount of computation. Moreover, the intersection of the three functions generally gives a few candidate points—the proper solution can be found by the application of a least squares criteria.

Concluding, we can observe that (r, θ) can be computed from this intersection. However, this involves a high computational cost. For this reason, a less expensive method will be discussed in the next section.

Geometrical computation

Our second approach consists in setting out the problem in the following way: now, the apparent diameter of the object will be considered as a measure, so that equations will be plainly understood. According to the previously described constraints concerning the objects to be tracked, we can consider that there is a direct relationship between the real diameter of the object (D) or any other parameter, and the apparent diameter viewed from a camera (d), as it was illustrated in Figure 12. If we assume that such a relationship exists, the following equations can be obtained, depending on the focal

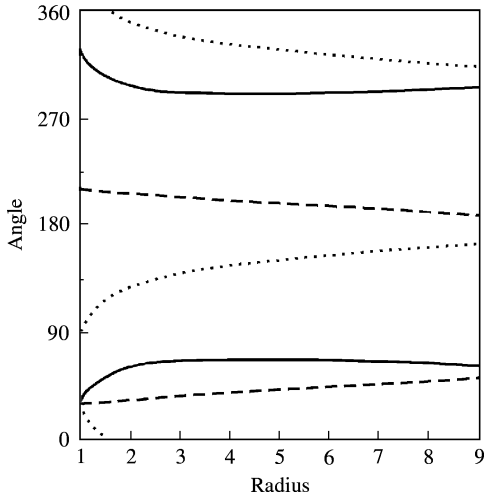

Figure 19. 2D plot of the intersection of every apparent area with the theoretical area function for every camera. ——A_1; – — — —A_2; . . .. . . A_3.

length of every camera (f):

$$L_1 = \frac{Df_1}{d_1} \tag{6}$$

$$L_2 = \frac{Df_2}{d_2} \tag{7}$$

$$L_3 = \frac{Df_3}{d_3} \tag{8}$$

As we are using three identical cameras with the same optics, it can be assumed that $f_1 = f_2 = f_3$. Then Eqns. (6–8) can be equaled using the Df_i factor, obtaining Eqn (9).

$$L_1 d_1 = L_2 d_2 = L_3 d_3 \tag{9}$$

Next, we want to find r and θ, depending on the apparent diameters d_1, d_2 and d_3 through a set of equations. Figure 20 shows a scheme of the trinocular system.

Using this figure, we can establish a set of equations, applying the theorem of the cosine to different triangles. Let us first take OP1 for finding Eqn (10), OP2 for Eqn (11), and OP3 for Eqn (12),

$$L_1^2 = \delta^2 + r^2 - 2\delta r\cos(\theta) \tag{10}$$

$$L_2^2 = \delta^2 + r^2 - 2\delta r\cos\left(\frac{2}{3}\pi - \theta\right) \tag{11}$$

$$L_3^2 = \delta^2 + r^2 - 2\delta r\cos\left(\frac{2}{3}\pi + \theta\right) \tag{12}$$

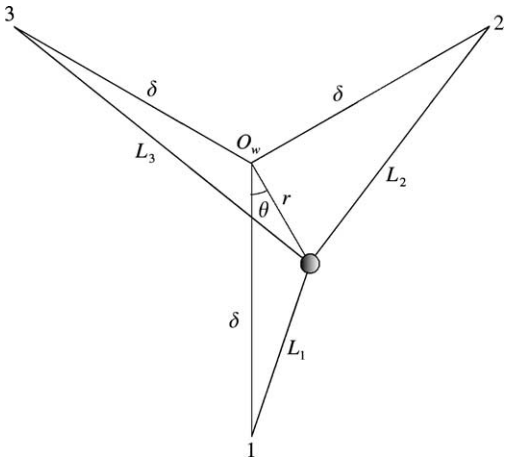

Figure 20. Scheme of the trinocular system.

where L_1, L_2, L_3, θ and s are unknowns. As was shown in Eqns. (6–8), L_i can be expressed as a function of D and d_i. Then isolating and equaling D, we can obtain:

$$L_2 = \frac{L_1 d_1}{d_2}, \text{and } L_3 = \frac{L_1 d_1}{d_3} \tag{13}$$

If we substitute a_2 and a_3 in Eqns. (10–12), we obtain the following system:

$$r^2 - 2\delta r\cos(\theta) + \delta^2 - D_1^2 = 0 \tag{14}$$

$$r^2 - 2\delta r\cos\left(\frac{2}{3}\pi - \theta\right) + \delta^2 - \left(\frac{L_1 d_1}{d_2}\right)^2 = 0 \tag{15}$$

$$r^2 - 2\delta r\cos\left(\frac{2}{3}\pi - \theta\right) + \delta^2 - \left(\frac{L_1 d_1}{d_3}\right)^2 = 0 \tag{16}$$

In order to simplify the nomenclature we introduce the following variable changes:

$$t_2 = \left(\frac{d_1}{d_2}\right)^2, t_3 = \left(\frac{d_1}{d_3}\right)^2, \text{and } L = L_1^2 \tag{17}$$

The resulting system, after expanding the cosine, is then expressed as shown in Eqns (18–20):

$$r^2 - 2\delta r\cos(\theta) + \delta^2 - L = 0 \tag{18}$$

$$r^2 + \delta r\cos(\theta) - \delta r\sqrt{3}\sin(\theta) + \delta^2 - Lt_2 = 0 \tag{19}$$

$$r^2 + \delta r\cos(\theta) - \delta r\sqrt{3}\sin(\theta) + \delta^2 - Lt_3 = 0 \tag{20}$$

In this way, we have a non-linear system with three equations and three unknowns. This equation system has been solved with the aid of a specific software

package for symbolic calculation. Once the system has been solved, it can be proved that the angle θ satisfies:

$$\theta = \text{arctg}\left(\frac{(t_3 - t_2)\sqrt{3}}{t_2 - t_3 - 2}\right). \quad (21)$$

On the other hand, the radius can be expressed as:

$$|r_1| = \left|\frac{1}{2}\frac{(\sqrt{C_1} + \sqrt{C_1}t_2 + \sqrt{C_1}t_3 - \sqrt{C_2})\delta}{C_1}\right| \quad (22)$$

$$|r_2| = \left|\frac{1}{2}\frac{(\sqrt{C_1} + \sqrt{C_1}t_2 + \sqrt{C_1}t_3 - \sqrt{C_2})\delta}{C_1}\right|, \quad (23)$$

where $C_1 = t_2^2 - t_2t_3 + t_3^2 - t_3 - t_2 + 1$, and $C_2 = -3C_1(t_2^2 - 2t_2 + 1 - 2t_2t_3 - 2t_3 + t_3^2)$. For all the radius in the range $[0,\delta]$ the right value is given by r_1.

To sum up, r and θ values have been obtained from a trigonometrical procedure. The advantage of this approach, in comparison with the previous one solved in the last section last, is based on the fact that it is faster to compute. Moreover, these (r, θ) functions can be easily preprogrammed using Look Up Tables (LUTs).

Pose estimation and prediction

In the previous section we have seen how an estimation of the position of the object can be obtained from its apparent area. In order to achieve the most accurate results, some sort of filtering has to be introduced in the system. Then, a first estimation of the position of the object is obtained through the use of Eqns (21–23).

First, the apparent area of the object is obtained from the image processors described in the previous section. Next, a first order filter carries out a noise filtering of the measured apparent areas (A_1, A_2, A_3). Because this filter just takes into account the present value of every variable (A_k) and its previous estimation (A_{k-1}), it is very fast to compute. The equation of the filter is as follows [22]:

$$\widetilde{A}_k = \widetilde{A}_{k-1} + k \cdot \left(A_k - \widetilde{A}_{k-1}\right), \quad (24)$$

where $\widetilde{A}_k$ is the estimated value of the apparent area a, measured from one of the cameras at instant t_k, and k is a constant of the filter. An Extended Kalman Filter (EKF) uses these data in order to filter and predict the position of the object in the future time instants. Tests have shown that these equations provide a good prediction of future locations of the object, even when taking noisy measurements.

It should be noted that the testing scenario described previously is only a tool to validate the accuracy of the results. Obviously, the system allows the computation of trajectories which are less constrained than the motion of the ball placed on the freight car. This track has just been used for testing, so that we could compare the real position of the object with the one estimated by the trinocular system. Thus, our system tracks the objects in unconstrained situations, where the object can move freely, instead of going along a predetermined track.

Temporal issues

Previously, we have analyzed the temporal characteristics of the image-processing algorithm. It should be noted that the cycle time of the image-processing algorithm is 20 ms when using standard PAL cameras. Therefore, in order to keep the cycle time of the whole system within 20 ms, the computation of r and θ, and the later filtering have to satisfy this explicit response-time constraint of 20 ms. Failing that, any further action or computation derived from our measures (like, for instance, the instantaneous velocity of the object) would entail system failure [23]. On the other hand, the choose of a host computer and an operating system should ensure that the response time of the system meets the critical requirements of a real-time application. The tests were carried out on a computer with a Pentium III 450 Mhz processor. As said before, we used a multitask operating system without real-time capabilities (Windows NT). However, the computational requirements of the trinocular algorithm and the subsequent filtering more than allow the use of this platform, including the host computer and the operating system. In this way, the host has 20 ms to perform these computations until the next data from the image-processing tool is available, and we obtain a cost-effective solution without the need of a more expensive real-time operating system.

Experimental Work

The experimental set-up

In order to test the results of our system, a calibrated structure has been built. Figure 21 shows a scheme of the platform we have used.

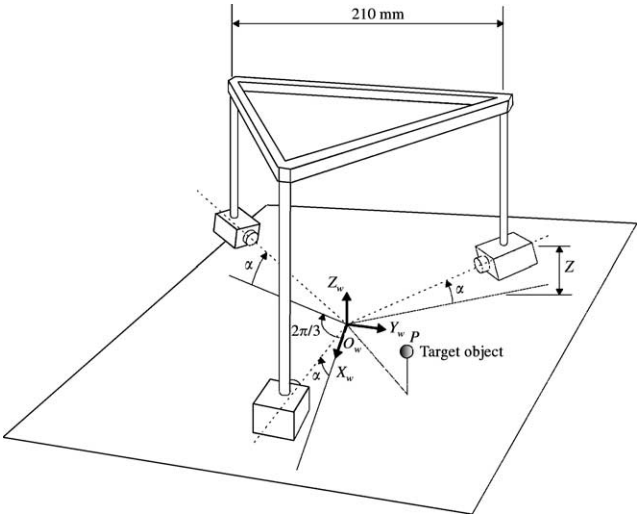

Figure 21. Testing platform. Angle α has been initially set to 0.

This platform was built in a tubular structure equipped with mechanical facilities to adjust pitch and roll degrees of freedom, as explained below. As derived from the equations in the last section, the geometry of the system relies on the fact that the three cameras are distributed around the scenario at any known positions. Eqns (17) and (18) have assumed that the angle between the cameras is $2\pi/3$ radians. This constraint is not a prerequisite of the system, since any known angle among the cameras could be used. In order to test the reliability of the results, it is also necessary to know what happens when there is some kind of inaccuracy in the angles formed by the position of the cameras. To start with, our structure provides approximately $2\pi/3$ radians among the three cameras. Then, the extrinsic parameters of the three cameras have been found by means of a calibration algorithm [24]. In this way, we know the real position and orientation of the cameras. The testing structure is granted analog sensors suited to each camera (angular potentiometers), which allow the modification of the pitch angle with 0.05 degrees of precision. In such a way, the orientation angle of the cameras can be properly adjusted, allowing the feedback from the sensorial system. For our initial tests the structure has been adjusted to exactly $2\pi/3$ radians. Then, these angles have been modified in order to test the system. It will be shown later in this paper, how small unknown variations in these angles appear to deteriorate the results in quite a small factor.

The scenario of our testing platform consists of three concentric tracks, on which an electric train circulates,

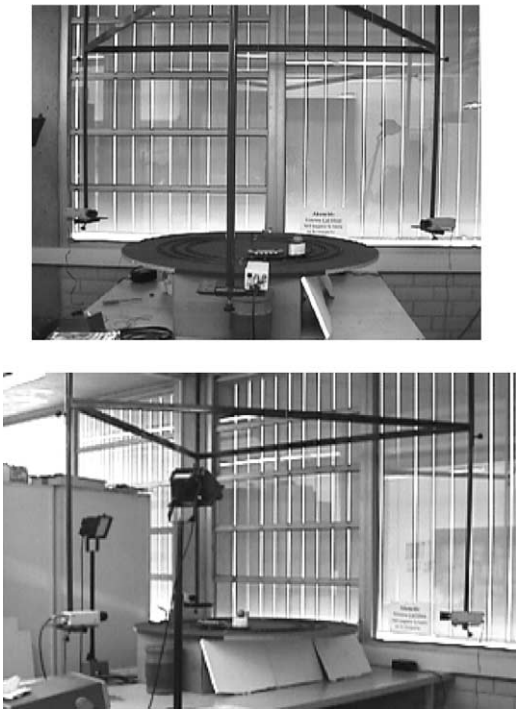

Figure 22. Photograph of our trinocular system.

as shown in Figure 22. With the aim of capturing the three images at the same time instant, cameras 2 and 3 work synchronized by the master one (camera 1). With this purpose, we have used three JVC TK1270 cameras with synchronism input.

The train drags a freight car on which a ball has been placed. In order to know the accuracy of r and θ provided by our trinocular algorithm, we need to know the real polar coordinates of the ball. Since the radius of these tracks is already known, we only need to evaluate the angle. With this purpose, a high precision incremental encoder has been placed at the center of the platform, as shown in Figure 23. Then, a rigid rod has been attached to its axis, revolving jointly around the encoder. The other side of the rod has been attached to the freight car, where the ball is located. Thus, as the train revolves, the platform provides a real measure of the angle value, which is compared with the value computed by our trinocular algorithm.

As we will see in the results section, the accuracy of the system demonstrates the validity and usefulness of the presented method.

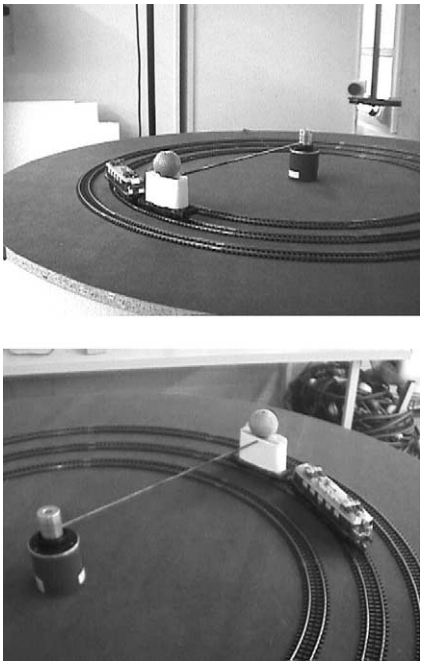

Figure 23. The testing scenario; a train drags a freight car on which a ball has been placed

Experimental results

The theoretical results presented in the last section for computing the position of an object have been verified in real–world experiments. The accuracy and stability of the object position has been studied with respect to various factors, such as pixel noise, due to the segmentation thresholds and calibration errors. In our tests we have considered a spherical ball with a diameter of 42 mm, placed over a wagon. We first experimented over a sequence of 415 real images, where the train performed a whole translation over the rail track. Real-time segmentation and tracking from the three cameras is performed in parallel by using three processing cards like the one detailed previously. Figure 24 illustrates the changes of the apparent area of the ball, measured from cameras 1, 2, and 3 as the object performs a complete turn. A detail over these area functions is also shown.

By using these three measured values of apparent area at every time instant (A_1,A_2,A_3), the position of the object can be determined by means of Eqns (21–23), as described in the section on geometrical computation. For every image over the sequence, a new measurement of the apparent areas from every camera allows the computation of the new position of the object (radius and angle). Representing these computed coordinates of the position of the object in polar coordinates, a reconstruction of the measured path followed by the train can be obtained, as shown in Figure 25.

In can be seen from Figure 25 how there exists some sort of inaccuracy in the measurements of the position of the object. With the aim of reducing this noise, the first order filter described in Eqn (24) has been used. Figure 26 shows the result of filtering the measurements of the three cameras.

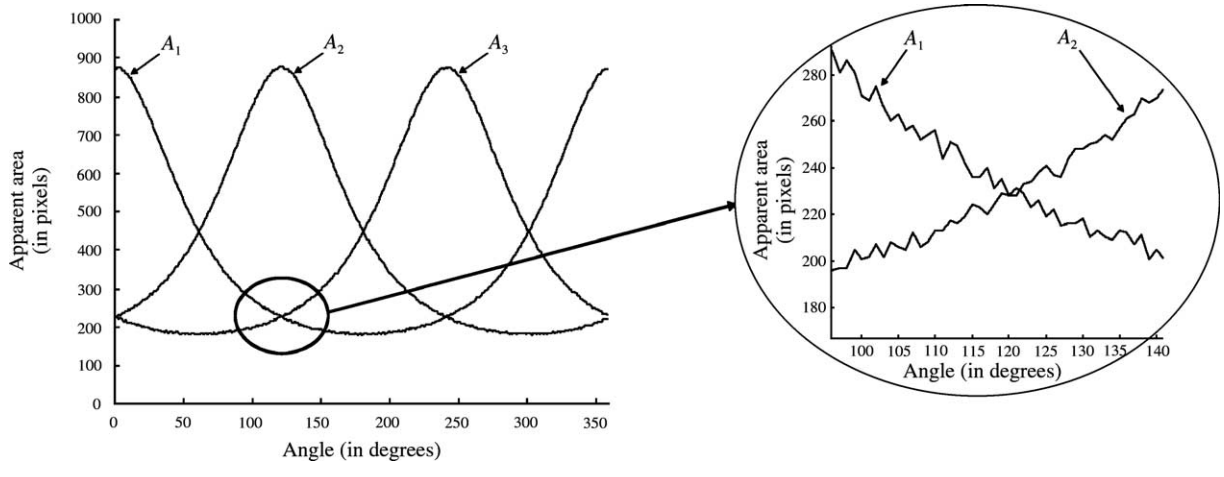

Figure 24. Raw apparent areas (A_1, A_2, A_3) measured respectively by camera 1, 2 and 3, and zoom on the marked zone of the raw data.

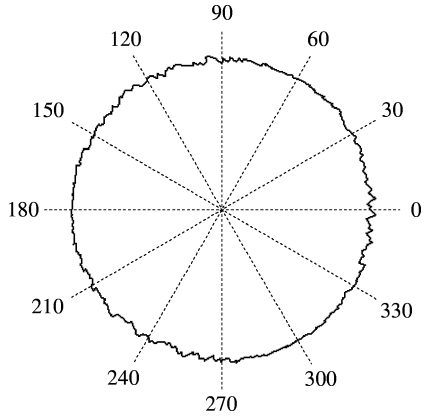

Figure 25. Measured train track.

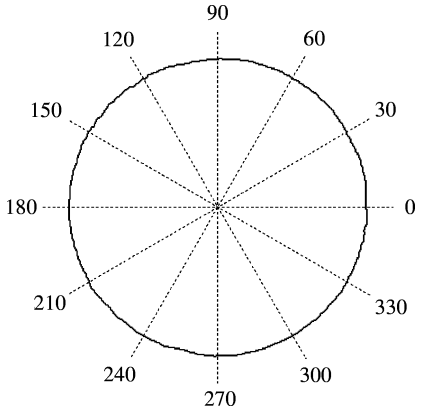

Figure 27. Filtered train track.

Now, we can use this filtered data for applying Eqns (26–28). Figure 27 shows the obtained results, plotted in polar coordinates.

Since we can guarantee the accuracy of our measurements thanks to the described experimental setup, the estimations of the trinocular system can be compared with the real measurements, as we know the real radius of the train track, and the encoder provides the real angle. Errors in the angle estimation can be computed from Table 1.

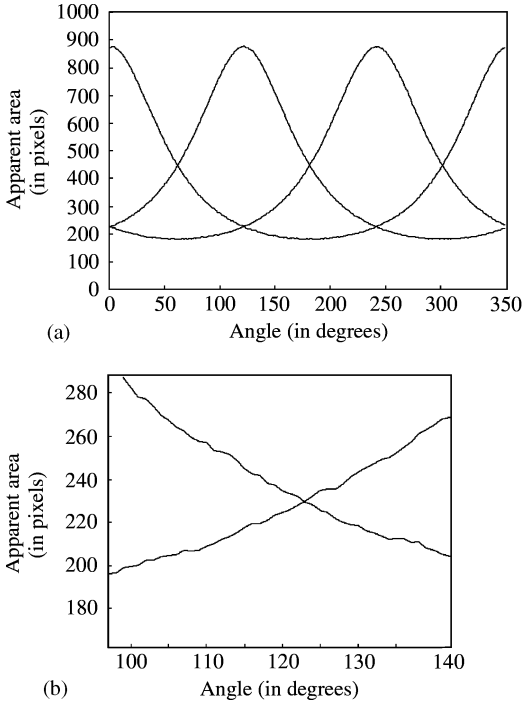

Figure 26. (a) Filtered areas with $k = 0.2$; (b) zoom over the filtered function.

From Table 1 it can be seen that filtering the data provides a significant improvement in the accuracy of the results. On the other hand, Table 2 shows the differences between the real and the estimated radius before and after filtering.

Next, a measure of the error should help to evaluate and compare our different approaches. Table 3 shows the quadratic error in the estimation of the radius and angle.

As said before, the cameras can be oriented to any position, provided that the relative angles among the cameras and the world coordinate center are known. A different problem is that derived from inaccuracies in the orientation angles of the cameras, due to calibration errors. Then, we should analyze what happens when there exists some sort of inaccuracy in the estimation of the angles formed by the cameras. It was explained previously that the testing structure is provided with angular potentiometers, which allow the modification of the pitch angle of the cameras with high precision. Hence, these sensors have been used for modifying the position of the cameras in the following way: camera 2 was moved one degree to the right, and camera 3 was moved 2 degrees also to the right. In this way the relative angles of the cameras were 0, 121 and 242 degrees, respectively. Next, the equations were applied, considering the theoretical $2\pi/3$ radians among them. The results achieved by the system are illustrated in Table 2, and a reconstruction of the path followed by the freight car is shown in Figure 28.

Table 1. Computed and filtered angle measurements

Real θ (in radians)	Before filtering		After filtering	
	Computed θ	$\Delta\theta$	Filtered θ	$\Delta\theta$
−3.124139	−3.115878	0.008261	−3.125878	0.001739
−3.106686	−3.109911	0.003225	−3.106718	0.000032
−3.089233	−3.085901	0.003332	−3.092808	0.003576
−3.071779	−3.079807	0.008027	−3.069802	0.001978
−3.054326	−3.059276	0.004950	−3.054780	0.000454
−3.036873	−3.041431	0.004558	−3.032440	0.004433
−3.019420	−3.019408	0.000012	−3.020340	0.000920
−3.001966	−2.994984	0.006983	−3.000084	0.001883
−2.984513	−2.995053	0.010540	−2.982484	0.002029
−2.967060	−2.991344	0.024284	−2.969146	0.002086
−2.949606	−2.972304	0.022697	−2.957670	0.008063
−2.932153	−2.920355	0.011798	−2.937554	0.005401

On the other hand, we wanted to test the accuracy of the system with respect to pixel noise due to inadequate segmentation thresholds. With this purpose we have computed the theoretical apparent area of the object, measured from cameras 1, 2 and 3, and we have introduced Gaussian noise with $E[x(n)]=0$ and var $[x(n)]=0.89$, as shown in Figure 29. This noise has the same distribution as pixel noise in imaging systems.

Again, a reconstruction of the path followed by the freight car is shown in Figure 30, expressed in polar coordinates.

As we can see, the accuracy on the object position decreases considerably with pixel noise, while it keeps quite stable with small calibration errors.

A second test has been carried out by using a more complex object than a sphere. Figure 30 shows a toy train placed over the freight car. In this case, the exact position of the cameras is unknown. The equations are found through a calibration turn, in which the apparent areas captured from every camera are stored. The knowledge of the position of the freight car at every time instant allows the representation of the functions shown in Figure 31(b).

This set-up can be applied to any situation where the object describes a repetitive motion, like a racing car in a circuit, or industrial applications like detecting part objects moving on a conveyor belt. In this case, the calibration turn should be performed at a known velocity in order to find the equations illustrated in Figure 31(b). The resulting motion in illustrated in Figure 32.

Table 2. Computed and filtered radius measurements

Real r (cm)	Before filtering		After filtering	
	Computed r	Δr	Filtered r	Δr
40.5500	41.0339	0.4839	40.5181	0.0319
40.5500	40.9036	0.3536	40.5869	0.0369
40.5500	40.9312	0.3812	40.6443	0.0943
40.5500	40.6789	0.1289	40.6323	0.0823
40.5500	40.4819	0.0681	40.5814	0.0314
40.5500	40.5974	0.0474	40.5680	0.0180
40.5500	41.1886	0.6386	40.6858	0.1358
40.5500	40.8985	0.3485	40.7107	0.1607
40.5500	40.7780	0.2280	40.7000	0.1500
40.5500	40.6549	0.1049	40.6488	0.9888
40.5500	40.5601	0.0101	40.5804	0.0304
40.5500	40.9075	0.3575	40.6134	0.0634

Table 3. Quadratic error measurement

Before filtering		After filtering	
e_r^2	e_θ^2	e_r^2	e_θ^2
0.28803677	0.39177788	0.06396748	0.15064421

Further Work and Conclusions

In this paper we have presented a new mathematical procedure for finding the position of an object through a mathematical modelization of a trinocular system, based on the apparent area of the objects. The accuracy achievable with this method depends largely on the nature of the pixel noise present in the image after the segmentation procedure. For this reason, a careful image filtering and signal post-filtering have to be performed. Due to the system's geometry, the accuracy of the method does not significantly decrease due to small errors in the calibration parameters of the cameras. A hardware processor has also been developed for real-time tracking, through color segmentation, image filtering and apparent area measurement. The developed hardware can track up to 15 objects, as long as they present well-defined and distinguishable chromatic characteristics. High-density programmable logic devices (FPGAs) have been used as a flexible technology for implementing real-time vision algorithms. The system allows the use of high-speed digital cameras, disabling the analog video converters and entering digital video. The shape of the objects is indifferent to the system, provided that they present the same apparent area when they are placed at the same position with respect to the camera. In order to prove the accuracy of the trinocular system an experimental platform has also been developed.

We have proposed a trinocular system where the intrinsic parameters of the camera are not needed, and a slight idea of the extrinsic parameters is enough for estimating depth. It has been proved that a careful use of filtering can help in performing measures without the help of any matching points or structured light. The obtained results show how real-time performance is attained, since image processing is performed at video-rate through a specialized hardware while post-processing (computation of the trinocular algorithm and the subsequent filtering) is also achieved at video-rate by means of a standard PC computer. As post-processing can be realized more than enough in less than 20 ms, there is no need to run a real-time operating system on the PC host.

Table 4. Quadratic error measurement introducing camera calibration errors

After filtering	
e_r^2	e_θ^2
0.18120454	0.36240536

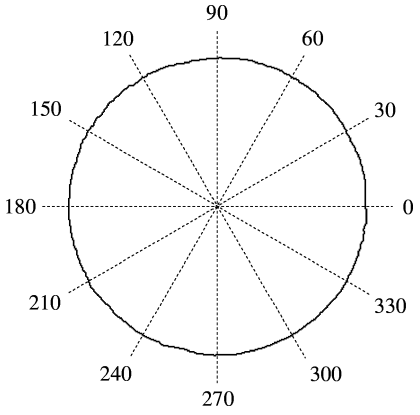

Figure 28. Reconstruction of the train track with error in the camera positions.

The system is being tested in a racing circuit in order to detect the instantaneous velocity of a motorbike rider in a curved area. In this case, the tracked object was the biker's jacket, which has well-known hue and saturation parameters. The biker was always approaching the static

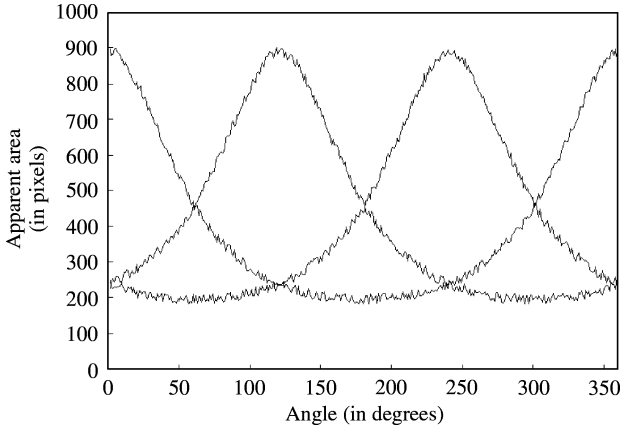

Figure 29. Area measured from cameras 1, 2 and 3 artificially affected with Gaussian noise.

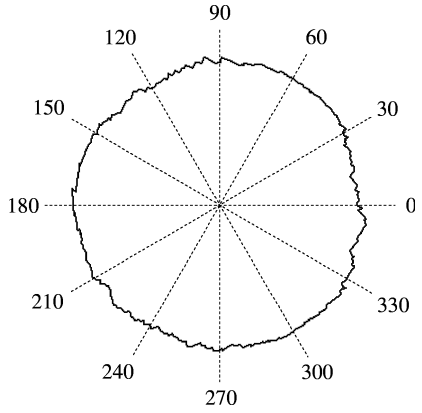

Figure 30. Resulting train track, after adding Gaussian noise to the apparent areas viewed from the three cameras.

cameras in the same position and orientation of his body. Thus, we could establish a relationship between the apparent area of the jacket and the position of the vehicle. The most interesting parameters in this application are deceleration and acceleration entering and exiting the curves.

For testing purposes, we are also applying the system to the robotic soccer competitions [25]. Here, the problem to be solved is controlling several moving robots playing soccer in an artificial field. There exist several categories depending on the size of the robots and the field [26]. In the medium and large size categories, it is not possible to cover the whole field by means of an overhead camera facing down. For this reason, perception and computation has to be performed on-board the robot. However, we are investigating the possibility of adapting our trinocular system in order to have a global view of the "world", simplifying the pose detection of the objects. It has been proved that fast spatial sensing is very important for these kinds of emulated competitions: the faster the spatial sensing, the faster feedback loops for controlling robot movements, and, thus, the better the team can play [27]. For this reason, a real-time vision system suits this application efficiently. Moreover, not only the ball can be tracked in

Table 5. Error measurement for the path reconstruction

After filtering	
e_r^2	e_θ^2
0.7567615	1.6670101

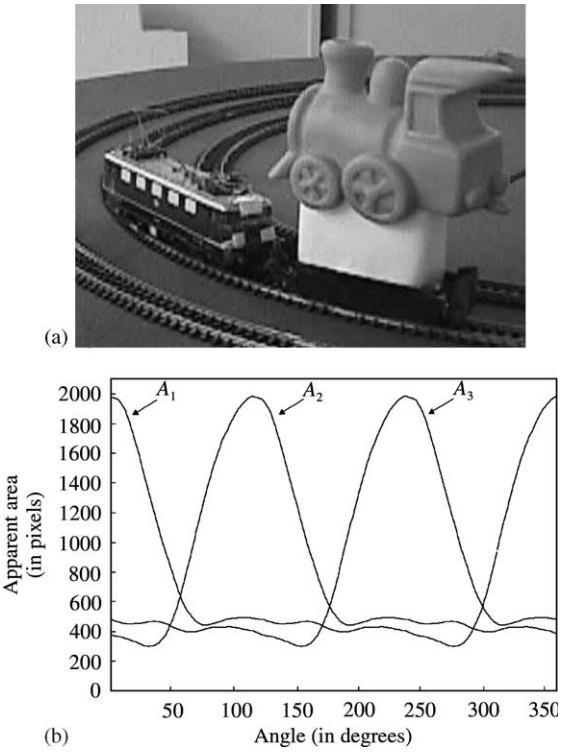

Figure 31. (a) A more complex object is used to measure its apparent area; (b) area measured from cameras 1, 2 and 3.

an efficient manner, but the robots can be tracked also by using the knowledge of their apparent area.

In the future, the system should be able to handle occlusions. One possible solution goes through taking into account the object dynamics, for instance using a global KF. We are also concerned about the behavior of

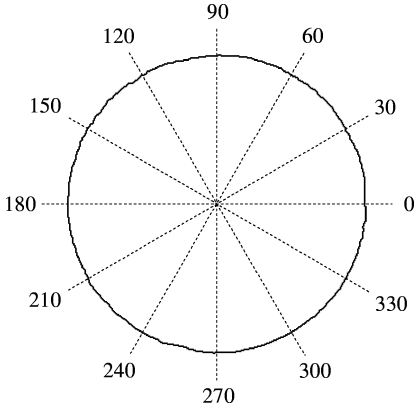

Figure 32. Reconstructed train track.

Table 6. Error measurement for the reconstruction of the path described by the toy train.

After filtering	
e_r^2	e_θ^2
0.10364789	0.19074945

our mathematical models when the objects move in 3D. Thus, the scenario is being modified generating known undulations on the tracks. In this way, the same testing platform can simulate 3D motions, by means of gentle curves and slopes. These vertical motions can be detected by the different cameras as a displacement of the object in the y axis of the image, and it can be processed independently from the 2D variables (r, θ). Thus, finding the height is reduced to a simple calibration problem, which is independently considered for each camera. This calibration depends totally on the computed distance between the object and each camera.

The possible applications of this system include object-tracking tasks, where either high processing speed is needed or video-rate performance is necessary.

Acknowledgements

This work has been funded by the CICYT MAR97-0925-C02-02 ''Garbi: Sistemas de control asistido de dos vehículos submarinos teleoperadores coordinadamente'' and CICYT TAP98-0955-C03-02 project ''Diseño de agentes físicos (DAFNE)'' of the Spanish government, and the 1998GR00234 ''Grup de Recerca Consolidat de Sistemes Integrats'' of the Catalan goverment.

References

1. Ito, M. & Ishii, A. (1986) Three-View Stereo Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**: 524–532.
2. Kitamura, Y. & Yachida, M. (1990) Three-dimensional data acquisition by trinocular vision. *Advanced Robotics* **4**: 29–42.
3. Batlle, J., Mouaddib, E. & Salvi, J. (1998) A Survey: Recent Progress in Coded Structured Light as a Technique to Solve the Correspondence Problem. *Int. Journal of Pattern Recognition* **31**: 963–982.
4. Shen, J. & Paillou, P. (1996) Trinocular sterovision by generalized hough transform. *Pattern Recognition* **29**: 1661–1672.
5. Quan, L. (1998) Algebraic Relations among Matching Constraints of Multiple Images, INRIA Technical Report 3345.
6. Weldon, E.J. & Liu, H. (1991) How accurately can direct motion vision determine depth? *IEEE Conference on Computer Vision and Pattern Recognition*, Lahaina, Maui, HI, USA, pp. 613–618.
7. Huber, J. & Graefe, V. (1993) Motion Stereo for Mobile Robots, in *IEEE International Symposium on Industrial Electronics ISIE' 93*. Budapest, Hungary, pp. 263–270.
8. Pedersini, F., Sarti, A. & Tubaro, S. (1997) Robust Area Matching. *IEEE International Conference on Image Processing*, Santa Barbara, CA, USA, Vol. 2, pp. 704–707.
9. Pedersini, F., Sarti, A. & Tubaro, S. (1996) Accurate 3D Reconstruction from Trinocular Views through Integration of Improved Edge-Matching and Area-Matching Techniques, *VIII European Signal Processing Conference*, Trieste, Italy, Vol. 1, pp. 300–306.
10. Vuillemin, J., Bertin, P., Roncin, D., Shand, M., Touati, H. & Boucard, P. (1996) Programmable Active Memories: Reconfigurable Systems Come of Age. *IEEE Transactions on VLSI Systems* **4**: 56–59.
11. Sethi, I.K. & Jain, R. (1987) Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9**: 56–73.
12. Deriche, R. & Faugeras, O. (1990) Tracking line segments, *First European Conference on Computer Vision*, Antibes, France, pp. 259–268.
13. Bascle, B. & Deriche, R. (1994) Region tracking through image sequences, INRIA Research report no. 2439.
14. García, R. Batlle, J. & Bischoff, R. (1996) Architecture of an object-based tracking system using colour segmentation. *Proceedings of the International Conference on Image and Signal Processing*, Manchester, UK, pp. 299–302.
15. Bouthemy, P. & François, E., Motion segmentation and qualitative dynamic scene analysis from an image sequence. *International Journal of Computer Vision* **10**: 157–182.
16. Carron, T. & Lambert, P. (1994) Color Edge Detection using jointly hue, saturation and intensity, In: *IEEE International Conference on Image Processing*, Austin, TX, Vol. 3, pp. 977–981.
17. Kim, H.C. & Park, H.W. (1996) Signal adaptative postprocessing for blocking effects reduction in JPEG images, In: *IEEE International Conference on Image Processing*, Lausanne, Switzerland, p. 17A5.
18. Celenk, M. (1990) A color clustering technique for image segmentation. *Computer Vision, Graphics and Image Processing* **52**: 145–170.
19. Perez, F. & Kock, C. (1994) Toward color image segmentation in analog VLSI: Algorithm and hardware. *International Journal of Computer Vision* **12**: 17–42.
20. Regincós-Isern, J., Batlle, J. (1996) A system to reduce the effect of CCDs saturation, In: *Proceedings of the IEEE International Conference on Image Processing*, Lausanne, Switzerland, Vol. 1, pp. 1001–1004.
21. Kanatani, K. (1991) Computational Projective Geometry. *International Journal of Computer Vision, Graphics and Image Processing* **54**: 333–348.
22. Isermann, R. (1991) *Digital control systems*. Springer-Verlag, London, UK.

23. Laplante, P.A. (1997) *Real-time systems design and analysis: an engineers handbook*, IEEE Press, New York, USA.
24. Toscani, G. (1987) *Systèmes de Calibration et Perception du Mouvement en Vision Artificielle*. Ph.D. Thesis, Université Paris Sud.
25. Oller, A., de la Rosa J. LI., Garcia, R., Ramon, J. A. & Figueras, A. (1999) Micro-robots playing soccer games: a real implementation based on a multi-agent decision-making structure. *Intelligent Automation and Soft Computing*, in press.
26. Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara & Osawa, E.-I. (1997) Robocup: a challenge problem for AI. *Artificial Intelligence Magazine* **18**: 73–85.
27. Sargent, R. Bailey, B., Witty, C., & Wright, A. (1997) Dynamic object capture using fast vision tracking. *Artificial Intelligence Magazine* **18**: 65–72.