PERGAMON

Pattern Recognition 32 (1999) 1961–1977

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/pattcog

# Feature correspondence and motion recovery in vehicle planar navigation

J.M. Sanchiz*, F. Pla

*Department of Computer Science, University Jaume I, Campus Penyeta Roja, 12072 Castelló, Spain*

Received 5 June 1997; received in revised form; accepted 2 September 1998

**Abstract**

This paper describes a strategy to feature point correspondence and motion recovery in vehicle navigation. A transformation of the image plane is proposed that keeps the motion of the vehicle on a plane parallel to the transformed image plane. This permits to define linear tracking filters to estimate the real-world positions of the features, and allows us to select the matches that accomplish the rigidity of the scene by a Hough transform. Candidate correspondences are selected by similarity, taking into account the smoothness of motion. Further processing brings out the final matching. The methods have been tested in a real application. © 1999 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

*Keywords:* Feature correspondence; Motion estimation; Autonomous vehicles

## 1. Introduction

The general problem of motion analysis from image sequences can be stated as to finding the $3 \times 3$ rotation matrix, $\mathbf{r}_{3D}$, and the 3D translation vector, $\mathbf{t}_{3D}$, that the camera co-ordinate system has to undergo in two consecutive images in order to match the projection of the scene viewed in both images. This is an ill-defined problem which can lead to several solutions for $\mathbf{r}_{3D}$ and $\mathbf{t}_{3D}$, and, due to the speed-scale ambiguity, only the direction of the translation can be recovered. The computation of these parameters is based on a former computation of the projected motion, that is, the motion of the projected scene on the image plane. The problem of finding the projected motion is approached by either the computation of the optical flow, basically using intensity differences in two consecutive images [1,2], or by the computation of feature correspondences [3–7]. This latter method seems more reliable for real-time, real-world applications since, once the features have been selected, the amount of data to process is significantly reduced.

In real-world applications some constraints are usually applied to the general problem [8,9], for example: the motion is known, then a 3D map of the scene can be recovered [10,11]; or there exist some landmarks in the scene whose real-world positions can be used as a reference [12–14]; or some of the motion parameters are fixed, a rotation angle or a component of the translation vector [15].

The latest situation is usually the case in autonomous vehicle navigation. A simple and quite common configuration of the camera is shown in Fig. 1. The camera is mounted on the vehicle and its height from the ground, $v$, and tilt angle, $\varphi$, are fixed. In some cases the camera could be mounted vertically, but this cannot be done generally since the mobile robot can occlude part of the field of view. The roll angle is set to zero as the vehicle is assumed not to roll. It is also assumed that the features lie on the ground plane. Such a configuration can be found in some works about road-following [9,15], indoor

* Corresponding author. Tel: 0034 964 72 83 53; fax: 0034 964 72 89 35.

*E-mail address:* sanchiz@inf.uji.es (J.M. Sanchiz)


PII: S0031-3203(98)00139-3

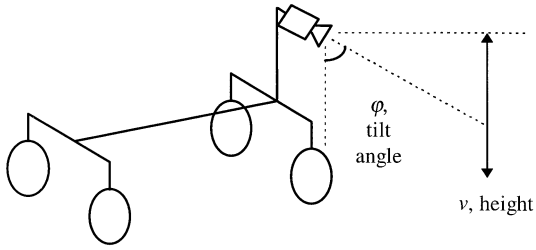


Fig. 1. Camera mounted on a vehicle.

navigation [10] or outdoor scenarios [11]. Feature tracking is a usual approach to motion estimation, although the nature of the extracted features depends on the type of scenes we are dealing with. Feature point tracking is quite a common approach, where the points are usually extracted from the grey-level images by some corner detector. This is the configuration that will be followed in the present work: an approach to the feature correspondence and motion recovery within this framework will be presented, together with a real-world application.

Once features have been extracted, a correspondence in consecutive images is needed to provide an estimation of the ego-motion of the vehicle. From the feature correspondence the motion parameters can be recovered by minimisation [16,17]. Then, from the motion parameters the present positions of the features can be estimated by updating the tracking filters assigned to each feature with the new observations. Kalman filters are commonly used for this task. Nevertheless, finding the correct feature correspondence is the key to satisfactory performance of the method. This is known as the data association problem [18,19], and consists of assigning each feature to the correct tracking filter.

Several works that address feature correspondence can be found in the literature. Some of them assume independently moving features, and find the correspondence by assuming smooth trajectories, applying convenient constraints [3,4], while they consider no attribute or characteristics of the features which could be used to perform an initial matching by similarity. Other works [5–7] preserve the affine structure of the feature set by assuming a rigid object or scene, but they assume that exactly the same number of features is available in the two images, and cannot cope with misdetected features.

In our approach, we take advantage of the special configuration of the camera and motion undergone in autonomous navigation, that is, the vehicle moves on the ground plane. Another constraint which applies is the rigidity of the scene. We assume that no other moving objects appear in the field of view, and we will take advantage of the global rigidity to reject outliers after an initial matching based on similarity between features. In this problem the motion is 2D, but is has 3D components when expressed in the camera coordinate system. Some of the computations inherent to the proposed approach require less effort if the image plane is parallel to the plane of motion, but these planes are not parallel in general. Therefore, we propose to remove the perspective effect by defining the *virtual image plane*, as the image plane which would have been used if the camera was exactly in a vertical position. The features are extracted from the original images and transferred to the virtual image plane, so one does not have to transfer whole grey-level images but just the selected features, saving computing time. The transformation is fixed and its parameters come from the camera calibration data.

A similar re-projection scheme, the *inverse perspective mapping* has been used to compute optical flow [20], or to find regions belonging to the ground plane in stereo images [21]. But the inverse perspective mapping re-projects a whole image while in our approach just a few number of feature positions are re-projected.

Some tasks can be simplified when working on the virtual image plane, like the rejection of outliers since, due to the rigidity of the scene, all of the correspondences have to be arisen from a rotation and a translation of this plane. Also, since the motion of the vehicle is parallel to the virtual image plane, it can be expressed as a 2D motion. Working in 2D simplifies the problem and saves computational efforts in some stages of the method. Another important advantage of using the virtual image plane is that the Kalman filters that estimate the position of each feature remain linear because the depth of the features in the virtual camera coordinate system does not change with motion on the ground plane.

The rest of the paper is organised as follows: Section 2 presents the problem this work is focused on. Section 3 presents the approach to feature correspondence. Section 4 explains the design of the tracking filters assigned to the features. Section 5 presents results with real image sequences in the context of an autonomous navigation application. Finally, some conclusions are drawn in Section 6.

## 2. Problem statement

The visual information registered from a camera on an autonomous vehicle can be mainly used, among others, to provide information of the frame-to-frame motion to the vehicle control system. From this information the absolute trajectory and position of the vehicle can be worked out. Since the vehicle moves over the ground, its motion is two-dimensional, the motion parameters consist of a $2 \times 2$ rotation matrix, $\mathbf{r}_{2D}$, and a 2D translation vector, $\mathbf{t}_{2D}$.

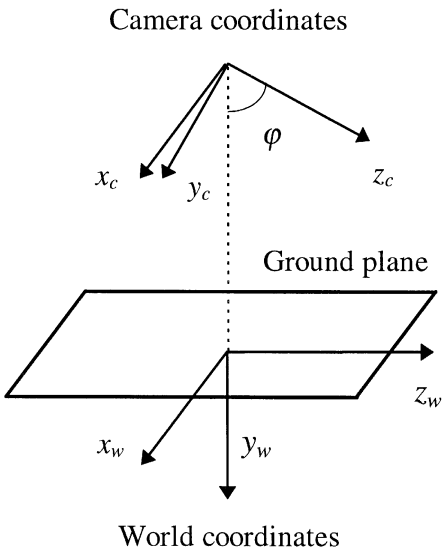


Fig. 2. Camera and world coordinates.

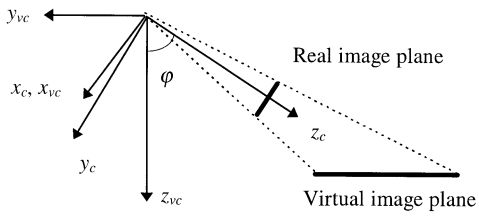


Fig. 3. Virtual vertical camera.

The relation between the camera and the world coordinate systems can be seen in Fig. 2. At time instant 0 (first frame) axis $x_w$ is aligned with axis $x_c$, in further frames this alignment will no longer exist (in general), given that the vehicle rotates and translates over the ground. With this camera configuration the vehicle motion is expressed as a 3D motion in the camera coordinate system. To avoid this, and constrain the camera motion to be 2D, we introduce the *virtual vertical camera* (Fig. 3). Its coordinate system is defined by applying a rotation of angle $\varphi$ to the real camera coordinate system around axis $x_c$,

The advantage of using the *virtual camera* is that, since the $(x_{vc}, y_{vc})$ plane is parallel to the ground plane, the vehicle motion will remain 2D when expressed in its coordinate system. There is no translation along the $z_{vc}$ direction, which will be convenient when finding the feature correspondences and when tracking the features through an image sequence. This will also make the Kalman filters defined to estimate the feature positions to remain linear.

The approach here presented takes advantage of several assumptions, and can be summarised as follows:

- *Ground plane motion*: Common in vehicle navigation. The features are transferred to the so called virtual image plane, which is parallel to the plane of motion.
- *Similarity between features*: An initial matching is made based on similarity measurements of the features in consecutive images.
- *Smoothness of motion*: The previous motion parameters are used to define search regions for candidate matches.
- *Rigidity of the scene*: The global rigidity is used to reject outliers through a Hough transform-like technique.
- *Poor feature extraction*: The tracking filter assigned to a feature is still updated if no new observation is found for it, and it is taken into account in the next frame. The method copes with misdetected features.
- *Planar feature set*: The features are assumed to lie on the same plane.

The transformation from the real image plane to the virtual image plane is done by applying a fixed rotation followed by a projection. The focal length, $f$, and the tilt angle, $\varphi$, are known and part of the calibration data. Let $(x_c, y_c, z_c)^T$, where $z_c = f$, be the camera co-ordinates of a point in the image plane. Let $(x'_{vc}, y'_{vc}, z'_{vc})^T$ be the coordinates of the same point expressed in the virtual camera coordinate system, and let $(x_{vc}, y_{vc}, z_{vc})^T$ be the coordinates of the corresponding projection of the same point on the virtual image plane. These coordinates can be found as follows:

$$\begin{pmatrix} x'_{vc} \\ y'_{vc} \\ z'_{vc} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ f \end{pmatrix};$$

$$\begin{pmatrix} x_{vc} \\ y_{vc} \\ z_{vc} \end{pmatrix} = \frac{f}{z'_{vc}} \begin{pmatrix} x'_{vc} \\ y'_{vc} \\ z'_{vc} \end{pmatrix}; \quad \begin{pmatrix} x_{vc} \\ y_{vc} \\ z_{vc} \end{pmatrix} = \begin{pmatrix} f \dfrac{x_c}{y_c \sin\varphi + f \cos\varphi} \\ f \dfrac{y_c \cos\varphi - f \sin\varphi}{y_c \sin\varphi + f \cos\varphi} \\ f \end{pmatrix}. \quad (1)$$

Assuming a previously calibrated camera, the calibration data should include the camera height, $v$, the tilt angle, $\varphi$, the focal length, $f$, and the real size of the image plane, expressed for example as introduced by Tsai [22]: number of sensors and distance between adjacent sensors in both directions of the image plane. No radial distortion is assumed. As mentioned, the calibration procedure is explained in detail in [22], but briefly consists of placing a set of coplanar test points whose 3D positions and projections are known (for simplicity the points can be placed on the ground plane), and finding the intrinsic and extrinsic calibration parameters by solving

an over-determined linear system of equations by minimisation.

Since the motion in undergone in the ground plane (parallel to the virtual image plane), the motion of the projected features in the virtual image plane can be obtained from the frame-to-frame motion. Let $\mathbf{r}_{k,k-1}$ ($2\times2$ rotation matrix) and $\mathbf{t}_{k,k-1}$ (2D translation vector) express the frame-to-frame motion, then the motion of the projected features can be expressed as

$$\begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k = \mathbf{r}_{k,k-1} \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} + \frac{f}{z_{vc}} \mathbf{t}_{k,k-1}, \tag{2}$$

and vice versa,

$$\begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} = \mathbf{r}_{k,k-1}^{-1} \left[ \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k - \frac{f}{z_{vc}} \mathbf{t}_{k,k-1} \right] \tag{3}$$

where the superscript *proj* indicates coordinates of the projected features, and $z_{vc}$ is the real depth of the feature, whose value is known from the calibration and equal to the camera height, since the features lie on the ground.

The frame-to-frame motion, as defined above, is opposed to the motion of the camera and so to the motion of the vehicle, that is, if the camera goes forward the projected features move backwards on the image plane. The absolute motion of the vehicle can be computed recursively from the frame-to-frame motion. Let $\mathbf{R}_k$ ($2\times2$ rotation matrix) and $\mathbf{T}_k$ (2D translation vector) be the absolute motion of the vehicle in world coordinates, then:

$$\begin{aligned} &\mathbf{R}_k = \mathbf{R}_{k-1}\,\mathbf{r}_{k,k-1}^{-1}; && \mathbf{R}_0 = \mathbf{I}, \\ &\mathbf{T}_k = \mathbf{T}_{k-1} - \mathbf{R}_k\,\mathbf{t}_{k,k-1}; && \mathbf{T}_0 = \mathbf{0}. \end{aligned} \tag{4}$$

The problem is to find $\mathbf{r}_{k,k-1}$ and $\mathbf{t}_{k,k-1}$ through a sequence of images. This implies to select the features and to track them estimating their real-world positions.

## 3. Feature correspondence

The kind of features to use depends mostly on the application, but a common characteristic is that they have to remain as much stable as possible through the sequence. So, line segments have been widely used mainly in indoor navigation [23,24], where doors, corridors, walls, and other man-made objects are quite common in the scenes. Corners computed from raw, grey-level images have also been widely used, either in indoor or outdoor applications [3,25,26], and some works can be found that try to unify both kind of features in a motion estimation framework [27,28]. Other useful features are segmented regions [29], dominant points in contours [30,31], or curves [32].

Due to the special characteristics of the application to which this work is mainly addressed, we have used points as features to be tracked (line segments can be discarded due to the nature of the outdoor scenes we deal with). So for the rest of the work we assume that a set of feature points is available for every image. The method to find the correspondence and motion recovery is independent of the way the feature points were extracted, the only requirement is that they are stable and that some measurements of the characteristics of the points can be provided.

A kind of feature is stable if the detected features do not often appear and disappear from image to image. Stability is closely related to the robustness of the method to extract the features. A quantification of the stability is desirable in order to check if a feature detection method is robust enough. In the following we propose a measurement for the stability of features:

Let $n_k$ be the number of features detected in image $k$. Let $det_{k,k-1}$ be the number of features that also appeared in image $k-1$ and were detected. Let $notdet_{k,k-1}$ be the number of features that also appeared at image $k-1$, but were not detected. Let $notview_{k,k-1}$ be the number of features in image $k$ that did not appear in image $k-1$ (due to the motion of the camera). Clearly $n_k = det_{k,k-1} + notdet_{k,k-1} + notview_{k,k-1}$. A possible definition of a figure of merit is

$$sta_k = 100\,\frac{\det_{k,k-1}}{n_k - notview_{k,k-1}} \quad \text{(in \%)}. \tag{5}$$

One drawback is that it is difficult, or almost impracticable, to automatically count the number of features belonging to each class. Instead, the features have to be matched manually.

Finding the correspondence between two sets of features in a pair of consecutive images is a key problem in feature-based motion estimation, and has been the focus of attention since the very first approach to motion recovery. One of the classic works was reported by Scott and Longuet-Higgins [4], another recently reported approach is the work by Salari and Sethi [33]. Our approach is different from these works in which it combines the assumptions of rigidity of the scene, smoothness of motion and similarity between features, taking advantage of the special configuration of the camera used in vehicle navigation to define the virtual image plane. We propose a tracking strategy based on three main stages: computation of candidate matches by smoothness of motion and similarity; rejection of outliers by the rigidity of the scene; and final matching and motion recovery, including the estimation of the feature positions from all of the previous observations. Working in the virtual image plane has the advantage of simplifying the problem since, once the feature points have been transferred to it, the problem becomes the one of finding the correspondence between two 2D point patterns undergoing a 2D motion, although these patterns may contain misdetected and newly appearing feature points.

The correspondence has to be solved as a first step to feature tracking, by which we mean to assign a new observation to a tracker (tracking filter) that estimates the real position of the feature based on previous observations; and also it has to be solved previously to estimating the camera motion from the set of correspondences. Each feature has a tracking filter (Kalman filter) assigned to it, that estimates its most likely position from all of the previous observations. The data to be estimated are the projections of the features on the virtual image plane, $(x_{vc}^{proj}, y_{vc}^{proj})$. The features observed in image $k_{-1}$ have already an associated tracker. The structure and design of the tracker is the same for all features, details on this will be given later. Features appearing in image $k$ have not been associated to a tracker yet, but they will be after the correspondence is found, then the new observation will be used to update the corresponding tracker. Features in image $k$ for which no correspondence is found are assumed to be new, and a new tracker is initiated for them.

The correspondence is found from features in frame $k$ to all of the estimated positions of the features that have been observed before, we will call these present tracks. Since features are assigned to present tracks, a feature that is not detected during some frames will be assigned to its corresponding track when detected again. When a feature is not detected a change of coordinates is made to its estimated position in order to update it on the virtual plane. The tracks are filtered before the correspondence is started to reject those that lie outside the present field of view, these are not taken into account in the present correspondence, which is done between tracks and features detected in the present frame.

The correspondence problem can be stated as follows: Let $n_k$ be the number of detected features in image $k$, and let $n_{k-1}$ be the estimated positions from the trackers at time $k-1$ that survive the filtering (tracks in image $k-1$). Solving the correspondence consists of finding a backward mapping, $\Psi_k$: $i \in [1, \ldots, n_k] \rightarrow j \in [0, 1, \ldots, n_{k-1}]$, and a forward mapping $\Theta_k$: $j \in [1, \ldots, n_{k-1}] \rightarrow i \in [0, 1, \ldots, n_k]$ following some criteria, and so that:

- $\Psi_k(i)$ is the corresponding track in image $k-1$ to feature $i$ in image $k$.
- $\Psi_k(i) = 0$ means feature $i$ has no correspondent in image $k-1$ (new appearing feature).
- $\Theta_k(j)$ is the corresponding feature in image $k$ to track $j$ in image $k-1$.
- $\Theta_k(j) = 0$ means feature $j$ has no correspondent in image $k$ (it has disappeared from the field of view, or it has not been detected).

The criteria we follow to find the mapping is to satisfy the following constraints:

- *Similarity between features*: Some properties can be computed for each feature that characterise them. Features in two consecutive images with close values of these characteristics are more likely to be correspondent. Since features in image $k$ are to be matched to tracks in image $k-1$, the characteristics of the last observation are stored for each track, and used to provide a measurement of similarity.
- *Smoothness of motion*: The expected motion is related to the previous motion estimation. We just look for corresponding features in an area near to the expected positions of the present ones.
- *Rigidity of the scene*: We assume that only the camera moves, so all the correspondences have to arise from the same motion.

A scheme of the proposed tracking strategy is shown in Fig. 4. The method can be divided into three main stages. Finding an initial matching by similarity between features. Making a selection by the global rigidity of the scene, and computing the rest of the matches. The motion parameters are found by minimisation, and the position of the features are estimated by updating the Kalman filters assigned to them with the new observations. A more detailed explanation of the tracking strategy is given in the following sub-sections.

### 3.1. Similarity between features

A general procedure to give a measure of similarity between features consists of computing a vector of characteristics for every feature, $(c_{1i}, c_{2i}, \ldots, c_{Ni})^T$. The meaning of these values is highly dependent on the method used to detect them (details on the characteristics that have been used in our application will be given in the results section). Then a distance between features can be defined as

$$d_{ij}^2 = \sum_{l=1}^{N} w_l^2 (c_{li} - c_{lj})^2, \tag{6}$$

where $d_{ij}$ is the distance between feature $i$ in image $k$ and feature $j$ in image $k-1$, and $w_l$ is the weight associated to characteristic $l$ ($l = 1, \ldots, N$).

Features are compared in terms of this distance, and a similarity matrix is built up. We call *one-way* matches to the entries that are a minimum either in their row or column, and *two-way* matches to the ones that are a minimum in both row and column. The latter will be used to further select the correct matches through a Hough transform-like technique, taking advantage of the rigidity of the scene.

### 3.2. Smoothness of motion

The previous estimated frame-to-frame motion $(\mathbf{r}_{k-1,k-2}, \mathbf{t}_{k-1,k-2})$ is used to search for correspondences. Given a feature $i$ in image $k$, the search area is located by

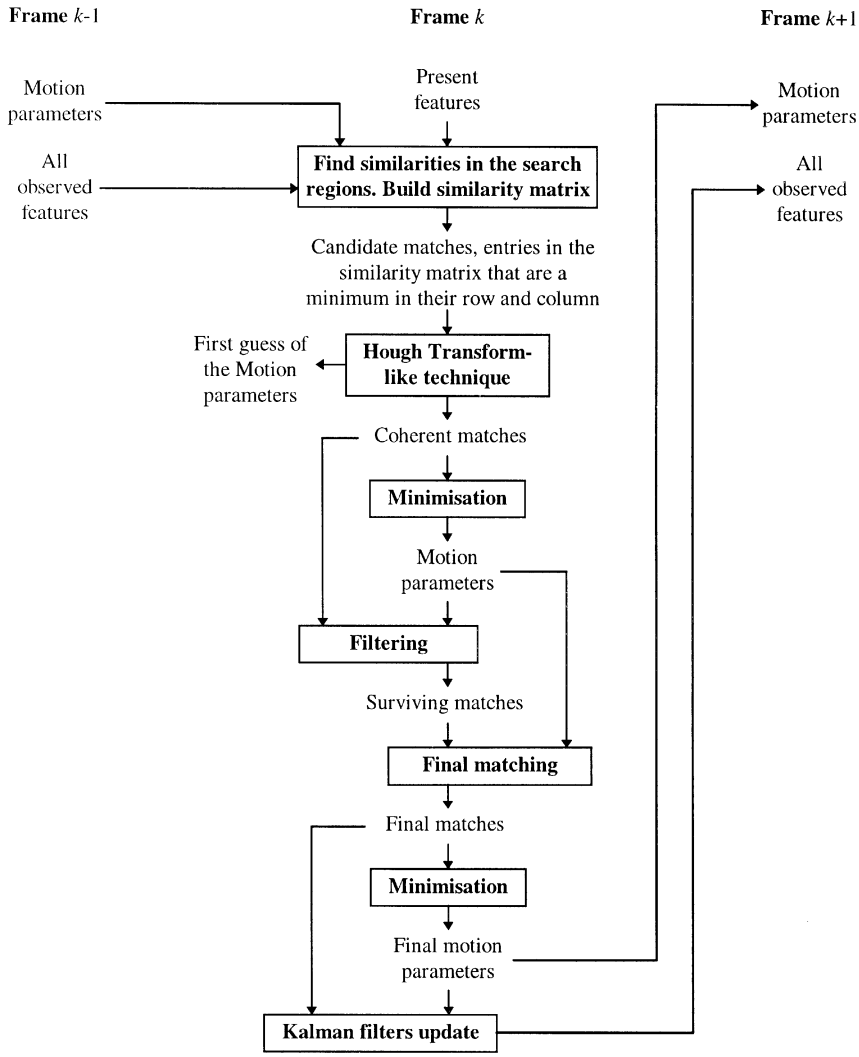


Fig. 4. Scheme of the method to solve the correspondence problem and perform a feature tracking.

back-projecting its coordinates on image $k-1$ (3), but an estimation of the depth, $z_{vc}$, is needed in Eq. (3), so we make the assumption that the depth is equal to the camera height, that is, the features lie on the ground. This is a reasonable approach for most autonomous navigation applications, where the camera is pointed to the ground; the same idea was used by Liu et al. [34] to recover the ego-motion from line and point correspondences on the ground plane. Finally the search area is set as an ellipse of centres **a** and **b**, whose main axis is oriented along the motion epipolar line. The coordinates of point **a** are set by back-projecting the coordinates of the feature point using a decreased value of $z_{vc}$, and the coordinates of **b** are set by back-projecting with an increased value of $z_{vc}$ (30% of increasing/decreasing has been used in our application).

The distance used can be considered a modified Euclidean distance since points inside a fixed distance threshold do not lie inside a circle, but inside an ellipse oriented in the direction of the epipolar line, found by using the previous value of the motion parameters. By this approach we favour the searching for correspondences in the direction of motion, which is mainly forwards although can have some rotational or transversal component. Nevertheless the type of distance used should not be crucial to the performance of the overall method, otherwise it would mean the method is not robust enough.

### 3.3. Rigidity of the scene

The rigidity of the scene constrains the correspondences to arise from the same motion of the camera. This

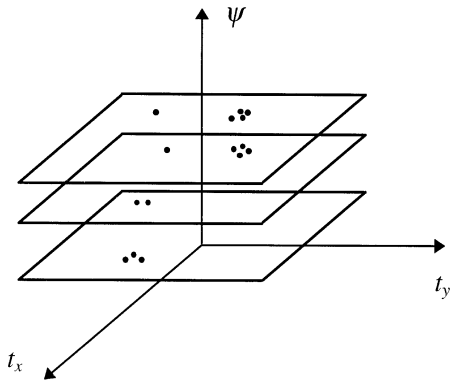


Fig. 5. Motion parameters space, planes of constant $\psi$ and points representing candidate correspondences.

means that, ideally, the values of $\mathbf{r}_{k,k-1}$ and $\mathbf{t}_{k,k-1}$ in Eqs. (2) and (3) have to be the same for all features. Once some candidate correspondences have been found by selecting similar features in the search areas, a Hough transform-like technique is applied to further select those ones having very close values of $\mathbf{r}_{k,k-1}$ and $\mathbf{t}_{k,k-1}$. For each correspondence, Eq. (2) can be separated into two equations with three unknowns: the rotation angle and the two components of the translation vector

$$\mathbf{r}_{k,k-1} = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}; \quad \mathbf{t}_{k,k-1} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}. \tag{7}$$

Fixing one unknown the other two can be found, so we fix and give values to the rotation angle and compute the translation. Doing this with all the correspondences we obtain a set of points in the 3D space formed by ($\psi$, $t_x$, $t_y$), arranged in planes of constant $\psi$ (Fig. 5). Applying a clustering to this set, and assuming that the erroneous correspondences will have random directions while the correct ones will represent nearly the same motion, we select the biggest cluster (supported by a bigger number of points) as a first estimate of ($\mathbf{r}_{k,k-1}$, $\mathbf{t}_{k,k-1}$). The points supporting it represent the set of good correspondences. A similar technique was used by Sanchiz et al. [29] to find correspondences of blobs.

The range of values to give to $\psi$ is centred on the last estimated value of the rotation angle, and it can be widened more or less to allow bigger variations in the rotation. $\psi$ varies in steps from a lower to an upper bound, the number of steps has to be set to a value that provides quite a fine estimate but not too much to reject correspondences that visually seem to come from the same motion of the camera (for example, we have used a range of 6° and 20 steps).

### 3.4. Procedure to find the correspondences

Following the criteria explained above leads to obtaining an initial set of correspondences of present features to existing tracks, and a first guess for the frame-to-frame motion. After selecting the good correspondences a better value of the motion parameters can be obtained by minimisation. The remaining matches can be computed by back-projecting the still non-matched features with the recovered motion and finding the most similar track in image $k-1$. The input to the process are the estimated feature positions and the motion parameters at frame $k-1$. The method provides the updated estimations of the feature positions and the new estimation of the motion parameters at frame $k+1$. An scheme of the approach is shown in Fig. 4. Its description is as follows:

*Step 1. Compute initial (or candidate) matches*

Build a distance matrix, **dm**. Each entry, **dm**$[i,j]$, represents the distance (or dissimilarity) between feature $i$ in image $k$ and track $j$ in image $k-1$:

*For* each feature $i$ in image $k$
  *For* each track $j$ in image $k-1$
    *If* track $j$ is inside the search region of feature $i$,
    make **dm**$[i, j] = d_{ij}$
    *else* make **dm**$[i, j] =$ *infinity*
  *EndFor*
*EndFor*

Find candidate correspondences as those pairs $(i,j)$ in which position **dm**$[i,j]$ is at the same time minimum in its row and its in column, this means that track $j$ is the most similar to feature $i$, and feature $i$ is the most similar to track $j$.

*Step 2. Select the matches arising from the same motion of the camera*

To select an initial set of consistent matches we propose a Hough transform-like technique which consists of giving values to the rotation angle and computing the corresponding translation for every pair of candidate matched features, thus obtaining a set of points in the 3D parameter space. After a clustering in this set, the points that support the biggest cluster provide the coherent matches. As it generally happens with Hough transform techniques, this method provides robustness while permits to aboard a 3D problem assuming just one independent unknown:

*For* each candidate correspondence ($i$, $j$)
  Give values to $\psi$ as explained above, and compute ($t_x$, $t_y$) from Eq. (2)
*EndFor*
Apply a clustering to the set of points ($\psi$, $t_x$, $t_y$)
Find the biggest cluster, ($\psi_0$, $t_{x,0}$, $t_{y,0}$)
Find a first guess for the motion parameters, ($\mathbf{r}_{0;k,k-1}$, $\mathbf{t}_{0;k,k-1}$).

The matches that originated the points that support the biggest cluster are marked as *coherent matches*, the others are discarded. The clustering method used here is the

sequential clustering with two thresholds introduced by Trahanias [35].

*Step 3. Compute the motion parameters by minimisation*
The coherent matches are used to prepare two sets of 3D points in world co-ordinates, points in the present frame and points in the previous frame. The projected co-ordinates $(x_{vc}^{proj}, y_{vc}^{proj})$ of all the features are known, and the depths $z_{vc}$ are set to the camera height. The positions in virtual camera coordinates of a feature are

$$(x_{vc}, y_{vc}, z_{vc})^{\mathrm{T}} = \left( x_{vc}^{proj} \frac{z_{vc}}{f}, y_{vc}^{proj} \frac{z_{vc}}{f}, z_{vc} \right)^{\mathrm{T}} \quad (8)$$

The minimisation procedure used is the one presented by Umeyama [16], this method has the advantage that it always finds a true rotation matrix. If we use the 3D positions of the points in the minimisation, a $3 \times 3$ rotation matrix and a 3D translation vector will be obtained as the motion parameters. But, as every pair of corresponding points have the same $z_{vc}$ coordinate, the motion will be in the plane $(x_{vc}, y_{vc})$, that is, the $z_{vc}$ coordinate of the translation is zero and the rotation is around the $z_{vc}$ axis. So we can work only with the $(x_{vc}, y_{vc})$ coordinates of the points and reduce the problem to a 2D one, finding a $2 \times 2$ rotation matrix, $\mathbf{r}_{1;k,k-1}$, and a 2D translation vector, $\mathbf{t}_{1;k,k-1}$. This simplification is possible because the feature point correspondence is found in the virtual image plane.

*Step 4. Apply a further filtering to the present matches*
The matches that represent a big variation from the motion parameters $(\mathbf{r}_{1;k,k-1}, \mathbf{t}_{1;k,k-1})$ are rejected. Once the motion is known, the new observed depth of a feature can be found by triangulation. From Eq. (2),

$$\begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k - \mathbf{r}_{1;k,k-1} \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} = \frac{f}{z_{vc}} \mathbf{t}_{1;k,k-1}. \quad (9)$$

Taking the module, which means to make the module of the translation due to this match the same as the module of the translation found by minimisation, and solving for $z_{vc}$ $(z_{vc} = z_{vc(\mathrm{obs})})$,

$$z_{vc\,(\mathrm{obs})} = f \frac{|\mathbf{t}_{1;k,k-1}|}{\left| \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k - \mathbf{r}_{1;k,k-1} \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} \right|}. \quad (10)$$

Once $z_{vc(\mathrm{obs})}$ is computed, we can solve for $\mathbf{t}_{k,k-1}$ $(\mathbf{t}_{k,k-1} = \mathbf{t}_{k,k-1\,(\mathrm{obs})})$:

$$\mathbf{t}_{k,k-1\,(\mathrm{obs})} = \frac{z_{vc\,(\mathrm{obs})}}{f} \left[ \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k - \mathbf{r}_{1;k,k-1} \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} \right]. \quad (11)$$

The filtering rejects the matches that produce a big variation in the depth, and those that represent a translation, assuming the same rotation, that differs from $\mathbf{t}_{1;k,k-1}$ in an angle bigger than a certain upper bound (in our application we have rejected variations in depth bigger than 30%, and variations of more than 5° in the direction of translation).

*Step 5. Find the final matching*
The remaining matches are found by computing $z_{vc\,(\mathrm{obs})}$ and $\mathbf{t}_{k,k-1\,(\mathrm{obs})}$ for all the possible correspondences of the still non-matched features. We compute the similarity distance for those ones whose $z_{vc\,(\mathrm{obs})}$ and $\mathbf{t}_{k,k-1\,(\mathrm{obs})}$ values are inside the limits, and successively pick up the most similar pairs:

*Repeat*
  *For* each still non-matched feature *i* in image *k*
    *For* each still non-matched feature *j* in image $k-1$
      Find $z_{vc(\mathrm{obs})}$ and $\mathbf{t}_{k,k-1(\mathrm{obs})}$ for match $(i, j)$ by triangulation
      *If* the change in $z_{vc}$ is less than 30% and the angle between $\mathbf{t}_{k,k-1(\mathrm{obs})}$
      *and* $\mathbf{t}_{k;k,k-1}$ is less than 5 degrees, make $\mathbf{dm}[i, j] = d_{ij}$ *else* make $\mathbf{dm}[i, j] =$ infinity
    *EndFor*
  *EndFor*
  Pick up the minimum in **dm**, (position $\mathbf{dm}[i, j]$)
  *If* $\mathbf{dm}[i, j] \langle\rangle$ infinity add this match to the set of coherent matches
*Until* $\mathbf{dm}[i, j] =$ infinity

*Step 6. Find the final value for the motion parameters*
A new (and definitive) value for the frame-to-frame motion $(\mathbf{r}_{k,k-1}, \mathbf{t}_{k,k-1})$ is found by minimisation using all of the matches. Umeyama's method is used again, as explained above.

## 4. Tracking features

Using the features transferred to the virtual image plane makes the tracking problem simpler. As this plane is parallel to the plane of motion, the $z_{vc}$ coordinate of the feature points remains unchanged in the virtual camera coordinate system while the vehicle moves, so the perspective projection of the features onto the virtual image plane represents just a constant scaling. This allows us to define linear Kalman filters to estimate the positions of the feature points. The 2D frame-to-frame motion undergone by the features is modelled as a linear discrete dynamic system to which a Kalman filtering is applied.

A tracking filter is initiated for every new feature appearing in the scene. Its function is to estimate the position of the feature in virtual camera coordinates from the observations of its projections, and from the estimated value of the motion parameters $(\mathbf{r}_{k,k-1}, \mathbf{t}_{k,k-1})$. More precisely, the data to estimate are the coordinates of the projection of a feature on the virtual image

plane, $(x_{vc}^{proj}, y_{vc}^{proj})$. Fixing the depth to the camera height ($z_{vc} = v$) the real-world position can be computed from Eq. (9).

The Kalman filter is used as a tracker, it estimates the best value of a state vector from a set of Gaussian noisy measurements in dynamic linear systems. In fact, the frame-to-frame motion can be expressed as a linear system if we use the coordinate axes of the virtual camera. The filter is updated recursively from each new observation. A deep discussion on Kalman filter theory can be found in [18,36], and applications of the Kalman filter to depth estimation in [37,38].

Let $\mathbf{x}_k$ be an $n$-dimensional state vector. From state $\mathbf{x}_{k-1}$, the system evolves to state $\mathbf{x}_k$ by applying the system model equation (12), where $\mathbf{\Phi}_k$ is the transition matrix and $\mathbf{\Gamma}_k\mathbf{u}_k$ is the input, but the new state is combined with noise, modelled as a Gaussian random $n$-dimensional vector $\mathbf{v}_k$, with $\mathbf{0}$ mean and covariance matrix $\mathbf{R}_1$. Let $\mathbf{y}_k$ be an $m$-dimensional measurement vector. From state $\mathbf{x}_k$, a measurement is obtained by applying matrix $\mathbf{C}_k$ (13), but the observation is corrupted again with noise, Gaussian random vector $\mathbf{e}_k$, with $\mathbf{0}$ mean and covariance matrix $\mathbf{R}_2$. From observation $\mathbf{y}_k$ and the previous estimation $\mathbf{x}_{k-1|k-1}$, a prediction of the state vector at instant $k$ is made, $\mathbf{x}_{k|k-1}$. Then, this prediction is updated with the Kalman gain matrix $\mathbf{K}_k$ to find the best prediction at instant $k$, $\mathbf{x}_{k|k}$. The covariance matrix of the state vector, $\mathbf{P}_{k|k}$, is predicted in the same way. The initial conditions are $\mathbf{x}_{0|0} = E[\mathbf{x}_0]$ and $\mathbf{P}_0 = cov[\mathbf{x}_0]$. The random variables $\mathbf{v}_k$ and $\mathbf{e}_k$ are supposed to be uncorrelated, $\mathrm{E}[\mathbf{v}_k\,\mathbf{e}_k^{\mathrm{T}}] = \mathbf{0}$

$$\text{System model:}\quad \mathbf{x}_k = \mathbf{\Phi}_{k-1}\,\mathbf{x}_{k-1} + \mathbf{\Gamma}_{k-1}\,\mathbf{u}_{k-1} + \mathbf{v}_k;\quad \mathbf{v}_k \in \mathrm{N}(\mathbf{0}, \mathbf{R}_1). \tag{12}$$

$$\text{Measurement model:}\quad \mathbf{y}_k = \mathbf{C}_k\,\mathbf{x}_k + \mathbf{e}_k;\quad \mathbf{e}_k \in \mathrm{N}(\mathbf{0}, \mathbf{R}_2). \tag{13}$$

Initial state: $E[\mathbf{x}_0] = \mathbf{x}_{0|0};\quad \mathrm{cov}[\mathbf{x}_0] = \mathbf{P}_0.$

Prediction at time instant $k-1$:

$$\mathbf{x}_{k|k-1} = \mathbf{\Phi}_{k-1}\,\mathbf{x}_{k-1|k-1} + \mathbf{\Gamma}_k\,\mathbf{u}_k. \tag{14}$$

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_{k-1}\,\mathbf{P}_{k-1|k-1}\,\mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{R}_1. \tag{15}$$

Prediction at time instant $k$:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\,\mathbf{C}_k^{\mathrm{T}}[\mathbf{C}_k\,\mathbf{P}_{k|k-1}\,\mathbf{C}_k^{\mathrm{T}} + \mathbf{R}_2]^{-1}, \tag{16}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k[\mathbf{y}_k - \mathbf{C}_k\,\mathbf{x}_{k|k-1}], \tag{17}$$

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k\,\mathbf{C}_k]\,\mathbf{P}_{k|k-1}. \tag{18}$$

### 4.1. Kalman filter design

The state vector is defined as

$$x_k = \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k. \tag{19}$$

The transition matrix is used to express the rotation

$$\Phi_k = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}_k, \quad \text{or similarly} \quad \Phi_k = \mathbf{r}_{k,k-1}. \tag{20}$$

The input part in Eq. (12) is used to model the translation

$$\mathbf{\Gamma}_k = \mathbf{I},\ \mathbf{u}_k = \begin{pmatrix} f\,\dfrac{t_x}{z_{vc}} \\ f\,\dfrac{t_y}{z_{vc}} \end{pmatrix}_k. \tag{21}$$

So, the transition from state $k$-1 to state $k$ is

$$\begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_k = \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix}_{k-1} \begin{pmatrix} x_{vc}^{proj} \\ y_{vc}^{proj} \end{pmatrix}_{k-1} + \begin{pmatrix} f\,\dfrac{t_x}{z_{vc}} \\ f\,\dfrac{t_y}{z_{vc}} \end{pmatrix}_{k-1}. \tag{22}$$

The measurements from the visual information are the coordinates of the projected features, $(x_{vc}^{proj}, y_{vc}^{proj})$, so the measurement matrix is the $2\times 2$ identity matrix, $\mathbf{C}_k = \mathbf{I}$.

Given a set of matches, $\Psi$, $\Theta$, where for feature $i$ in image $k$, the corresponding track in image $k-1$ is $j = \Psi(i)$, and similarly $i = \Theta(j)$, the Kalman filter associated to track $j$ is updated with the data observed for feature $i$. New Kalman filters are initiated for newly observed features, those for which $\Psi(i) = 0$. The Kalman filters for which no new observation is found, $\Theta(j) = 0$, just suffer a change of coordinates (2), so their uncertainty (covariance matrix of the state vector, $\mathbf{P}_{k|k}$) is not reduced in this frame. By this approach the system copes with misdetected features, that is, these features are still taken into account in the following frames by keeping record of their last estimated positions, but referred to the camera co-ordinate system at the present frame. If the feature is detected in a forthcoming frame, it will be assigned to the correct tracking filter.

The covariance matrices, $\mathbf{P}_0$, $\mathbf{R}_1$ and $\mathbf{R}_2$, are initiated as

$$\mathbf{P}_0 = \mathbf{R}_1 = \mathbf{R}_2 = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}. \tag{23}$$

where $\sigma_x$ and $\sigma_y$ are set to a certain fraction of the real size of the field of view. These matrices represent the uncertainty in the initial state $\mathbf{x}_0$ ($\mathbf{P}_0$), the uncertainty in the state update Eq. (12) ($\mathbf{R}_1$), and in the measurement Eq. (13) ($\mathbf{R}_2$). The covariance matrix of the state vector, $\mathbf{P}_{k|k}$, has been used in other approaches to define search regions when looking for candidate matches by means of defining a distance (the search region is the one that falls

inside a certain distance threshold). The Mahalanobis distance has been often used [18,19]. We do not use this statistical knowledge in our approach, instead the search regions are computed from the previous motion by allowing a certain variance range in the depth (see previous sections). This can be considered a practical approach which saves computations. Nevertheless, if the overall method is robust enough, its performance should not depend on the type of distance used to define the search regions when looking for candidate matches.

## 5. Experimental results

In our application, the context is an autonomous vehicle that navigates in an outdoor crop field. The scenes we deal with consist of a perspective view of a piece of field where only natural objects, plants, appear. The purpose of the application is to spray on the plants or weeds automatically. To accomplish this, the images are segmented to divide the scene into three classes, regions of class "soil", regions of class "plant", and regions of class "weed". The vehicle is equipped with a bar of nozzles to perform the spraying. The same images are used to identify the plants and to compute the motion parameters, which are then used to place the images on a map of the field, built up while the vehicle moves. Exploring the map along the nozzle bar allows us to open those nozzles that are over a plant or weed. The motion estimation is intended to be passed to the vehicle control system, thus closing the loop and trying to accomplish autonomous row following (real-time autonomous row following was reported by Sanchiz et al. [39], here the visual motion was obtained from a real-time Hough Transform-based row finder reported by Marchant and Brivot [40]).

The images are segmented with the method presented by Brivot and Marchant [41]. To detect features we use the contours of the regions of class "plant", and find dominant points in them. By using just plants, and not weeds, we can assume that all the feature points lie in a common plane. The plants (different type of vegetables like cabbages or cauliflowers) have a certain height (around 30 cm), while the weeds are at ground level. A contour following algorithm was applied to code the boundaries, although in a real-time implementation the contours can be found and chain-coded at frame rate (10 Hz) from a hardware chain encoder [42] (module that segments and finds the contour chain-code of the resulting blobs). The dominant points in the contours were found by applying the neural network-based algorithm for dominant point detection reported by Sanchiz et al. [30,31]. Briefly, the algorithm consist of training a neural network to provide a measurement of the relevance of a contour point, the net is applied to all the contour points thus giving a cyclic discrete signal, which is thresholded. The dominant points are computed as the centroids of the segments that overpass the threshold. The advantage of using a neural network-based algorithm is its possibility to speed up the process.

The tracking method explained in this paper has been tested with several image sequences (30 images each) obtained from a camera mounted on a manually driven vehicle undergoing zigzag motion. Fig. 6 shows some sample images from a sequence, the segmentation, the extracted contours, and the dominant points.

The stability of the dominant points was measured with the approach presented in Section 2. This sequence resulted in a stability of about 85%, so we may say that the dominant point detection algorithm behaves satisfactorily enough. The reason for the stability not to be better is mainly due to segmentation errors, that significantly change the contours of the objets in areas that were assigned to a different class in the previous image (a piece of plant that now appears as weed for example). The dominant point detection algorithm finds different points in these areas.

For our features, two translation and rotation-independent characteristics that can be used for similarity measurements are the convexity and the orientation of the contour in a small area around a dominant point. Assuming both characteristics have the same importance, we fix $N = 2$, $w_1 = 1$, and $w_2 = 1$ in Eq. (6). Fig. 7 shows the angles of convexity $\alpha$ and orientation $\beta$ at a dominant point $\mathbf{p}$. Two points are found on either side of the contour, $\mathbf{p}_a$ and $\mathbf{p}_b$, so that the distance between $\mathbf{p}$ and $\mathbf{p}_a$, and between $\mathbf{p}$ and $\mathbf{p}_b$ is as close as possible to a given value. The angle of convexity is the angle between segments $\mathbf{pp}_a$ and $\mathbf{pp}_b$, and ranges from $0^+$ to $180^\circ$ for convex areas and from $0^-$ to $-180^\circ$ concave ones. The angle of orientation is the angle between the bisecting line and the horizontal axis. The threshold distance for segments $\mathbf{pp}_a$ and $\mathbf{pp}_b$ has been fixed to 10% of the contour length.

Fig. 8 shows the matching process step by step. For this sequence the calibration was: camera height $v = 1200$ mm, tilt angle was $\varphi = 66^\circ$, focal length $f = 40$ mm; the vehicle speed was about 1.25 m/s and the frame rate was 200 ms. Two consecutive images with the contours and the dominant points outlined can be seen in Fig. 8a and b. Fig. 8c shows the features of both images transferred to the virtual image plane, together with the result of the initial matching. Fig. 8d shows the selected correspondences after applying the Hough transform. Fig. 8e shows the final correspondences and Fig. 8f the correspondences on the original image. The rate of successful correspondences was over 95% through a sequence of 30 images, this rate was determined by manually identifying the correct matches, the incorrect, and the missed ones in every image of the sequence.

From the frame-to-frame motion $(\mathbf{r}_{k,k-1}, \mathbf{t}_{k,k-1})$ the absolute position and orientation of the vehicle $(\mathbf{R}_k, \mathbf{T}_k)$ are found from Eq. (4). In order to measure the accuracy

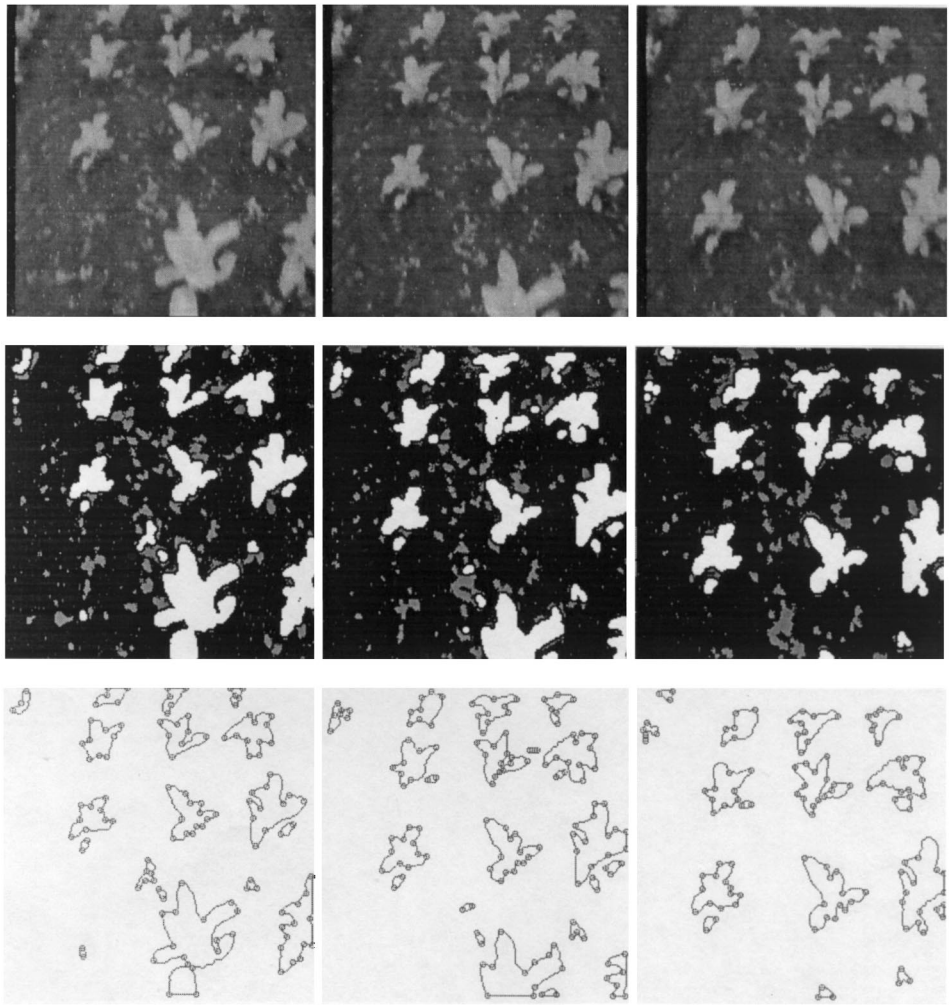

Fig. 6. First row: consecutive grey-level images. Second row: segmented in plants (white), weeds (grey) and soil (black). Third row: dominant points detected by a neural network-based algorithm.

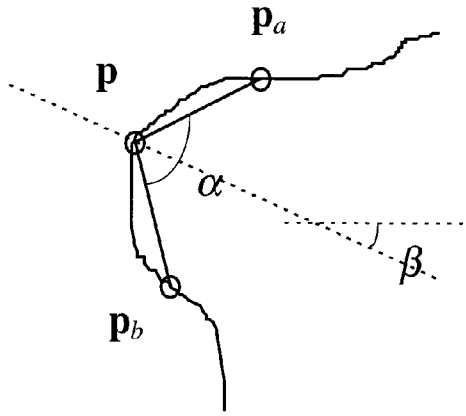


Fig. 7. Convexity, $\alpha$, and orientation, $\beta$, at a dominant point.

of the motion estimation, $\mathbf{R}_k$ and $\mathbf{T}_k$ were used to place every image over the ground plane, thus building a map of the field including the whole area travelled by the vehicle. Since the images overlap in a certain extent, a majority voting scheme was used to determine the classification of the pixels, counting the times that a pixel is assigned a certain class ("plant", "weed" or "soil"). The crop is aligned in rows, in order to check the accuracy of the method every plant was manually assigned to a certain row, and straight lines were fitted to the centres of the blobs of class "plant". The root-mean-square error of the fit, the parallelism and the distance between neighbouring lines are measurements that indicate the accuracy of the map, and so of the estimated motion. Fig. 9 shows the final matching for the first images of the sequence, and a map built in a $256 \times 512$ image at a scale

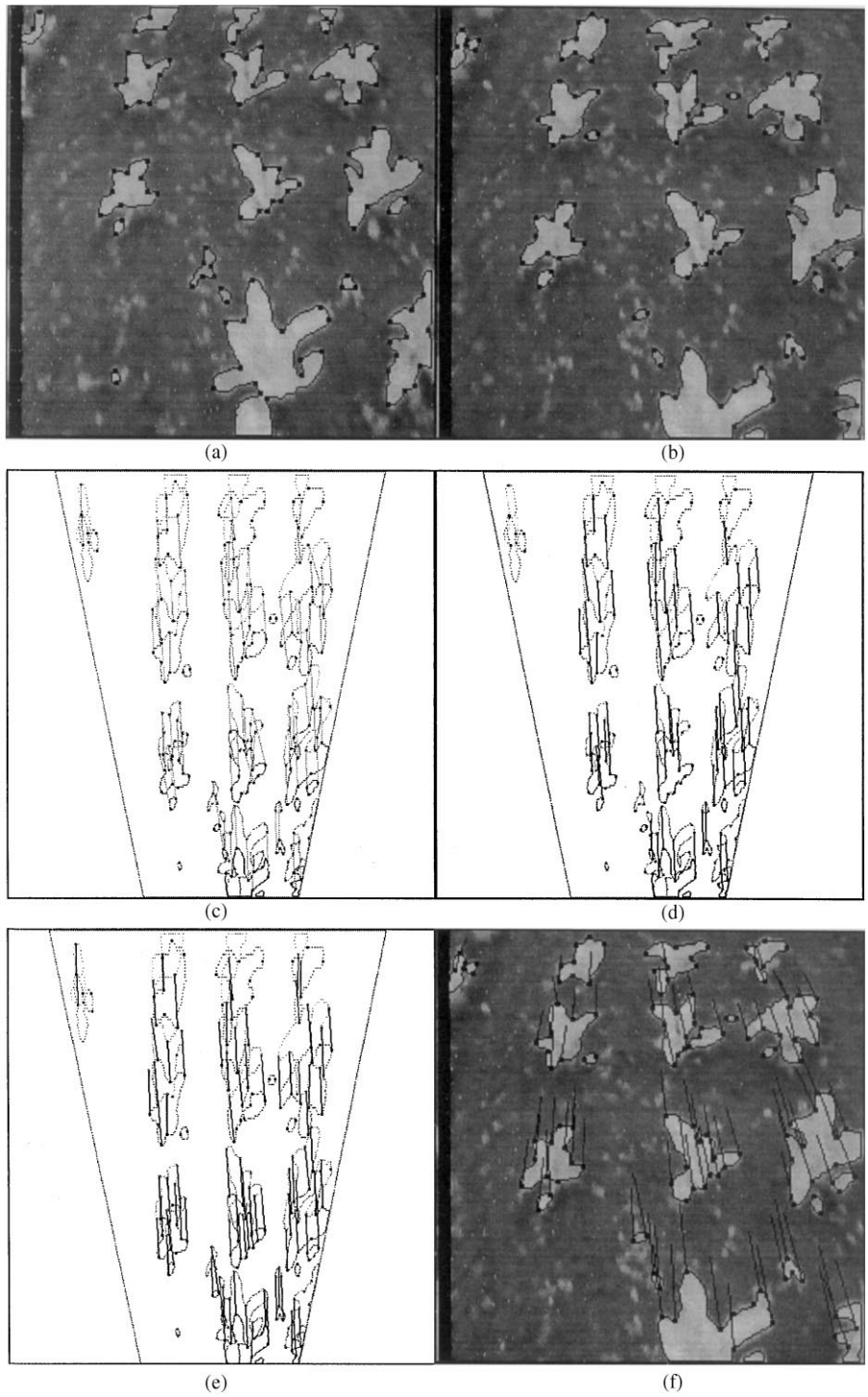


Fig. 8. (a), (b) Two consecutive images; (c) initial matching on the virtual image plane; (d) selected matches after applying the Hough Transform; (e) final correspondences; (f) correspondences over the original image.

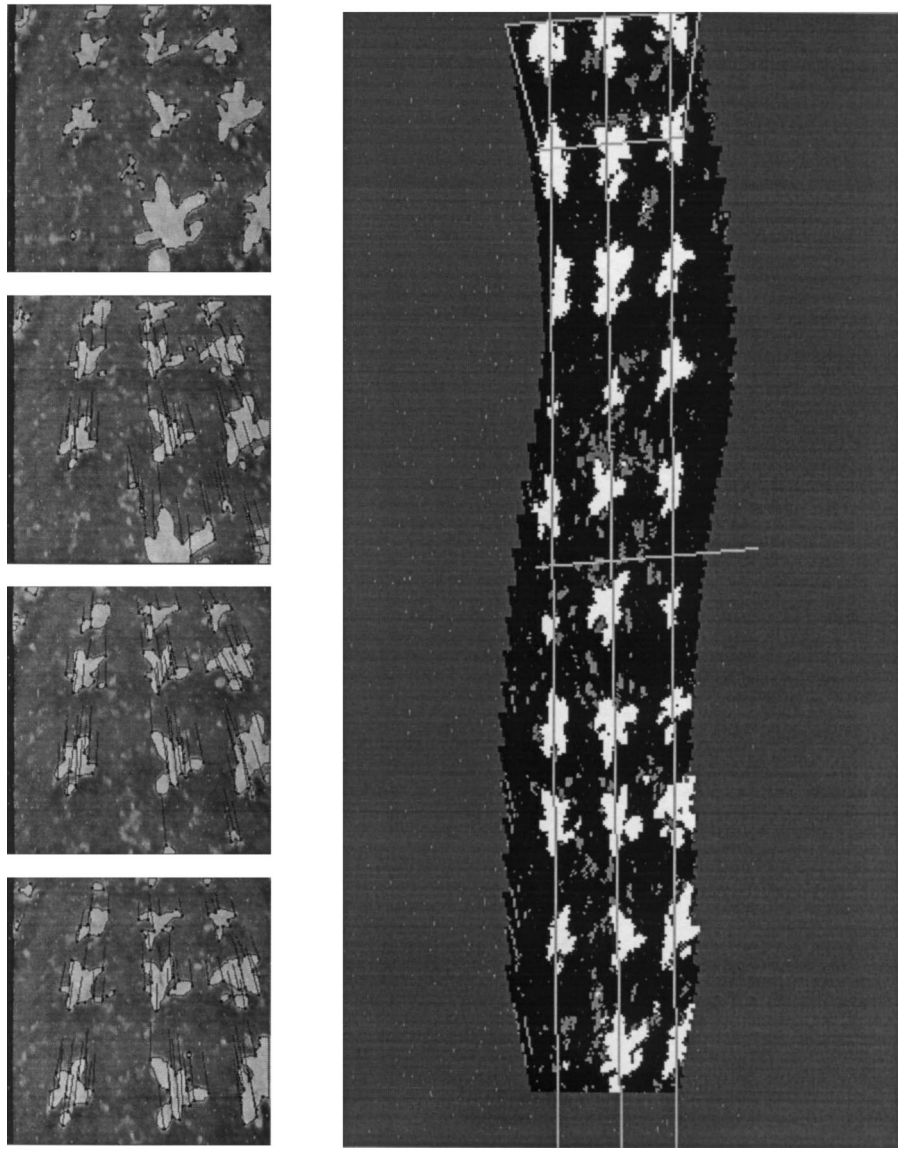

Fig. 9. Left: final matching in four consecutive images. Right: map built from the motion recovered from the feature point correspondences, with lines fitted to the centres of the plants in the rows. The last field of view and position of the nozzle bar are outlined.

of 15 mm per pixel, with the last field of view and position of the nozzle bar outlined. The rms error of the fitted lines was 32.7, 27.2 and 8.6 mm, respectively, the angle between neighbouring lines was 0.42 and 0.54°, and the distance, measured at the centre of the map, was 413.2 and 408.3 mm (400 mm is the approximate real-world distance between rows in the crop). It has also to be noted that the lines were fitted to the centres of the blobs of class "plant", which are not exactly over the lines that pass through the centre of the crop rows, nevertheless the results are quite satisfactory.

Fig. 10 shows another example from a different sequence. In this test the vehicle speed was 0.25 m/s and the frame rate 140 ms. The calibration was $v = 1770$ mm, $\varphi = 90°$ (vertical camera), $f = 28$ mm, and the map is drawn at 6 mm per pixel. In fact, in this sequence, the experimental vehicle was not used, and the camera was mounted through a bar to a tractor which moved along one side of the field. The distance travelled after 30 images is quite short, so this is not a good example to check the straightness of the rows of the map.

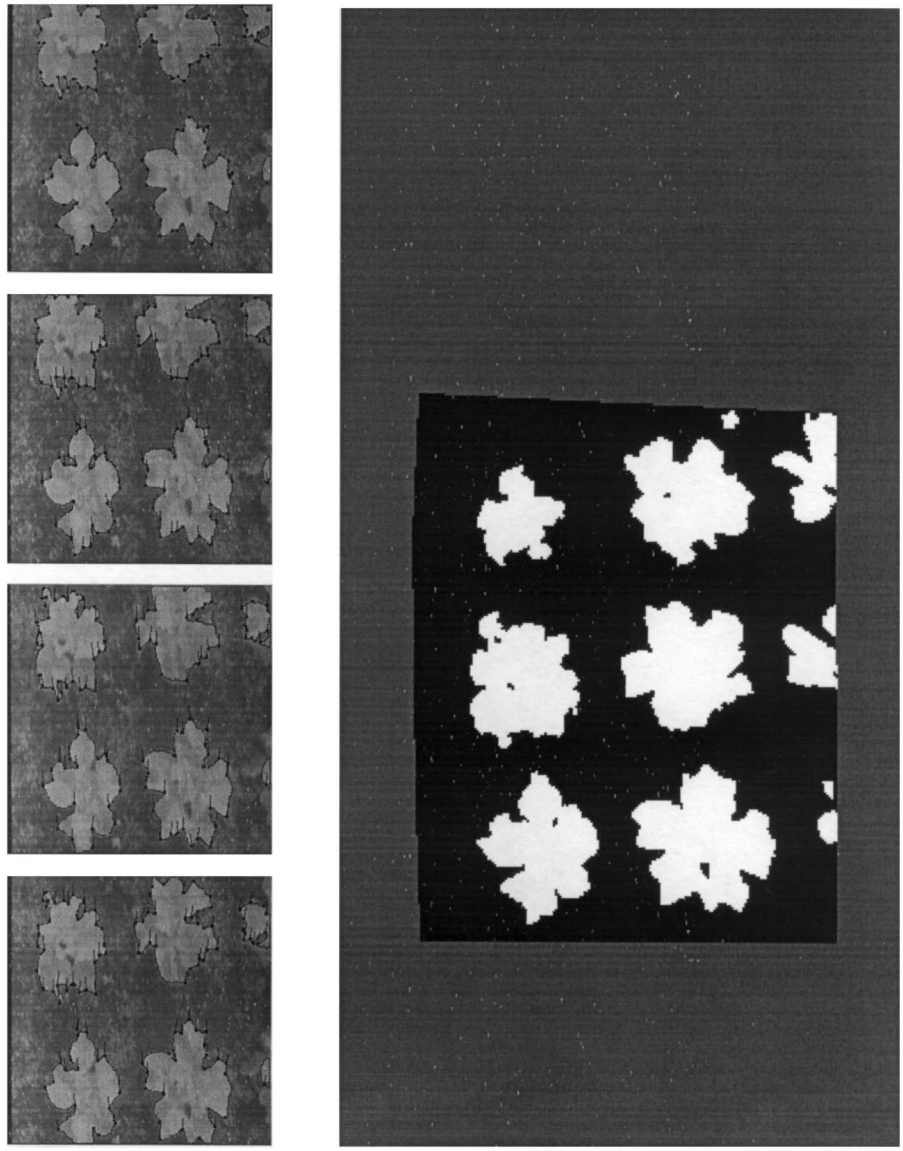

Fig. 10. Left: final matching in four consecutive images. Right: map built from the motion recovered from the feature point correspondences.

## 6. Conclusions

A feature point-based strategy to solve the correspondence problem and the tracking of features has been presented. The method is intended for autonomous navigation applications, where a general constraint is that the motion is undergone on the ground plane. The special configuration of the camera in this kind of applications is used to define the virtual image plane, parallel to the plane of motion. This simplifies the problem by reducing it to finding the correspondence between two 2D point patterns. It also permits a simple rejection of outliers by a Hough transform-like technique, and allows us to define linear Kalman filters to estimate the feature positions. A tracking strategy has been presented to solve the correspondence problem, track the features through a sequence, and recover the motion parameters. The strategy is based on the assumptions of smoothness of motion, similarity between features, and rigidity of the scene.

The system provides an ego-motion estimation at the present frame and the estimation of the feature positions in the real world. The tracking strategy, which copes with misdetected features, can be divided into three

main stages:

- A set of candidate matches is found by computing similarities between feature points in a search region, computed from the previous motion parameters for each new feature observation.
- The rigidity of the scene is used through a Hough transform-like technique to select the set of consistent matches, thus performing a rejection of outliers.
- The motion parameters are found by minimisation, and a final matching is computed. The tracking filters, Kalman filters defined to estimate the feature positions, are updated with the new observations.

The method has been applied to a real-world vehicle navigation application, consisting of an autonomous agricultural vehicle navigating in a crop. Dominant points in contours have been used as feature points. Measurements of similarity have been defined for them. The recovered motion has been used to build up terrain maps, which are used for the final objective of the application. Since the real value of the motion undergone by the vehicle in the tests is unknown, the recovered maps are compared to the real-world expectations by defining some application-oriented measurements, thus obtaining a qualitative idea of the accuracy of the recovered motion. The results obtained so far have been quite satisfactory for the objectives of the application.

## 7. Summary

This paper presents a strategy to find the correspondence of feature points in the context of vehicle planar navigation. From the point correspondences, the ego-motion is recovered, and a map of the area travelled by the vehicle is drawn up. Results of the methods in a real application, consisting of an autonomous agricultural vehicle navigating in a crop field, are also presented.

After a feature extraction, the feature points are tracked through an image sequence, finding the correspondence from frame to frame. The tracking strategy is based on the assumptions of similarity between features, smoothness of motion, and rigidity of the scene. The feature points are transferred to the defined virtual image plane, parallel to the plane of motion, to simplify the problem. Working in this plane allows us to select the candidate matches that accomplish the rigidity of the scene by a Hough transform technique, and also to define linear tracking filters (Kalman filters) to estimate the feature real-world positions. The method can cope with poor feature extraction, that is, if a feature point is miss-detected in some frames, it is still assigned to the correct tracking filter when detected again.

Very briefly, the tracking strategy can be divided into three main stages:

- First, candidate matches are computed by similarity. The most similar feature point in the previous frame, which lies inside a search region, is selected as a candidate match. A search region is defined for each new observation taking into account the previous motion parameters.
- The matches that accomplish the rigidity of the scene are selected through a Hough transform-like technique, these are the consistent matches. The method consists of, for each candidate match, drawing points in the motion parameter space (three parameters) and applying a clustering to this set of points. The biggest cluster provides a first guess for the motion parameters, while the matches that support it are considered the consistent matches.
- Finally, the complete matching is found. The motion parameters are computed by minimisation from all of the correspondences, and the new feature observations are assigned to the corresponding Kalman filters, which are updated. A new Kalman filter is defined for every feature appearing for the first time in the scene, to estimate their positions from the forthcoming observations of these feature points.

In our application, after a segmentation of the image, feature points are computed as dominant points in contours by a neural network-based approach. Some characteristics of these feature points are defined to perform similarity measurements. Finally, the accuracy of the results is discussed. Since the exact motion undergone by the vehicle is unknown, a qualitative idea of the accuracy is obtained by comparing the map, drawn from the recovered motion, to the real-world expectations.

## References

[1] B.K.P. Horn, B.G. Schunk, Determining optical flow: a retrospective, Arti. Intell. 59 (1) (1993) 81–87.
[2] J. Weber, J. Malik, Robust computation of optical flow in a multi-scale differential framework, Int. J. Comput. Vision 14 (1995) 67–81.
[3] I.K. Sethi, R. Jain, Finding trajectories of feature points in a monocular image sequence, IEEE Trans. Pattern Anal. Mach. Intell. 9 (1) (1987) 56–73.
[4] G.L. Scott, H.C. Longuet-Higgins, An algorithm for associating the features of two images, Proc. Roy. Soc. London B 224 (1991) 21–26.
[5] L.S. Shapiro, J.M. Brady, Feature-based correspondence: an eigenvector approach, Image Vision Comput. 10 (5) (1992) 283–288.
[6] G. Sudhir, S. Banerjee, A. Zisserman, Finding point correspondences in motion sequences preserving affine structure, Proc. British Machine Vision Conf., 1993, pp. 359–368.

[7] D. Skea, L. Barrodale, R. Kuwahara, R. Poeckert, A control point matching algorithm, Pattern Recognition 26 (2) (1993) 269–276.
[8] E.D. Dickmanns, V. Graefe, Dynamic monocular machine vision, Mach. Vision Appl. 1 (1998) 223–240.
[9] E.D. Dickmanns, V. Graefe, Applications of dynamic monocular machine vision, Mach. Vision Appl. 1 (1998) 241–261.
[10] N. Ayache, O. Faugeras, Building, registering and fusing noisy visual maps, Proc. 1st. Int. Conf. on Computer Vision, 1987, pp. 73–82.
[11] D. Charnley, R. Blissett, Surface reconstruction from outdoor image sequences, Proc. Alvey Vision Conf., 1988, pp. 153–158.
[12] M. Turk, K.D. Morgenthaler, K.D. Gremban, M. Marra, A vision system for autonomous land vehicle navigation, IEEE Trans. Pattern Anal. Mach. Intell. 10 (3) (1988) 342–360.
[13] D.T. Lawson, Terrain models for an autonomous land vehicle, Proc. IEEE Conf. Robotics and Automation, 1987, pp. 1127–1130.
[14] J.Y. Zheng, S. Tsuji, Panoramic representation for route recognition by a mobile robot, Int. J. Comput. Vision 9 (1) (1992) 55–76.
[15] A.D. Morgan, E.L. Dagless, D.J. Milford, B.T. Thomas, Road edge tracking for robot road following, Proc. Alvey Vision Conf., 1988, pp. 179–184.
[16] S. Umeyama, Least-squares estimation of transformation parameters between two point patterns, IEEE Trans. Pattern Anal. Mach. Intell. 13 (1991) 376–380.
[17] O. Faugeras, Three-Dimensional Computer Vision, a Geometric Viewpoint, MIT Press, Cambridge, MA, USA. ISBN 0-262-06158-9. 1993.
[18] Y. Bar-Shalom, T.E. Fortmann, Tracking and data association, In: W.F. Armes (Ed.), Mathematics and Science in Engineering, vol. 179, Academic Press, New York, 1988.
[19] B. Rao, Data association methods for tracking systems, in: A. Blake, A. Yuille (Eds.) Active Vision, MIT Press, Cambridge, MA, USA. ISBN 0-262-02351-2. 1992, pp. 91–105.
[20] H.A. Mallot, H.H. Bülthoff, J.J. Little, S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection, Biol. Cybernet. 64 (1991) 177–185.
[21] M. Bertozzi, A. Broggi. Vision-based vehicle guidance, IEEE Comput. July (1997) 49–55.
[22] R.Y. Tsai, An efficient and accurate camera calibration technique for 3D machine vision, Proc. IEEE Computer Society Conf. Computer Vision Pattern Recognition, 1986, pp. 364–374.
[23] Y. Liu, T. S. Huang, Three dimensional motion determination from real scene images using straight line correspondences, Pattern Recognition 25 (1993) 617–639.
[24] Z. Zhang, O. Faugeras, Determining motion from 3D line segment matches: a comparative study, Image Vision Comput. 9 (1991) 10–19.
[25] T.N. Tan, K.D. Baker, G.D. Sullivan, 3D structure and motion estimation from 2D image sequences, Image Vision Comput. 11 (1993) 203–210.
[26] W. Burger, B Bhanu, Estimating 3-D Egomotion from perspective image sequences, IEEE Trans. Pattern Anal. Mach. Intell. 12 (11) (1990) 1040–1058.
[27] Y.F. Wang, N. Karandikar, K. Aggarwal, Analysis of video image sequences using point and line correspondences, Pattern Recognition 24 (1993) 1065–1084.
[28] M.V. Correira, A.C. Campilho, A.J. Padilla, Motion estimation framework, Proc. 6th Spanish Symp. on Pattern Recognition and Image Analysis, 1995, pp. 605–614.
[29] J.M. Sanchiz, F. Pla, J.A. Marchant, R. Brivot, Structure from motion techniques applied to crop field mapping, Image Vision Comput. 14 (5) (1996) 353–363.
[30] J.M. Sanchiz, J.M. Iñesta, F. Pla, A neural network-based algorithm to detect dominant points from the chain-code of a contour, Proc. 13th Int. Conf. Pattern Recognition, Vienna, Austria, vol. IV, 1996, pp. 330–334.
[31] J.M. Sanchiz, F. Pla, J.M. Iñesta, Using neural networks to detect dominant points chain-coded contours, Int. J. Pattern Recognition Arti. Intell. 15 (5) (1998) 661–676.
[32] M. Isard, A. Blake, Contour tracking by stochastic propagation of conditional density, Proc. 4th. European Conf. Computer Vision, Lecture Notes on Computer Science, vol. 1064, Springer, Berlin, 1996, pp. 343–356.
[33] V. Salari, I.K. Sethi, Feature point correspondence in the presence of occlusion, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1) (1990) 87–91.
[34] Y. Liu, T.S. Huang, O.D Faugeras, Determination of camera location from 2D to 3D line and point correspondences, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1) (1990) 28–37.
[35] P. Trahanias, E. Skordalaskis, An efficient sequential clustering method, Pattern Recognition 22 (1989) 449–453.
[36] H. W. Sorenson, Least-squares estimation: from Gauss to Kalman, in: H.W. Sorenson (Ed.), Kalman filtering: theory and application, The Institute of Electrical and Electronic Engineers, 1985, pp. 7–12.
[37] R. Matthies, T. Kanade, R. Szeliski, Kalman filter-based algorithms for estimating depth from image sequences, Int. J. Comput. Vision 3 (1989) 209–236.
[38] Z. Zhang, Strategies for tracking tokens in a cluttered scene, Proc. British Machine Vision Conf., vol. 1, 1993, pp. 205–216.
[39] J.M. Sanchiz, J.A. Marchant, F. Pla, A. Hague, Real-time visual sensing for task planning in a field navigation vehicle, Real-Time Imaging 5 (1) (1998) 55–65.
[40] J.A. Marchant, R. Brivot, Real-time tracking of plat rows using a Hough Transform, Real-Time Imaging 1 (1995) 363–371.
[41] R. Brivot, J.A. Marchant, Segmentation of plants and weeds using infrared images, Proc. IEE. Vision, Image Signal Process. 143 (2) (1996) 118–124.
[42] J.A. Marchant, R.D. Tillet, Software and transputer system design for high speed grading of agricultural produce, Mechatronics 4 (3) (1994) 281–293.

**About the Author**—JOSÉ MIGUEL SANCHIZ received a Technical Engineering degree in Electronic Equipment from the Technical University of Catalonia, Spain, in 1985, a degree in Physics (Industrial and Automatics speciality) from UNED University, Madrid, Spain, in 1993, and the Ph.D. from the University Jaume I, Castelló, Spain, in March 1997. He was Invited Scientist at the Silsoe Research Institute, UK, in 1994 and 1995, he is currently a lecturer at the Department of Compuiter Science of the University Jaume I. His main research interests are target tracking, motion analysis, feature detection and neural network applications to computer vision. Dr. Sanchiz is member of the AERFAI (Spanish Association for Pattern Recogintion and Image Analysis), of the IAPR (International Association for Pattern Recognition), and affiliate of the IEEE Computer Society.

**About the Author**—FILIBERTO PLA received the degree in Physics (Electricity, Electronics and Computer Science speciality) from the University of València, Spain, 1989, and the Ph.D. from the same Univeristy in 1993. He was Research Assistant at the IVIA (Valencian Institute of Agricultural Research), Spain, from 1990 to 1992, Research Assistant at the CEMAGREF, France, in 1991, and Research Fellow at the Silsoe Research Institute, UK, in 1993, and at the Department of Electrical and Electronic Engineering of the University of Surrey, UK, in 1996. He is currently at the University Jaume I in Spain, where he joined in 1993 as a Reader in the Department of Computer Science. His current research interests include stereo vision, motion analysis, colour image processing, featue selection, binary decision trees and neural network-based classifiers. Dr. Pla is member of the AERFAI and of the IAPR.