3D Understanding by Ground Plane Estimation Based on Virtual Camera Motion

Adrià Pérez Rovira ADRIA.PEREZ@UAB.ES
Deparamente de Ciències de la Computació
Univeristat Autònoma de Barcelona
Barcelona, Espanya

Tutor: Xavier Binefa Valls

Abstract

In this paper we present a novel method to obtain the ground plane orientation of a 3D scenario viewed from a static camera. We assume that multiple objects are moving rigidly on a ground plane observed by a fixed camera. After an automatic tracking of some object features we do a classical "Structure from Motion" (SfM) method which gives a static object viewed from a hypothetical camera called "virtual camera". This camera shares the same intrinsic parameters with the real camera but moves differently due to real object motion. Assuming that the object rotates around the plane normal (equivalent to say over the ground plane), we can infer that the virtual camera will rotate around the same vector. Therefore, we use the virtual camera positions along the sequence to estimate this rotation axis, the ground plane normal. After an accurate error evaluation of the method, we use the plane information to describe how to obtain different useful information about the objects as their real position, orientation, size or speed.

Keywords: Ground Plane, Virtual Camera, Factorization, Camera Pose, Structure from Motion

1. Introduction

A three-dimensional scene viewed from a single static camera is a very common situation. In that case, the lost of deepness perception makes quite more difficult all the possible computer vision tasks. For that reason, the different methods usually try to restrict the problems of tracking, 3D reconstruction, etc. doing some kind of assumption, implying a lost of generality in benefit of higher robustness.

The famous American psychologist J. J. Gibson noted, more than 50 years ago, that ground plane perception is of vital importance to human mobility and scene understanding (Gibson (1950)). Quoting from Gibson,"There is literally no such thing as a perception of space without the perception of a continuous background surface". His "ground theory" hypothesis suggested that the spatial character of the visual world is given not by the objects in it but by the ground and the horizon.

Therefore, one typical assumption is that the scene (or part of it) can be approximated by a plane. Known this plane, we would be able to compute the 3D position of any point lying on the plane just intersecting the projection ray of the 2D image point with the ground

plane. Knowing the real 3D position of the objects is a powerful information. With this knowledge we are able to compute several object properties as their position, velocity, size or orientation.

But the plane is not only a deepness estimator tool. For instance, if this plane orientation is known, we can reformulate the problem of tracking moving objects into a more robust formulation, taking into account that the movement of the vehicle has to be done over the ground plane, eliminating its freedom in the vertical direction. This technique is used in Yuan and Medioni (2006), where a planar-translation constraint is used to determine the object translation in a sequence viewed from a moving camera.

The information about plane orientation can also be used to achieve an accurate background subtraction. In Yuan and Medioni (2006), 4 co-planar points are selected manually and tracked along the sequence obtaining the homographies which describe the camera motion. In this paper (as in many others) the plane is also used to calibrate the intrinsic and extrinsic parameters of the camera, as detailed in Malis and Cipolla (2002) and Hartley and Zisserman (2000).

A lot of literature is written about the obtention of the ground plane orientation, but usually different points of view of the 3D scene are used to retrieve this plane as viewed in Se and Brady (2002). Basically, a stereo vision system or a moving camera is used for this purpose, but exists one marginalized approach to obtain this ground plane: The movement of the objects displacing on it. Our approach relies on this idea to obtain the plane orientation from a single point of view.

In this paper we present a method which uses the trajectory of at least 4 points of a rigid object to obtain its shape and projections with a factorization method. After that, we extract the plane orientation using the information implicit in the computed projections. Specifically we find the rotation axis which describes the movement of the different projections as explained in section (2). Lately, as detailed in section (4), we use the plane orientation to obtain 3D information of the objects moving over the plane. This information will be their real position, velocity, orientation, size and distance from the camera. This object attributes are very useful for a surveillance system able to sort all the possible targets in terms of danger in a higher level layer. This information would also be very useful in an object recognition method, where the object size and orientation can help to achieve faster and more accurate results.

To use the method detailed in this paper we need the trajectories of some features of the same object. For our experiments we have developed a robust tracking over Infra-Red images based on Comaniciu et al. (2000), Zhang et al. (2004) and Martinez et al. (2007). Therefore, we have always worked with a kind of image more robust to luminance changes but with less information than color images, since RGB have 3 channels versus 1 in Infra-Red. Both, in Infra-Red and RGB videos, it is very useful to use the well-known Kalman filter (Kalman (1960)) or another predictive model in order to reduce tracking noise and to

work with partial occlusions.

To be able to work with real distance measurements as position, size, etc. and not only with a relative scale of it, we must know the distance between the camera and the scene. This information can be easily obtained using a single telemeter which gives the distance from the camera to the central point of the plane viewed from it.

Firstly, in section (2) we introduce some preliminary concepts as the factorization algorithm, and we detail our method to obtain the plane orientation. In section (3) we study the performance of our approach using synthetic data and evaluating the obtained error for the method and, finally, in section (4) we compute different object properties using the information of the plane orientation and the distance between the camera and the scene.

2. Ground Plane Estimation

As briefed in section (1), we estimate the ground plane based on a factorization algorithm (Tomasi and Kanade (1992), Hartley and Zisserman (2000), Torresani and Hertzmann (2004)) which computes both the three-dimensional shape of the tracked features and (for each frame) the projections transforming those 3D points into the bi-dimensional points observed by the camera.

In first place we introduce some preliminary mathematical and geometric concepts (section 2.1) and lately, in section (2.2), we describe our novel method to obtain the ground plane orientation.

2.1 Preliminaries

If we denote 3D points as $X = (X, Y, Z)^T$, and the image 2D points as $x = (u, v)^T$, we can write the projection equation as:

$$x = R \cdot X + t \tag{1}$$

where R is a 2×3 matrix and t a 2D-vector. But, as common in such minimization problems, the translation vector t can be eliminated in advance by choosing the centroid of the points as the origin of the coordinate system. Then, equation (1) is simplified into:

$$x = R \cdot X \tag{2}$$

The goal of the factorization method is to find a reconstruction to minimize geometric error in image coordinate measurements. In other words, the factorization algorithm computes the projections $\{R^f\}$ and the 3D points $\{X_p\}$ (where f is the frame index and p the point index), such that the distance between the estimated image points $\hat{x}_p^f = R^f \cdot X_p$ and measured image points x_p^i is minimized.

$$\min_{R^f, X_p} \sum_{fp} \| x_p^f - \hat{x}_p^f \|^2 \quad = \quad \min_{R^f, X_p} \sum_{fp} \| x_p^f - (R^f X_p) \|^2 \tag{3}$$

Now, the minimization problem has a very simple form when written as a matrix. A matrix W is built with the bi-dimensional position of the tracked points at each frame (taking into account that at least 4 points are needed) sorted in the top rows the u (horizontal tracked features) and in the lower ones the v (the vertical coordinate). Each column of W contains all the observations for a single point (p), while each row contains all the observed u-coordinates (at the top) or v-coordinates for a single frame (f).

$$W = \begin{bmatrix} u_{11} & \dots & u_{1p} \\ \vdots & & \vdots \\ u_{f1} & \dots & u_{fp} \\ v_{11} & \dots & v_{1p} \\ \vdots & & \vdots \\ v_{f1} & \dots & v_{fp} \end{bmatrix} \tag{4}$$

Suppose that the camera orientation at frame f is represented by orthonoraml vectors i_f, j_f and k_f, where i_f corresponds to the *u-axis*, j_f to the *v-axis* and k_f is the cross product of both. The vectors i_f and j_f are collected over F frames into a motion matrix $\hat{R} \in R^{2F \times 3}$ such that

$$\hat{R} = \begin{bmatrix} i_1^T \\ \vdots \\ i_F^T \\ j_1^T \\ \vdots \\ j_F^T \end{bmatrix} \tag{5}$$

Let s_p be the location of the feature p in a fixed world coordinate system, whose origin is set at the mass center of all the P feature points. These 3D-vectors are collected into a shape matrix $\hat{S} \in R^{3 \times P}$ as follows

$$\hat{S} = [s_1 \dots s_P] \tag{6}$$

The factorization algorithm decomposes the bi-dimensional tracked features sorted in W into the projections collected in $\hat{R}$ which transform the 3D static object ($\hat{S}$) into the same 2D image points ($\hat{x}^f$) as observed by the static camera at each frame.

$$W = \hat{R} \cdot \hat{S} \tag{7}$$

Equation (7) has not a unique solution, since an arbitrary invertible 3x3 matrix M may be inserted in the decomposition as follows:

$$W = \hat{R} \cdot \hat{S} = R \cdot M \cdot M^{-1} \cdot S \tag{8}$$

and consequently,

$$R = \hat{R} \cdot M \tag{9}$$
$$S = M^{-1} \cdot \hat{S} \tag{10}$$

Using this property we are able to update the affine reconstruction to a metric reconstruction by adding geometric restrictions while the equality is still true. In our case we force the projection axes for each frame to be orthonormals (with norm 1 and linearly independent). As sketched in Tomasi and Kanade (1992) and detailed in Morita and Kanade (1997). We apply this restriction using the normalization constraints as follows,

$$i_f^T i_f = j_f^T j_f = 1 \quad and \quad i_f^T j_f = 0, \tag{11}$$

obtaining the system of $3F$ overdeterminated equations, such that

$$\begin{aligned} \hat{i}_f^T L \hat{i}_f &= 1 \\ \hat{j}_f^T L \hat{j}_f &= 1 \\ \hat{i}_f^T L \hat{j}_f &= 0 \end{aligned} \tag{12}$$

where $L \in R^{3\times3}$ is a symmetric matrix defined as

$$L = AA^T \tag{13}$$

and $\hat{i}_f$ and $\hat{j}_f$ are the rows of $\hat{R}$. By denoting $i_f^T = [i_{f1}, i_{f2}, i_{f3}]$, $j_f^T = [j_{f1}, j_{f2}, j_{f3}]$, and

$$L = \begin{bmatrix} l_1 & l_2 & l_3 \\ l_2 & l_4 & l_5 \\ l_3 & l_5 & l_6 \end{bmatrix} \tag{14}$$

The system (12) can be rewritten as

$$Gl = c, \tag{15}$$

where $G \in R^{3F\times6}$, $l \in R^6$ and $c \in R^{3F}$ are defined by

$$G = \begin{bmatrix} g^T(i_1, i_1) \\ \vdots \\ g^T(i_F, i_F) \\ g^T(j_1, j_1) \\ \vdots \\ g^T(j_F, j_F) \\ g^T(i_1, j_1) \\ \vdots \\ g^T(i_F, j_F) \end{bmatrix}, \quad l = \begin{bmatrix} l_1 \\ \vdots \\ l_6 \end{bmatrix} \quad and \quad c = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{16}$$

and

$$g^T(a_f, b_f) = [\quad a_{f1}b_{f1} \quad a_{f1}b_{f2} + a_{f2}b_{f1} \quad a_{f1}b_{f3} + a_{f3}b_{f1} \\ a_{f2}b_{f2} \quad a_{f2}b_{f3} + a_{f3}b_{f2} \quad a_{f3}b_{f3} \quad] \tag{17}$$

The simplest solution of the system is given by the pseudoinverse method, such that

$$l = (G^T G)^{-1} G^T c \tag{18}$$

With the vector l we build the symmetric matrix L, whose eigendecomposition gives A. If we decompose L as $L = U\Sigma V$, we obtain A as

$$A = U\Sigma^{\frac{1}{2}} \tag{19}$$

As a result, using the properties deduced from equation (8) the shape S and the projections R are derived according $R = \hat{R}A$ and $S = A^{-1}\hat{S}$.

2.2 Ground Plane Normal Estimation

The estimated projection for each frame is denoted by R_f, and it is a 2×3 matrix composed by two 3d-vectors as rows (i_f and j_f in figure (1)) which gives us the 2D point by means of formula (2). We can also understand those projections as a "virtual camera" moving thorough the scene. This virtual camera shares the same intrinsic parameters with the real one, but moves differently due to object motion. This motion is estimated by the factorization algorithm trying to minimize equation (3). Doing so, in absence of noise, there is an equality between the image points (x^f) and the ones obtained projecting the shape S through R_f.

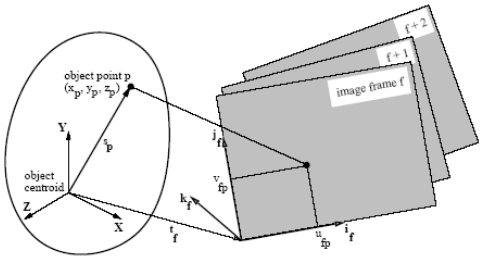

Figure 1: The system of reference in the factorization step.

We transform each frame projection R_f into a full camera pose (3-axes of orientation and a 3D position). We reuse the i_f and j_f vectors gived by R_f as the right and up camera axis respectively (where f is the frame index) and we obtain the forward axis as the normalized cross product of both.

$$C_f^i = i_f \tag{20}$$
$$C_f^j = j_f \tag{21}$$
$$C_f^k = C_f^i \times C_f^j \tag{22}$$

And all 3 axes expressed together as a matrix:

$$C_f = \begin{bmatrix} C_f^i & C_f^j & C_f^k \end{bmatrix} \tag{23}$$

Then, we have to compute the 3D-vector T_f from the center of world coordinate system to the camera positon. This translation vector is the inverted forward axis (C_f^k) of the camera multiplied by the distance between the camera and the world coordinate origin d. This vector can also be understood as the camera position in the world coordinate system.

$$T_f = -C_f^k \cdot d \tag{24}$$

Once we have both the camera axes (C) and its position (T_f), we are able to transform any 3D point from world into camera coordinate system as follows:

$$X^c = C_f \cdot X^w + T_f \tag{25}$$

where the superscripts c (w) refers to camera (world) coordinate system.

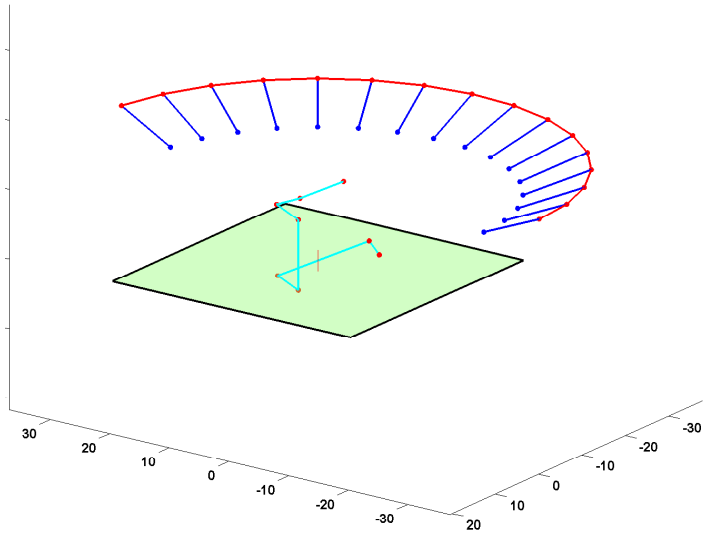

Figure 2: A fixed shape and the virtual camera rotating 180 degrees around the plane normal (different positions and the camera center path are shown), always looking at the center of the object.

After obtaining the virtual camera trajectory, we can use the properties deduced of its motion to estimate the ground plane of the scene. First of all, we know that the virtual camera moves just in the opposite way as the real object does. While the reality consists on a moving object viewed from a static camera, the factorization algorithm output is a static and rigid object viewed from a moving camera. Then, as the virtual camera projection gives us the same bi-dimensional points as the real camera does (in absence of noise), the object movement has to be implicit in the virtual camera to satisfy the equation (3).

For instance, if a real object is rotating 30 degrees over the plane in clockwise, the virtual camera does the rotation in anti-clockwise for 30 degrees while the object shape stills invariant.

Secondly, we know that the tracked vehicles rotate always over the plane (using the plane normal as the rotation axis). So, the virtual camera uses the same axis to rotate around the static object satisfying equation (7), as shown in figure (2).

We are able to decompose any rotation 3×3 matrix obtained between two virtual camera poses (with indices f and g) into a rotational axis $\omega_{f,g}$ and the angle of rotation $\theta_{f,g}$ with the formulas derived from the Rodrigues rotation theorem as follows,

$$\theta_{f,g} = arcos\left(\frac{trace(Q_{f,g})-1}{2}\right) \tag{26}$$

$$\omega_{f,g} = \frac{1}{2\sin\theta_{f,g}}\begin{bmatrix} R(3,2) - R(2,3) \\ R(1,3) - R(3,1) \\ R(2,1) - R(1,2) \end{bmatrix} \tag{27}$$

where $Q_{f,g}$ is the rotation matrix between the virtual camera in frames f and g, computed as:

$$Q_{f,g} = C_g \cdot C_f^{-1} \tag{28}$$

At this point is important to notice that the rotational axes obtained with this method can be expressed as a 3D-vector v or $-v$, both with same direction but in opposite sense. To cope with this uncertainty, we firstly compute the mean vector v_m of all the rotational axes obtained between all the virtual camera pose pairs $\{\omega_{f,g}\}$ weighted with their rotational angle $(\theta_{f,g})$. Then, we compute the dot product between each $\omega_{f,g}$ and v_m. If the result is a negative value, we invert the vector $(\omega_{f,g} = -\omega_{f,g})$. Doing this we force all the vectors $\{\omega_{f,g}\}$ to be oriented in the same sense. As seen in figure (3)

Finally we estimate the rotational axis of all the scene as the mean vector of all the corrected $\omega_{f,g}$. Being this one the ground plane normal.

Once we have computed the plane, we do a base change for an easier manipulation of the three-dimensional data. With this purpose we build the matrix M (in equation (8)) which transforms the arbitrary R^3 space obtained from the factorization into a plane adapted one,

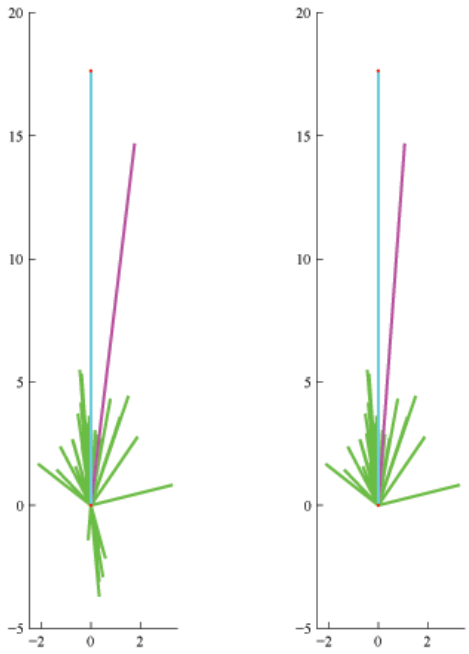

Figure 3: Left image shows all the rotation axes (in green) computed for a noisy synthetic data set of 50 virtual cameras. In the right image we have corrected the sense of the vectors and the estimated ground plane (in magenta) appears closer to the real rotation axis (in cyan).

where the normal of the estimated plane corresponds with the *z-axis*. The *x-axis* and *y-axis* will be two arbitrary perpendicular vectors between them and with the *z-axis*, forming all three an orthonormal base. So *z-axis* is the plane normal and *x-axis* and *y-axis* are two vectors in the ground plane.

Specifically, we build the M matrix as:

$$V_z = \frac{v_m}{\| v_m \|} \tag{29}$$

$$V_y = \frac{(-V_z(y), Vz(x), 0)^T}{\| (-V_z(y), Vz(x), 0)^T) \|} \tag{30}$$

$$V_x = V_y \times V_z \tag{31}$$

$$M = (\overrightarrow{V_x}, \overrightarrow{V_y}, \overrightarrow{V_z}) \tag{32}$$

As M is a orthonormal matrix, we know that $M^{-1} = M^T$, thus, we can compute the new shape (S_n) and its projections (R_n) as:

$$S_n = M \cdot S \tag{33}$$
$$R_n = R \cdot M^T \tag{34}$$

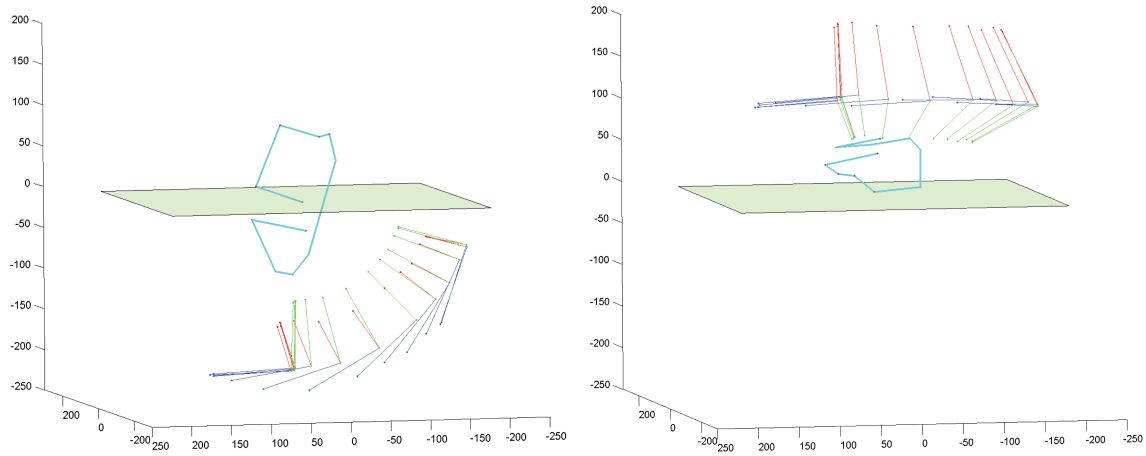

Figure 4: Left image shows the shape S and the virtual cameras R in the arbitrary space given by the factorization algorithm, in the right image S_n and R_n are aligned to our adapted space where the ground normal corresponds with the *z-axis*.

At this point we are able to compute the pitch and roll angles of the camera using as reference the ground plane estimated. To obtain the pitch angle we project (for each virtual camera pose) the ground normal vector, $(0, 0, 1)$ in our transformed space, to the plane defined by the up (C_f^j) and the forward (C_f^k) vectors. We use this $2D$ vector to compute the pitch angle, just doing the arccosine of the dot product between the computed vector and $(1, 0)$. For the roll angle we project the $(0, 0, 1)$ vector into the plane defined by the right and up vectors (C_f^i and C_f^j) and do the arcsine of the dot product between this vector and $(1, 0)$.

In a perfect scenario, without noise, the pitch and roll angle will be the same for each frame, but to cope with real data, with the noise added in tracking and factorization steps, we compute the mean roll and pitch angle of all the virtual cameras.

Using the previous camera pitch and roll angles we can build the 3-vector axes of the "scene camera" (C) using the rotation matrices composition theorem as follows:

$$R = R_{roll} \cdot R_{pitch} \tag{35}$$
$$R = \begin{pmatrix} \cos(\theta_{roll}) & 0 & -\sin(\theta_{roll}) \\ 0 & 1 & 0 \\ \sin(\theta_{roll}) & 0 & \cos(\theta_{roll}) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_{pitch}) & \sin(\theta_{pitch}) \\ 0 & -\sin(\theta_{pitch}) & \cos(\theta_{pitch}) \end{pmatrix} \tag{36}$$

$$C = R \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{37}$$

Finally, if we know the distance (d) between the camera and the central part of the image (the world center), the scene camera center (c) would be situated in the opposite direction of C^k at distance d:

$$c = -d \cdot C^k \tag{38}$$

3. Evaluation Study

In this section, the performance of the proposed approach is studied by using synthetic data. Considering different values of noise in the virtual camera positions, from 0 to 1, where 1 is a gaussian noise level of 100% of the data. Different rotation angles are also evaluated, from 5 degrees to 360.

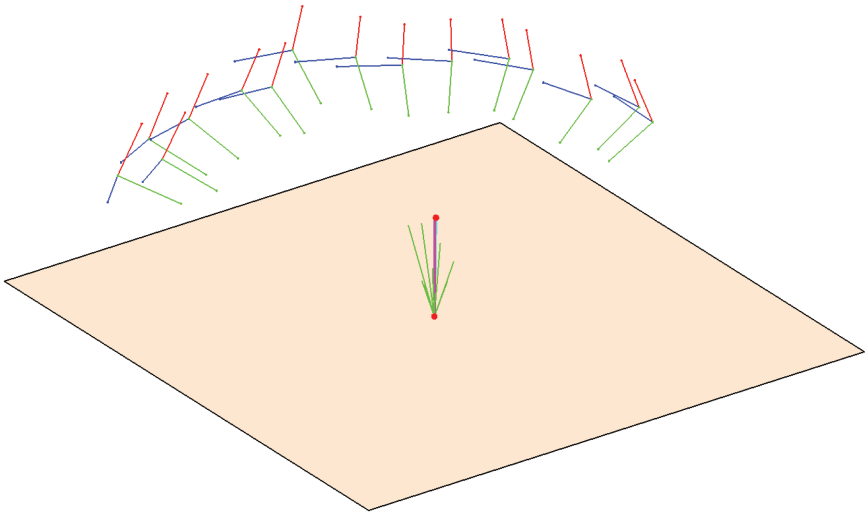

Figure 5: Synthetic data for 15 virtual cameras with a 0.1 noise level, rotating 135 degrees. The computed rotation axis (magenta) have an error of 2.38 degrees about the real axis (cyan).

The synthetic scenarios are generated with 15 virtual cameras and the evaluation is done for 20 iterations evaluating the mean, the maximum, and the error variance. The error value expressed in all our tests is the angle (in degrees) between the real rotation axis and the computed one. One example of this synthetic scenarios can be seen in figure (5).

In figure (6) we can see the error surface with the rotation angle in the *x-axis*, the noise level in the *y-axis* and the error in the *z-axis*. The opaque black zone shows the conditions where the error is lower than 5 degrees and the translucent one is for the zone where the error is under 10 degrees.

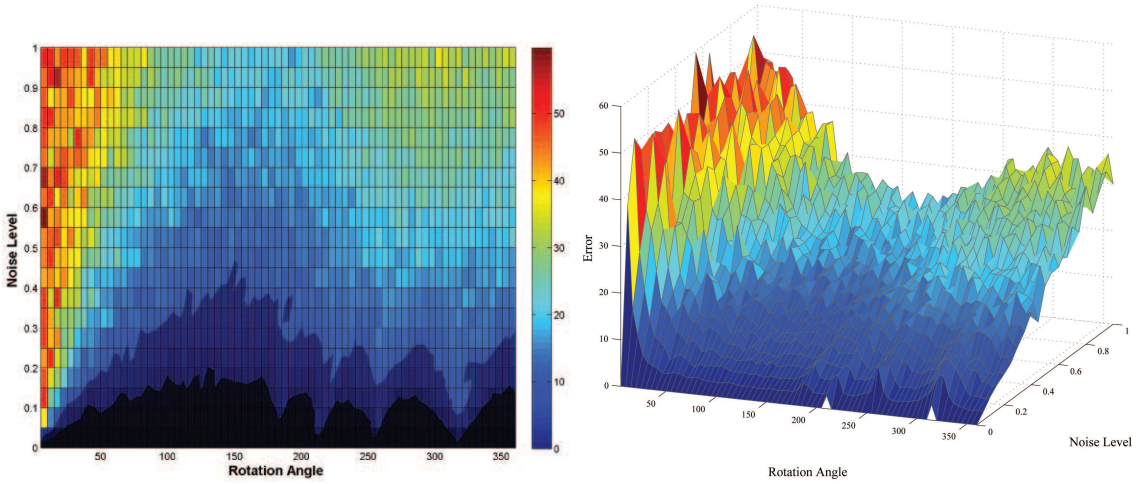

Figure 6: Error Mean Surface. At left image a dark plane cuts the surface showing the zone where the error is lower than 5 degrees and another translucent plane shows the zone where the error is lower than 10 degrees. At the right the same surface is plotted in an oblique point of view.

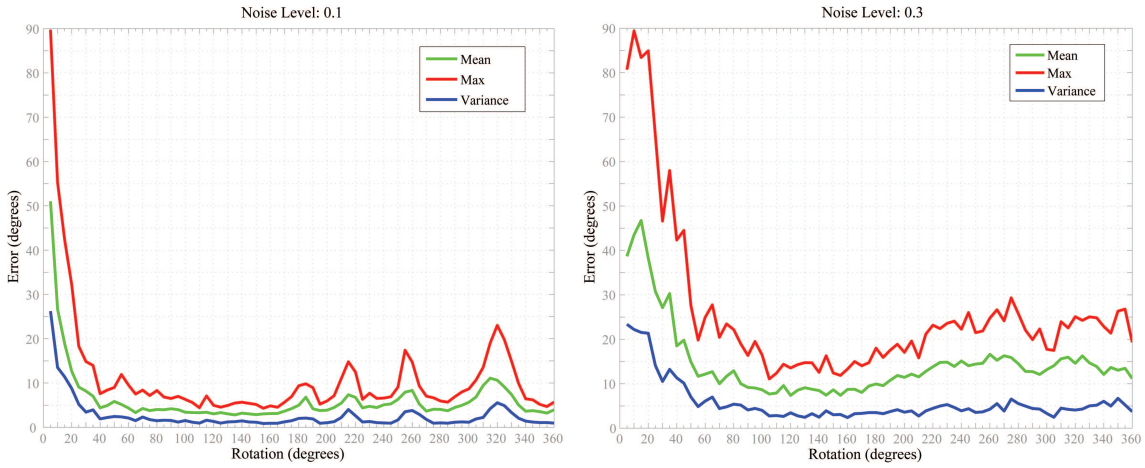

Figure 7: Error value (in degrees) for a synthetic data set with a 0.1 noise level (at the left) and for another synthetic data set with a noise level of 0.3 (at the right).

Seeing the error surface in figure (6) we can guarantee accurate results with a 0.1 noise level. In figure (7) we see the surface section for this noise level for all the possible rotations. Studying the error evolution of this figure, we can observe our method is best working with an object rotation between 40 and 170 degrees. Something really feasible in real conditions.

Although only in small presence of noise we are able to achieve accurate results with an error lower than 5 degrees, with 3 times more noise, we obtain errors lower than 10 degrees in a wide rotation range, between 70 and 170 degrees.

It is curious that having a rotation angle over 170 degrees the method looses some accuracy. This can be explained for the uncertainty produced by 2 virtual camera positions in opposite positions. When this particular case appears, the rotation axis between both positions is an ambiguity, as it could be any one. For illustrate this case, imagine a traveler who wants to move from the South Pole to the North Pole, which one is the shortest way? Does not exist only one, any straight path over the surface would be a valid solution.

4. 3D Object Motion and Pose Estimation

At this point we have already found the scene camera pose (orientation and position) in correspondence with the ground plane coordinate system. Knowing all the intrinsic parameters of the camera, we are able to transform each bi-dimension point $x(u, v)^T$ of a frame into a 3D real position $X(x, y, z)^T$ over the estimated plane.

If we don't know the height of the point (the distance to the ground), it is impossible to do this transformation due the ambiguity of its deep. But if the point is on the surface, or close to it, we can do this transformation easily. We trace a ray from the camera focal center to the scene through the image point x in our image plane. Then, we intersect this ray with the ground plane previously estimated and we obtain the 3D point in the scene.

With this capability we are able to compute the relative position of the objects and their orientation. But if we want to do a step beyond as to compute their absolute position or their real size and velocity, we have to know the distance between the camera and the plane. This information can be provided by a single telemeter. In this section we describe how to calculate those object properties knowing only the plane orientation and the distance to it.

4.1 Object Orientation

The first information we compute for each object is its orientation at each frame. More concretely, we compute the 3×3 rotation matrix which rotates the constant shape S_n into the oriented object viewed from each virtual camera. Being the same structure for all the sequence, but in an independent orientation for each frame.

As we know that the rotation have to be done around the plane normal, one simple idea to avoid the noise in the *z-axis* of the virtual cameras is project al theme over the computed

plane. Once we have projected the scene camera C into the ground plane, as we have done with all the virtual cameras $\{C_f\}$, the rotation matrices we are looking for are the matrices which transform each projected C_f into the projected C.

4.2 Scale factor

As the factorization algorithm does not preserve the scale factor of the objects, it's mandatory to compute this scalar value to be able to estimate correctly further steps.

Taking into account that we are looking for a relationship between the 3D coordinate system and the 2D one, we look for the longer distance between 2 bi-dimensional object points of the first frame in the horizontal axis and store this distance. In the 3D world, we rotate the object shape S_n with the previously rotation matrix computed in (4.1) for the first frame, and we search for the distance in the *x-axis* between the same pair of points. The scale factor will be then the distance computed in the 3D world divided by the distance computed in the bi-dimensional projection.

4.3 3D Trajectory

Another important information which can be computed with the ground plane orientation is the 3D trajectory of the moving objects. This trajectory is the object position over the estimated plane at each frame.

To do this, we firstly search the lower point (in the *z-axis*) of the object shape (S_n) and we store this scalar value called z_{min}. Doing this, we assume that this 3D point is on the ground, also we are not able to be sure about it. Luckily, this assumption is not critical due the distance of this point to the ground is usually small enough to produce a nearly inappreciable error. Then, we project the vector between $(0,0,0)^T$ and $(0,0,z_{min})^T$ into the image plane and we obtain and store this bi-dimensional vector (v_z).

Now, we compute the mass center of the bi-dimensional tracked points at each frame and subtract to it the vector v_z. Doing this, we are computing the mass center of the tracked features and moving it down until reach the ground. If we project this point into the 3D estimated ground plane, the resulting 3D point will be the mass center of the ponits projected vertically into the plane. Thus, we have the position of the object over the plane at each frame.

In figure (8) we can see this trajectory for an object, plotted over the first frame of the sequence (at the right image) and plotted over our estimated plane, while in figure (9) we can see the bus trajectory registered over an aerial terrain image extracted from Google Earth of the real scenario where the sequence was filmed.

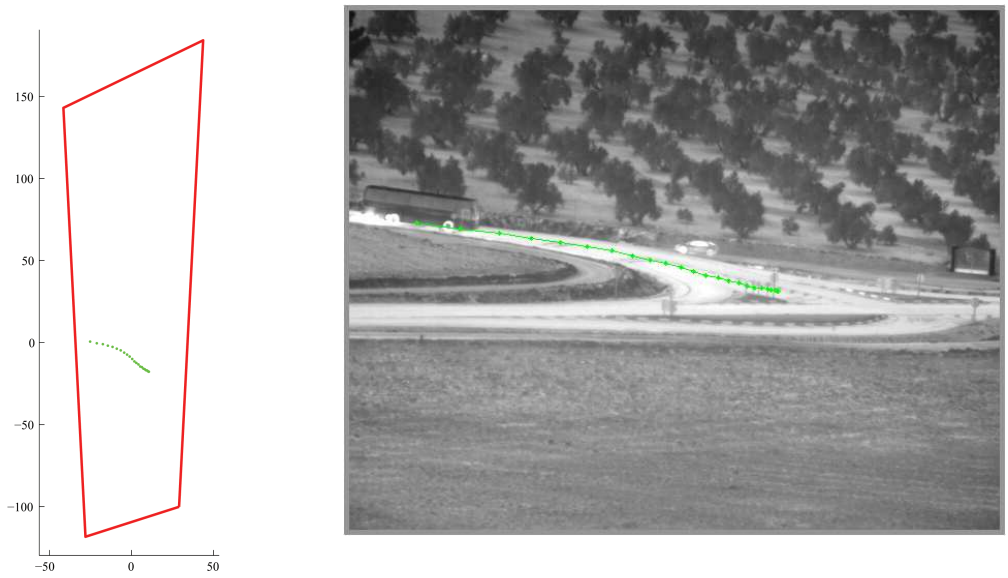

Figure 8: Left image shows the vehicle trajectory computed over the plane (with an estimated 12 degrees inclination) for an object tracked during 25 frames (in a sequence sub-sampled at 2 fps). Right image shows this trajectory projected over the first frame.

Figure 9: The estimated plane (in red), the camera axes and the trajectory described by a vehicle registered over an aerial images of the real terrain (obtained from Google Earth) in two different points of view.

4.4 Getting the Object Bounding Box

With all the information computed until the moment (the scaled shape, the object orientation along the video and the its trajectory over the plane), we are able to fit a bounding box to the object. As we only have structural information of the tracked points, we can not guarantee the correctness about the bounding box size. But if the tracked points are close to the object contours the bounding box will be just a little smaller than the real object. Although this incorrectness, this information can be really useful for an hypothetical classifier able to distinguish between a truck or a car only knowing the bounding box size. Obviously those bounding boxes can be used for an augmented reality environment, with the size, position and orientation information implicit over the video as seen in figure (10).

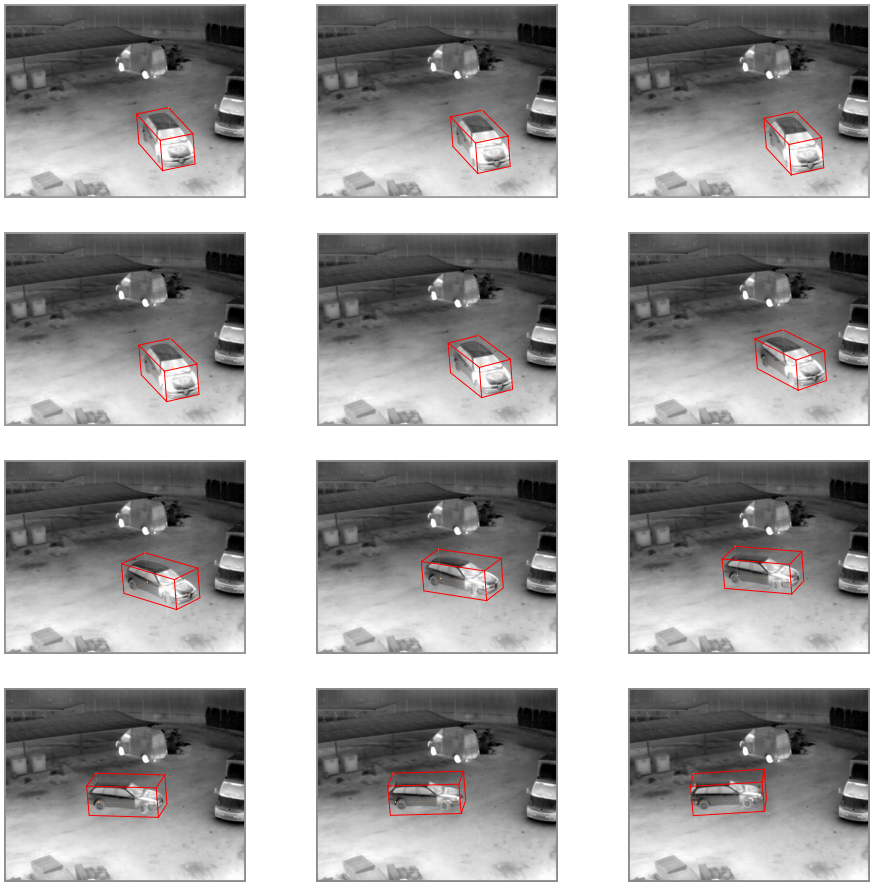

Figure 10: Video sequence of a moving vehicle with its bounding box oriented and placed at each step over the frames for an hypothetic augmented reality environment. In this sequence only those 12 frames are used during all the process with a vehicle rotation of 75 degrees.

As we are working with rigid vehicles oriented in the same direction as their trajectory does, we can use the known trajectory of the object at the first frame to obtain the axes of the bounding box, while the vertical component will be the *z-axis* of our three-dimensional space as shown in figure (11).

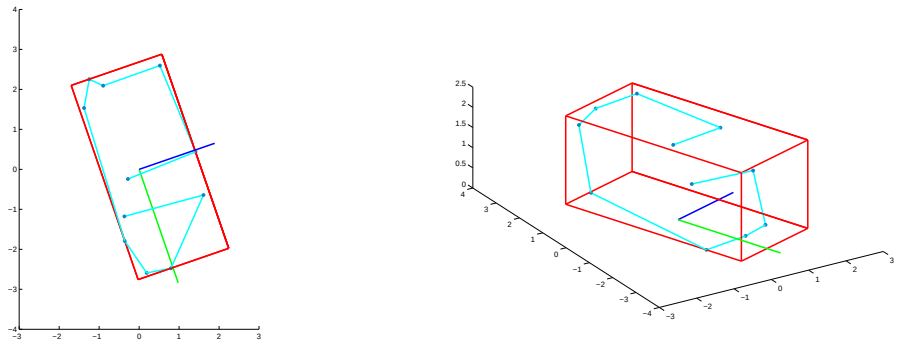

Figure 11: 2 different perspectives of the bounding box fitted to an object shape. The green line on the floor is the estimated trajectory direction, and the blue one is its perpendicular. Both are used as axes of the bounding box.

To obtain this fitted bounding box, we project each 3D point of the object shape (rotated as it is viewed in the first frame using the matrix computed in (4.1)) into its trajectory vector, obtaining the forward and the backward points. To obtain the lateral extremes, we project all the 3D points into the perpendicular trajectory vector (computed as the cross product between the trajectory vector and the ground normal, to be sure it is lying on the ground too) and store the maximum distance at both sides. Finally, the vertical component of the bounding box is computed projecting all the points into the vertical *z-axis*.

Finally, as we know the position of all the box corners, we are able to detect which lines remain occluded viewed from our scene camera. This is not a feature at all, but makes us able to plot only the visible lines as it is done in figure(10).

4.5 Object Speed

Another information which can be very useful in some application is the real speed of the object during the filmed sequence at each frame f. To compute this velocity and speed we use the object position and the frame ratio as follows:

$$Velocity_f = (X_f - X_{f-1}) \cdot fps \tag{39}$$

$$Speed_f = \| Velocity_f \| \tag{40}$$

where fps are the frames per second of the video.

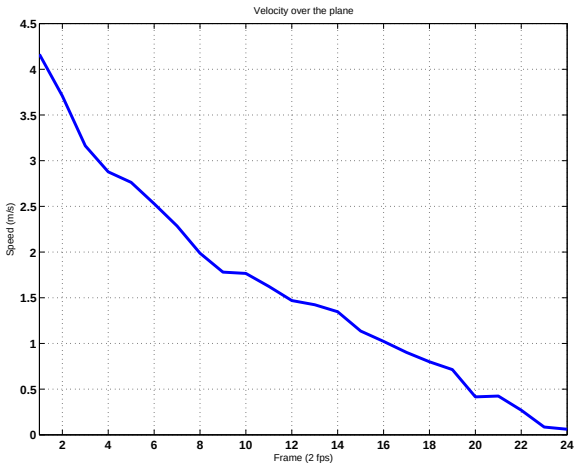

Figure 12: Real speed (in m/sec) estimated for the vehicle while it is stopping due a traffic sign.

Just as an example, in figure (12) we can see the decreasing speed of the bus, seen in figure (8), due the stop traffic sign.

5. Conclusions and Future Work

We have presented a new approach to obtain the ground plane of a scene viewed from a static camera. From the specifications of the problem, with a fixed camera, it is important to notice that the plane have to be estimated once. After that, we can re-estimate it using as many objects as desired, obtaining a very robust plane approximation, with the only restriction that all the objects tracked must move over the same plane. Then, the plane orientation is used to obtain very useful information of the objects as we are able to work with real distance magnitudes in an euclidian 3D space. Specifically we obtain their real position, distance to the camera, size, speed and orientation.

In a first and simple application of this method we can build a robust system able to distinguish the importance of the objects in a military environment. A big and fast vehicle moving toward the camera will be potentially more dangerous than a small one oriented in an opposite direction. We are also able to detect vehicles moving grouped in an specific formation too.

Another important application of the information extracted for the objects will be a robust object recognition algorithm. With the size information and the object orientation we are able to compare the objects viewed from the static camera with a data base of ve-

hicles as we know the orientation of both. This capability to compare 2 objects oriented in the same direction would make the recognition method faster and more robust than the conventional ones, where the unknown orientation of the objects force it to compare each observed object with all the vehicles stored in the data base in several different orientations. Usually, an efficient object recognition method compares each object with all the possible points of view stored. For instance, in Opelt et al. (2006), each object detected is compared with 18 possible poses rotated about vertical and horizontal axis. Consequently, we can develop an algorithm 18 times faster without loosing any accuracy.

In a more theoretical way, we can use the estimated plane information to perform adapted trackings, more accurate homography estimations, better motion segmentation, etc. achieving more accurate results as we can restrict the hypothetical object movement into a plane, eliminating the possibility of the objects to move freely in the vertical direction.

It is important to notice the low computational cost of the algorithm, having only 2 steps with significant computational cost. The first one is the factorization step, where the singular value decomposition requires $14FP^2 + 11P^3/3$ computations for a $2F \times P$ measurement matrix with $2F \geq P$. Something easily affordable with current computer hardware. The other step which can be critical is the computation of the rotation axes between all the possible virtual camera pairs. The computational cost to obtain a single axis is quite low, but if we have a lot of virtual cameras the number of possible pairs increases exponentially. But we can use no more than a dozen virtual cameras as shown in figure (10), to achieve accurate results. So, for instance, we can use the tracking points of 1 out of 10 frames. For the showed sequence of figure (10) the whole process, including the information extraction described in section (4) take 0.85 seconds executed in Matlab, a computational cost far more than acceptable.

In the experimental tests we have seen that small tracking errors can be magnified in the factorization step resulting on noise in the virtual camera movement. One possible solution to this problem is to track points over two (or more) different object faces. Unfortunately, this multi-face tracking is not always possible. In those cases we should track more points during more frames to obtain similar accuracy in the factorization algorithm step. So we can conclude our method is robust (with an object rotation angle big enough) in appropriate circumstances, but high dependant of the tracking and its accuracy.

Another important requirement to achieve accurate results is the amount of vehicle rotation. As bigger the noise is, a bigger angle of rotation is needed to estimate correctly the plane orientation. As showed in section (3) a minimum object rotation of 40 degrees will be enough to achieve acceptable results in sequences with a low noise level. Obviously, as more rotation we have, more accuracy is guaranteed. In real sequences is not usual to have rotations over 100-120 degrees because the own object would occlude some of the tracked points. But as seen in figure (7), rotations over this value give results equal or less accurate. This inaccuracy is explained with the idea as 2 opposed points (with a rotation of 180 degrees) can have any possible rotation axe. So having a rotation angle lower than 150 is

an advantage for us.

Although in this paper we present a static camera (with constant position and orientation), we have tested this method for a pivoting camera (with constant position but allowed to rotate). In order to cope with this camera rotation we just need to compute the homography between frames and use it to transform a 2D pixel position of any frame into the first frame coordinate system, used along the sequence as reference frame. The method used for obtaining this homography is a pyramidal optical-flow based on Trucco and Verri (1998) and Irani et al. (1995).

There are several possible directions for the future research. The most useful and challenging would be the development of an iterative process using the plane approximation for a better tracking or a Structure from Motion method done recursively with the planar-moving constraint. This method would make our algorithm more robust at the same time we obtain a more accurate tracking.

References

D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR '00: Proceedings of the 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 142–151, 2000.

J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. 2000. ISBN 0-521-62304-9.

M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 605, 1995. ISBN 0-8186-7042-8.

R. Kalman. A new approach to linear filtering and prediction problems. *Transaction of the ASME – Journal of Basic Engineering*, 24, 1960.

E. Malis and R. Cipolla. Camera self-calibration from unknown planar structures enforcing the multiview constraints between collineations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1268–1272, 2002. ISSN 0162-8828.

B. Martinez, A. Perez, L. Ferraz, and X. Binefa. Structure restriction for tracking through multiple views and occlusions. *IbPRIA '07: Iberian Conference on Pattern Recognition and Image Analysis*, 1:121–128, 2007.

T. Morita and T. Kanade. A sequential factorization method for recovering shape and motion from image streams. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(8):858–867, 1997. ISSN 0162-8828.

A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3–10, 2006. ISBN 0-7695-2597-0.

S. Se and M. Brady. Ground plane estimation, error analysis and applications. *Robotics and Autonomous Systems*, 39(2):59–71, 2002.

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9(2), 1992.

L. Torresani and A. Hertzmann. Automatic non-rigid 3d modeling from video. In *ECCV (2)*, pages 299–312, 2004.

E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. 1998. ISBN 0132611082.

C. Yuan and G. Medioni. 3d reconstruction of background and objects moving on ground plane viewed from a moving camera. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2261–2268, 2006. ISBN 0-7695-2597-0.

H. Zhang, Z. Huang, W. Huang, and L. Li. Kernel-based method for tracking objects with rotation and translation. In *ICPR '04: Proceedings of the Pattern Recognition*, pages 728–731, 2004. ISBN 0-7695-2128-2.