Combining Motion from Texture and Lines for Visual Navigation

Konstantinos Bitsakos, Li Yi and Cornelia Fermüller
Center for Automation Research, University of Maryland, College Park, MD 20742
kbits@cs.umd.edu, liyi@umiacs.umd.edu and fer@cfar.umd.edu

***Abstract*— Two novel methods for computing 3D structure information from video for a piecewise planar scene are presented. The first method is based on a new line constraint, which clearly separates the estimation of distance from the estimation of slant. The second method exploits the concepts of phase correlation to compute from the change of image frequencies of a textured plane, distance and slant information. The two different estimates together with structure estimates from classical image motion are combined and integrated over time using an extended Kalman filter. The estimation of the scene structure is demonstrated experimentally in a motion control algorithm that allows the robot to move along a corridor. We demonstrate the efficacy of each individual method and their combination and show that the method allows for visual navigation in textured as well as un-textured environments.**

I. Introduction

Changes (over multiple frames) on the boundaries and the texture provide complimentary information about the shape and the 3D position of an object. Thus, combining methods based on boundary extraction with ones on textured regions results in more robust and accurate estimation. Especially, for relatively simple environments, such as corridors, it is often the case that only one type of cue will be present and thus only one type of method will provide reliable measurements. Furthermore, in such environments the predominant shape of objects is planar and the object boundaries are usually lines.

Motivated by the above observations, this paper proposes two methods to estimate the position of planar objects; the first considers the change of the texture and the second the change of lines. More specifically, the main contributions of the paper are:

- We present a novel image line constraint for estimating the 3D orientation of planes (Sec. III).
- We describe a novel technique estimating shape from change of texture for planar objects based on harmonic analysis (Sec. IV).
- We present experimental results on how accurate the two methods perform in real indoor environments. The integration of the two methods with the odometry readings from the robot's wheels using an extended Kalman filter, outperforms the results obtained by each method in isolation (Sec. VI).
- We experimentally show that the proposed method allows for navigation in environments where little texture is present using a simple motion control policy (Sec. VIII).

A. Related Work

The computer vision community has long studied the structure from motion (SfM) problem ([16],[13]) and recently focused on large-scale 3D reconstruction (e.g. [1]). Following the success of Simultaneous Localization and Mapping (SLAM) using range (especially laser) sensors ([28]), the robotics community has migrated the existing methods to work with data from cameras. Usually, the environment is represented with a set of image feature points, whose pose is tracked over multiple frames ([10]). Usually, image features are more informative than range data, but the estimation of their 3D position is much less accurate. Straight lines are common in man-made environments and are arguably more reliable features than points, thus they have been used before in structure from motion ([4], [30]) and SLAM ([25]). Our method is about computing 3D structure information in a simplified SfM situation, but very robustly. We use a formulation of line constraints that separates slant from distance estimation. Thus, it is different from the ones classically used in SfM.

On the other end of the spectrum there are methods belonging to the *mapless visual navigation* category ([11]), where no prior knowledge about the environment is assumed and no spatial representation of it, is created. Most of that work is inspired by biological systems. A survey of such methods implementing the centering behavior can be found in [27]. More specifically, systems capable of avoiding walls and navigating in indoors environments using direct flow-based visual information obtained from a single wide-FOV camera facing forwards ([8], [7], [12]), multiple cameras facing sideways ([23], [2]) or panoramic cameras ([3]), have been implemented. Our approach is also different from the aforementioned, because we first estimate an intermediate state of the environment (in terms of surface normals) and we use this for navigation.

The general method for estimating the stretch and shift of a signal using the log of the magnitude of the Fourier transform, known as *Cepstral analysis*, was first introduced by Bogert et al. [5] and was made widely known by Oppenheim and Schafer [22]. It is commonly used in speech processing [19] to separate different parts of the speech signal.

Frequency based techniques exploiting the phase shift theorem have been used in computer vision for image registration (in conjunction with the log-polar transform of an image), e.g. [26], [18], [15] and optical flow computation ([14]). Phase correlation, however, has not been used for

shape estimation.

II. Problem Statement and terminology

In this section we introduce some common symbols that are used in the rest of the paper and present the problem that we tackle in the following three sections. For simplicity and improved readability reasons, all the equations in Sec. III,IV and V are expressed in the camera coordinate system (where the images were acquired). In Sec. VI and VII we transfer the estimates in the robot-centric coordinate system (Fig. 5(b)). Vectors are denoted with an overhead arrow and matrices with bold letters.

We denote with $\overrightarrow{T}$, $\mathbf{R}$ the translation and rotation between two frames respectively, with $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$ a plane in the 3D world and with $\overrightarrow{n} = \frac{\overrightarrow{N}}{|\overrightarrow{N}|}$, the plane normal. Also $\overrightarrow{P} = (X, Y, Z)^T$ is a 3D point. When $\overrightarrow{P}$ belongs to $\overrightarrow{N}$ then $\overrightarrow{P} \cdot \overrightarrow{N} = 1 \Leftrightarrow \alpha X + \beta Y + \gamma Z = 1$. The image plane is assumed to lie on the plane $\mathbb{I} : Z = f$, where f is the focal length of the camera. Then, the projection of $\overrightarrow{P}$ on $\mathbb{I}$ is $\overrightarrow{p} = (x, y, f)^T = \frac{f}{Z}(X, Y, Z)^T$. The inverse depth at $\overrightarrow{P}$ amounts to

$$\frac{1}{Z} = \alpha\frac{x}{f} + \beta\frac{y}{f} + \gamma \tag{1}$$

Given the translation and rotation of the camera between two images we seek to *estimate the plane parameters* $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$.

III. Orientation and Distance from lines

We describe a constraint for recovering the orientation of a world plane from image lines. The constraint can be used in two ways: first as a multiple view constraint, where we use the images of a single line in 3D in two views [17]; second as a single view constraint where we use the images of two parallel lines in 3D in one view.

A. Single Line in Multiple Frames

As shown in Fig. 1(a), consider two views with camera centers O_1 and O_2, which are related by a rotation $\mathbf{R}$ and a translation $\overrightarrow{T}$. A 3D line $\mathcal{L}$ lies on the plane with surface normal $\vec{n} = \frac{\vec{N}}{|N|}$. $\mathcal{L}$ is projected in the two views as l_1 and l_2. Let $\vec{l_{m1}}$ be the representation of l_1 in the first camera coordinate system as a unit vector perpendicular to the plane through $\mathcal{L}$ and O_1. Similarly, let $\vec{l_{m2}}$ be the representation of l_2 in the second camera coordinate system as a unit vector perpendicular to the plane through $\mathcal{L}$ and O_2. The two planes perpendicular to $\vec{l_{m1}}$ and $\vec{l_{m2}}$ intersect in $\mathcal{L}$[1]. Expressing this relation in the first camera coordinate system, we have

$$\mathcal{L} \parallel \vec{l_{m1}} \times \mathbf{R}^T \vec{l_{m2}}, \tag{2}$$

and since $\vec{n}$ is perpendicular to $\mathcal{L}$, we have

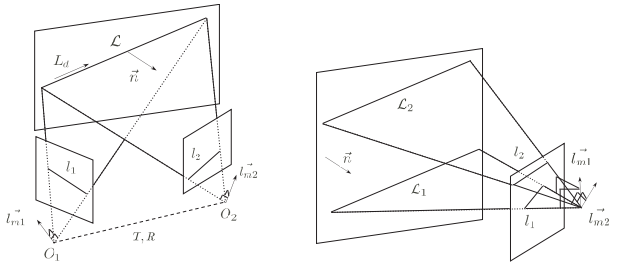
$$(\vec{l_{m1}} \times \mathbf{R}^T \vec{l_{m2}}) \cdot \vec{n} = 0. \tag{3}$$

[1]The necessary and sufficient condition for the two planes to be different is that the translation $\overrightarrow{T}$ is not parallel to the line $\mathcal{L}$.

Practically, we want to avoid computing the correspondence of two lines in two frames, so we adopt the continuous representation of Eq. 3 as

$$(l_1 \times (\dot{l_1} - \vec{\omega} \times l_1)) \cdot \vec{n} = 0, \tag{4}$$

where l_1 denotes l_{m1}, $\vec{\omega}$ is the angular velocity of the robot and $\dot{l_1}$ is the temporal derivative of the line that can be computed from the normal flow.



(a) Line constraint in multiple views.

(b) Line constraint in a single view.

Fig. 1. a) A single line is projected to two images from different viewpoints. b) Two 3D lines, belonging to the same plane, are projected to two image lines.

This is the linear equation we use to estimate $\vec{n}$. Notice, this constraint (which intuitively is known as orientation disparity in visual psychology) allows us to estimate the surface normal (that is the shape) of the plane in view, using only rotation information. At this point we should also note that no distance information is encoded to vector $\vec{n}$, which is of unit length.

B. Two or More Lines in the Same Frame

We can use the constraint in Eq. 4 also from one view. Imaging that two views are related by translation only, or similarly consider two parallel lines in one view. Given two lines l_1 and l_2 that are projected from two parallel lines, $\mathcal{L}_1$ and $\mathcal{L}_2$, in the 3D scene, we recover the orientation of $\mathcal{L}_1$ and $\mathcal{L}_2$ using Eq. 2 (Fig. 1(b)). Assuming $\mathcal{L}_1$ and $\mathcal{L}_2$ lie in the same wall, which is perpendicular to the ground, and $\vec{n} = \frac{\vec{N}}{|N|}$ as its surface normal, we then recover the surface normal of the wall from

$$(\vec{l_{m1}} \times \vec{l_{m2}}) \cdot \vec{n} = 0. \tag{5}$$

If we have more than two lines that are generated by parallel 3D lines, we can average results from Eq. 5.

The constraints discussed above provide better information than vanishing point. From two or more 3D lines, a general plane can be reconstructed. In our case, the plane is perpendicular to the ground plane, thus the surface normal can be described by only one parameter, i.e. $\frac{\alpha}{\gamma}$ (because $\vec{N} = (\alpha, 0, \gamma)^T$). In general, the robot can move based on the position with respect to the line.

C. Distance estimation

After we have computed the slant of the plane, we can also estimate its distance. For this we need the translation

T. The distance $d_{\mathcal{L}}$ of the line $\mathcal{L}$ from the camera amounts to [9]

$$d_{\mathcal{L}} = \frac{(l_1 \cdot \overrightarrow{T})}{(\dot{l}_1 + (l_1 \times \vec{\omega}))^T (l_1 \times \overrightarrow{L}_d)}, \tag{6}$$

with $\overrightarrow{L}_d$ a unit vector parallel to $\mathcal{L}$, computed as

$$\overrightarrow{L}_d = \frac{l_1 \times (\dot{l}_1 + l_1 \times \vec{\omega})}{|l_1 \times (\dot{l}_1 + l_1 \times \omega)|} \tag{7}$$

and the distance d of the plane from the camera is computed as

$$d = d_{\mathcal{L}} \overrightarrow{n} \cdot (l_1 \times \overrightarrow{L}_d) \tag{8}$$

D. Implementation details

To obtain accurate measurements of lines, we modified P. Kovesi's Matlab code[2]. The unoptimized Matlab version of the slant estimation code based on lines runs in ~ 1.5 seconds per iteration on our test bed (a 1.5 GHz Pentium M laptop with 768MB RAM).

(a) First Image (b) Second Image (c) Third Image

Fig. 2. Three frames of our line testing sequence, with the detected lines drawn in yellow color. In all cases the lines are well localized.

In Fig. 2 we present three representative frames obtained from the front camera. Note that we did not introduce any artificial landmarks, thus only objects existing in the environment, like doors and door frames are present. To find "good" lines to track, we further assume that the longest lines present in the scene are the ones on the boundary between the floor and the walls. Thus, using a threshold on the line length we are able to remove all other lines. In Figs. 4(a), 4(b) we present the distance and slant estimates which we obtained using the line constraint for a test sequence of 20 frames. We observe that the slant is estimated with good accuracy, while the distance estimation is not very accurate.

IV. Harmonic shape from texture for planar surfaces

A. Theory

In this section we assume that the camera is parallel, and the wall perpendicular to the ground. Thus $\overrightarrow{N}$ further simplifies to $(\alpha, 0, \gamma)^T$ and Eq. 1 becomes

$$\frac{1}{Z} = \alpha \frac{x}{f} + \gamma \tag{9}$$

Consider that we acquire two images I_1 and I_2 and that we know (from the odometry readings) the translation $\overrightarrow{T} = (T_x, 0, T_z)^T$ and rotation $\mathbf{R}$ relating I_1 and I_2. The first step is to locate corresponding epipolar lines on the two images (Fig. 3) using the procedure described in Alg. 1.

[2]http://www.csse.uwa.edu.au/~pk/research/matlabfns

Algorithm 1 Match Epipolar Lines

Input:

p	:	Image point in first image
T, R	:	Translation/Rotation
K	:	Camera matrix
D	:	Reference distance, randomly chosen

Output:

$[p_1, p_2]$	:	Set of corresponding points in first and second image along the epipolar lines

Algorithm:

Compute Essential Matrix	:	$E = [T]_x R$
Compute Fundamental Matrix	:	$F = K^{-T} E K^{-1}$
Compute Epipolar Line in Second Image	:	$l_2 = Fp$

Compute Corresponding epipolar line in first image using D

Interpolating the image intensity values along the epipolar lines, it is possible to rectify the two images, thus obtaining images I_1^R and I_2^R, where the epipolar lines are collinear and parallel to the horizontal axis

$$\forall x, y \quad I_2^R(x, y) = I_1^R(x + \frac{T'}{Z}, y) \tag{10}$$

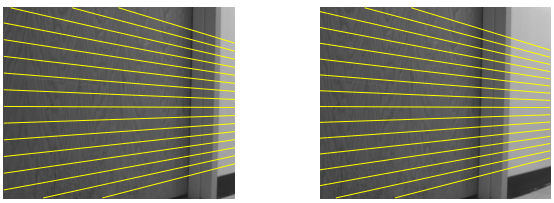
where the new translation vector is $T' = \sqrt{T_x^2 + T_z^2}$ and the new plane parameters are $(\alpha', 0, \gamma')^T = R_{RECT}(\alpha, 0, \gamma)^T$ with R_{RECT} being the rectification (rotation) matrix.

Combining Eqs. 9 and 10 and dropping for simplicity the prime notation we obtain

$$\forall x, y \quad I_2^R(x, y) = I_1^R((1 + \alpha T)x + \gamma T, y), \tag{11}$$

We can estimate α and γ using phase correlation (Table I) between the signals along the set of two epipolar lines in two steps [27]. First, we estimate α using phase correlation on the magnitude of the Fourier transform of the two signals in logarithmic coordinates (Eq. 15). Then, we warp the signals, using the estimate for α, so that only the translation component is present. Finally, we estimate γ using phase correlation on the warped signals (Eq. 17). The complete algorithm along with the equations are presented in Alg. 2.

While the algorithm presented here, solves for two (α, γ) of the three plane parameters, it is possible to obtain all three



(a) First Image (b) Second Image

Fig. 3. The epipolar lines for two frames. The translation vector is $T = [-0.011\ 0\ 0.011]^T$ meters and there was no rotation.

parameters by performing a geometric transformation on the variables and exploiting 2D phase correlation.

Algorithm 2 Estimate Plane Parameters α, γ

Input:

I_1^R, I_2^R	:	Image signals along Epipolar Lines
T	:	Translation

Output:

α, γ	:	Plane parameters

- Signals along the epipolar line y

$$\forall x, \quad I_2^R(x,y) = I_1^R((1+\alpha T)x + \gamma T, y)$$

- Compute the Fourier Transform ($\mathcal{I}_1^R, \mathcal{I}_2^R$) of I_1^R, I_2^R

$$\mathcal{F}_{x,y}\{I_2^R\}(u,v) = \frac{e^{2\pi i \frac{\gamma T}{1+\alpha T} u} \mathcal{F}_{x,y}\{I_1^R\}(\frac{u}{1+\alpha T}, v)}{|1+\alpha T|} \quad (12)$$

- Consider the Magnitude of $\mathcal{I}_1^R, \mathcal{I}_2^R$ and logarithmically transform (u,v)

$$|\mathcal{I}_2^R(\log u, v)| = \frac{|\mathcal{I}_1^R(\log u - \log(1+\alpha T), v)|}{|1+\alpha T|} \quad (13)$$

- Compute the Normalized Cross-power Spectrum (NCS_1) of $|\mathcal{I}_1^R|$, $|\mathcal{I}_2^R|$

$$NCS_1(\eta, w) = e^{2\pi i \eta \log(1+\alpha T)} \quad (14)$$

- Compute α taking the Inverse Fourier transform of NCS_1

$$\alpha = \frac{e^{u - argmax(\mathcal{F}^{-1}\{NCS_1\})} - 1}{T} \quad (15)$$

- Take the Normalized Cross-power Spectrum NCS_2 of $\mathcal{I}_1^R(\frac{u}{1+\alpha T}, v), \mathcal{I}_2^R(u,v)$ from Eq. 12

$$NCS_2(u,v) = e^{-2\pi i \frac{\gamma T}{1+\alpha T} u} \quad (16)$$

- Compute γ

$$\gamma = -\frac{(1+\alpha T) argmax(\mathcal{F}^{-1}\{NCS_2\})}{T} \quad (17)$$

TABLE I
PHASE CORRELATION CONCEPT

- Let 2D signals s_1 and s_2 be related by a translation (x_0, y_0) only, i.e.

$$s_2(x,y) = s_1(x - x_0, y - y_0)$$

- Their corresponding Fourier transforms are related by a phase shift which encodes the translation, i.e.

$$\mathcal{S}_2(u,v) = e^{-2\pi i(ux_0 + vy_0)} \mathcal{S}_1(u,v)$$

- The phase shift can be extracted from the Normalized Cross-power Spectrum of the two signals, which is defined as

$$NCS = \frac{\mathcal{S}_1(u,v)\mathcal{S}_2^*(u,v)}{|\mathcal{S}_1(u,v)\mathcal{S}_2^*(u,v)|} = e^{2\pi i(ux_0 + vy_0)}$$

- Thus, the inverse Fourier transform of NCS is a delta function around the translation point $(-x_0, -y_0)$

$$\mathcal{F}^{-1}\{NCS\}(x,y) = \delta(x + x_0, y + y_0)$$

B. Implementation details

In Fig. 4(c), 4(d) we present the results of applying this method to a series of images obtained by the left side camera of our robot. In this experiment, we used 81 epipolar lines. The red crosses denote the distance and slant estimates for each pair of frames. While slant estimation is quite accurate, still the line method provided superior results. On the other hand, this method outperformed both the line based technique and the normal flow based technique (described in Section V) in the distance estimation.

Another advantage of the method is its computational simplicity. Thus, the unoptimized Matlab code runs in ~ 1.5 seconds for an image of 81×1024 pixels (i.e., 81 epipolar lines of 1024 pixels each), with most of the time spent on warping the 2 signals in order to compute Eq. 16.

V. PLANE PARAMETERS FROM NORMAL FLOW

A. Theory

As described before, $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$ denotes a plane in the 3D world and $\overrightarrow{P} = (X, Y, Z)^T$ a point on that plane ($\overrightarrow{P} \cdot \overrightarrow{N} = 1$) and Eq. 1 is valid. When the camera moves with instantaneous rotational velocity $\overrightarrow{\Omega} = (\Omega_x, \Omega_y, \Omega_z)^T$ and translational velocity $\overrightarrow{t} = (t_x, t_y, t_z)^T$ the relative motion of the point is $V(\overrightarrow{P}) = -\overrightarrow{t} - \overrightarrow{\Omega} \times \overrightarrow{P}$. The corresponding motion of the image point $\overrightarrow{p}$ is

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} t_z x - t_x f \\ t_z y - t_y f \end{pmatrix} + \quad (18)$$

$$\begin{pmatrix} \Omega_z y - \Omega_y f + \frac{\Omega_x xy - \Omega_y x^2}{f} \\ -\Omega_z x + \Omega_x f + \frac{-\Omega_y xy + \Omega_x y^2}{f} \end{pmatrix}. \quad (19)$$

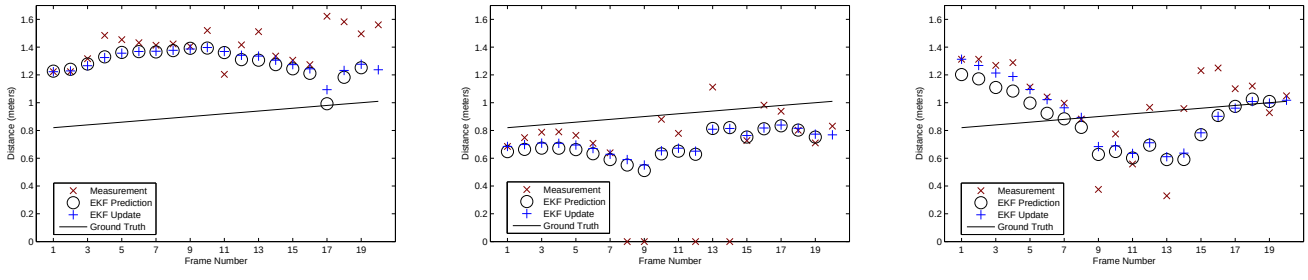
Substituting equations (1) and (19) into the image brightness consistency constraint

$$\frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0, \quad (20)$$

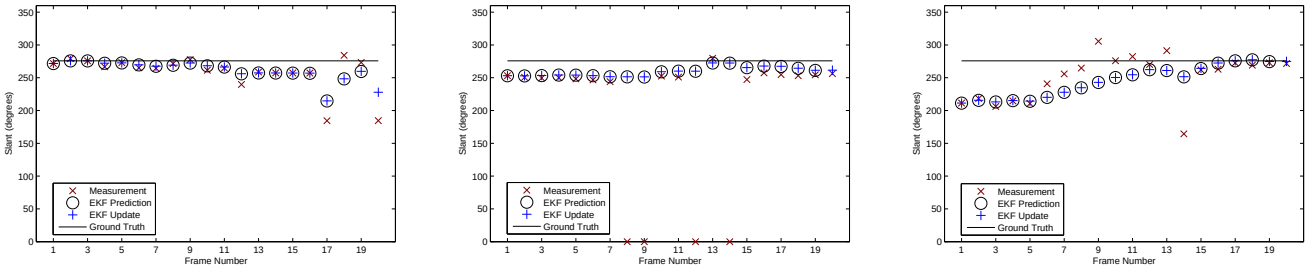
we obtain an equation bilinear in the motion parameters and the plane parameters. Note that $I(x,y,t)$ represents the image intensity at point (x,y) and time t. In our case we have restricted motion (i.e. $\Omega_x = \Omega_z = 0$ and $t_y = 0$), so we can further simplify the equation

$$\begin{array}{l} A(x\ y\ f)(\alpha\ \beta\ \gamma)^T = B \text{ ,where} \\ A = \frac{I_x}{f}(xt_z - ft_x) + \frac{I_y}{f} yt_z, \\ B = I_x f \Omega_y + \frac{\Omega_y}{f}(I_x x^2 + I_y xy) - I_t \end{array} . \quad (21)$$

According to Eq. 21, knowing the motion parameters, the camera intrinsic parameters (i.e., focal length and principal point) and the image intensity derivatives, plane estimation amounts to solving a linear system of equations for the parameters (α, β, γ).



(a) Distance using the line module (c) Distance using the texture module (e) Distance using the normal flow module



(b) Slant using the line module (d) Slant using the texture module (f) Slant using the normal flow module

Fig. 4. The results of one test run. We display the estimates of each module with a cross, the extended Kalman filter prediction (Eq. 25) with a circle and the final estimate after integration with the measurement (Eq. 28) with a plus sign. In some frames no reliable estimate could be obtained using the harmonic texture method(second column). In these cases, we display the red cross on the bottom of the corresponding figure. Also note the first EKF update is based solely on image estimates.

B. Implementation

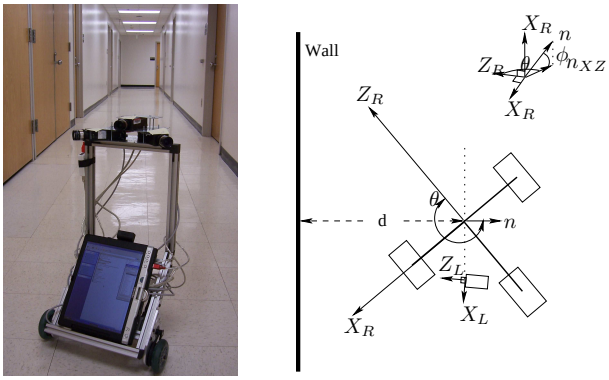
To calculate the normal flow we used the gradient based method of Lucas and Kanade ([20]) using the filtering and differentiation kernels proposed by Simoncelli ([24]) on 5 consecutive frames. For performance reasons, we first reduced the size of the image by one quarter, so we are computing the gradients on a 256×192 array (as opposed to the whole 1024×768 original images). The image size reduction has the additional advantage of reducing the pixel displacement between successive frames, thus resulting in more accurate results for plane estimation. The unoptimized Matlab version of the code runs in ~ 0.4 seconds on our testbed, with most of the time spent in computing the spatial and temporal gradients.

In Figs. 4(e), 4(f) we display the results of running the normal flow based plane estimation algorithm in the same test sequence used for Figs. 4(a), 4(b), 4(c) and 4(d). It is clear that this method is less accurate in distance and slant estimation compared to the texture and the line method, respectively.

VI. Extended Kalman Filter

Integration of the individual measurements over time is performed using an extended Kalman filter (EKF). First, let us define a robot-centric coordinate system $O_R X_R Y_R Z_R$ as follows (Fig. 5(b)); the center O_R coincides with the midpoint of the two front wheels of the robot, the X_R axis points to the left wheel of the robot, the Y_R axis points upwards and the Z_R axis forward.

As state variables for the Kalman filter we use the *distance/slant/tilt* parametrization of the plane, $\mathbf{S}(t) = [d, \theta, \phi]^T$. If we denote $\overrightarrow{n}_{XZ}$ the projection of $\overrightarrow{n}$ on the $Y = 0$ plane, then we define the *slant* θ to be the angle between the Z_R axis and $\overrightarrow{n}_{XZ}$, as shown in Fig. 5(b). *Tilt* ϕ is the angle between the Y component of n and the XZ plane. Thus the transformation between the two different parameterizations is



(a) Photo of Robot (b) Robot Sketch

Fig. 5. a)The ER1 robot equipped with 3 Firewire cameras. The height of the robot is $\sim$ 70 cm. In the background, part of the corridor, where we conducted some experiments, is shown. All the walls and doors are textureless and there exist significant specular highlights on both the walls and the floor caused by the light sources. b) The distance and angle θ between the robot and the wall are defined with respect to a coordinate system attached to the robot. The surface normal projected on the $X - Z$ plane (n_{XZ}) is also displayed.

$$\begin{bmatrix} d &=& \frac{1}{\sqrt{\alpha^2+\beta^2+\gamma^2}} \\ \theta &=& \arctan\left(\frac{\alpha}{\gamma}\right) \\ \phi &=& \arccos\left(\frac{\beta}{d}\right) \end{bmatrix}$$

Assuming that the control vector $\mathbf{U}(t)$ consists of the instantaneous translational and rotational velocities of the robot ($v(t), \omega(t)$) respectively and Δt denotes a time interval, the evolution of the system over time can be described as

$$\mathbf{S}(t+\Delta t) = \mathbf{F}(\mathbf{S}(t), \mathbf{U}(t)) \Leftrightarrow$$

$$\begin{bmatrix} d(t+\Delta t) &=& d(t) + v(t)\cos\theta(t)\Delta t + \epsilon_{11} \\ \theta(t+\Delta t) &=& \theta(t) - \omega(t)\Delta t + \epsilon_{12} \\ \phi(t+\Delta t) &=& \phi(t) + \epsilon_{13} \end{bmatrix}, \quad (22)$$

where we use the assumption that $\cos\theta(t) \simeq \cos\theta(t+\Delta t)$, i.e. the rotational velocity $\omega(t)$ is small and approximately constant over Δt and the discretization step Δt is also small. Furthermore, we denote with ϵ_{1i} the errors in the state prediction (with covariance $\mathbf{Q}$).

Our measurement vectors (Z_1, Z_2, Z_3) consist of the plane parameters calculated using the different methods described in Sections III, IV and V respectively, converted to the distance/slant/tilt parametrization. We consider the combined measurement to be a weighted linear combination of the individual measurements i.e., $Z(t) = \sum_{i=1}^{3} C_i Z_i$, where the weights C_i encode the (inverse) uncertainty of the estimates using different methods, which we derived as follows.

The line module bases the accuracy of the plane estimation on how well it detects and localizes the line. The harmonic texture module is using the magnitude of the Inverse Fourier transform of the Normalized Cross-power Spectrum (Eqs. 15,17) and the normal flow module is using the condition number of the linear system (Eq. 21).

The system evolution (Eq. 22) is not linear with respect to the state vector $\mathbf{S}(t)$ and the control vector $\mathbf{U}(t)$. That's why we need to use an extended Kalman filter and linearize the equations by considering the Jacobian matrix as shown in Table II.

A. Results

Figs. 4(a), 4(b), 4(c), 4(d), 4(e) and 4(f) depict the results when we combined the line, texture and normal flow methods, respectively with the odometry measurements using the EKF. More specifically, in these figures, black circles denote the prediction about the current state using only the previous state and dead reckoning information (Eq. 25), while blue pluses denote the final prediction of the state after the measurements from each individual module are also considered (Eq. 28). It is clear that integration of measurements over time *significatly* improves the accuracy and robustness of the method.

VII. Motion Control

An important part of any navigation system is the motion control subsystem. In this particular setting the goal is to move along the corridor avoiding the obstacles that might lie ahead of us. The motion control strategy described below refers to the "wall-following" behavior. Using the same policy one could implement the "centering" behavior.

TABLE II
EXTENDED KALMAN FILTER EQUATIONS

Jacobian of system evolution with respect to the state vector $S(t)$

$$\mathbf{A}(t) = \begin{bmatrix} 1 & -v(t)\sin\theta(t)\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (23)$$

Jacobian of system evolution with respect to the control vector $U(t)$

$$\mathbf{W}(t) = \begin{bmatrix} \cos\theta(t)\Delta t & 0 & v(t)\cos\theta(t) \\ 0 & -\Delta t & -\omega(t) \\ 0 & 0 & 0 \end{bmatrix} \quad (24)$$

State prediction equations (Mean $\hat{\mathbf{S}}$ and Covariance $\hat{\mathbf{P}}$)

$$\begin{bmatrix} \hat{d}(t+\Delta t) &=& \bar{d}(t) + \bar{v}(t)\cos\bar{\theta}(t)\bar{\Delta t} \\ \hat{\theta}(t+\Delta t) &=& \bar{\theta}(t) - \bar{\omega}(t)\bar{\Delta t} \\ \hat{\phi}(t+\Delta t) &=& \bar{\phi}(t) \end{bmatrix}. \quad (25)$$

$$\hat{\mathbf{P}}(t+\Delta t) = \mathbf{A}(t)\bar{\mathbf{P}}(t)\mathbf{A}(t)^T + \mathbf{W}(t)\mathbf{Q}(t)\mathbf{W}(t)^T \quad (26)$$

Kalman Gain $\mathbf{K}$

$$\mathbf{K}_i(t) = \hat{\mathbf{P}}(t)(\hat{\mathbf{P}}(t) + \mathbf{R}(t))^{-1} \quad (27)$$

Measurement update equations (Mean $\bar{\mathbf{S}}$ and Covariance $\bar{\mathbf{P}}$)

$$\bar{\mathbf{S}}(t+\Delta t) = \hat{\mathbf{S}}(t+\Delta t) + \mathbf{K}(t)(\mathbf{Z}(t+\Delta t) - \hat{\mathbf{S}}(t+\Delta t)) \quad (28)$$

$$\bar{\mathbf{P}}(t+\Delta t) = (\mathbf{I} - \mathbf{K}(t))\hat{\mathbf{P}}(t+\Delta t) \quad (29)$$

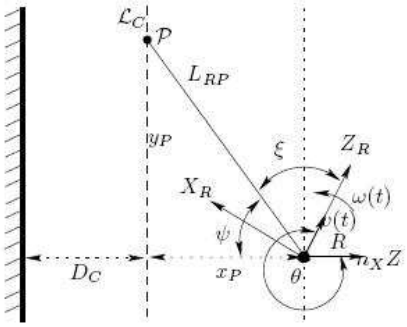


Fig. 6. The robot R is moving with translational and rotational velocities $v(t), \omega(t)$ respectively, while it is located x_P units away from the virtual line $\mathcal{L}_C$.

Let's define the input to the motion control algorithm to be the state vector of the Kalman filter, that denotes the position of the left wall with respect to the robot. Ideally, we want the robot to remain at a constant distance (denoted with D_C) from the wall, thus following the line $\mathcal{L}_C$ as shown in Fig. 6. In practice, the robot's trajectory is restricted by motion dynamics as well as the constraint that the rotational and translational velocities should remain constant, while the camera is recording the frames. As a consequence, the system is only allowed to perform small motion changes between two successive frames, thus it is hard to follow the virtual line. Instead, a point $\mathcal{P}$ along the line $\mathcal{L}_C$ is picked and the

robot's motion is regulated accordingly, so that it approaches $\mathcal{P}$. Next we describe how to do this.

Let's assume that point $\mathcal{P}$ is y_P meters away from the robot along the line $\mathcal{L}_C$ and forms an angle ψ as shown in Fig. 6. Furthermore, the robot is situated x_P units away from $\mathcal{L}_C$ and is moving with instantaneous translational and rotational speed $v(t), \omega(t)$ respectively. Note that the translational velocity is always along the direction of the Z-axis of the robot and the rotational velocity is around the Y-axis. Then, we have:

$$\psi = \arctan(\frac{y_P}{x_P}) \quad (30)$$

$$\xi = \theta - \pi - \psi \quad (31)$$

The line segment L_{RP} has length $D = \sqrt{x_P^2 + y_P^2}$. An approximation of the time that is required by the robot to reach point $\mathcal{P}$ is $\Delta t = \frac{D}{v(t)}$. The new rotational velocity $(\omega(t + \Delta t))$ of the robot should be:

$$\omega(t + \Delta t) = \frac{\xi}{\Delta t} = v(t)\frac{\theta - \pi - \arctan\frac{y_P}{x_P}}{\sqrt{x_P^2 + y_P^2}} \quad (32)$$

VIII. Experiments

We have used the robotic platform ER1 from Evolution Robotics. On top of it, we have placed a front and two side Firewire cameras (SONY XCD-X700). The side cameras form angles $(\sim 45^o, \sim -45^o)$ with the front camera as shown in Fig. 5(a). In the following experiments we used the left side camera and the front camera. We run the texture based as well as the normal flow based code on the left side camera and the line-based code on the front camera.

The goal of the experiments is to convey two messages;

- The accuracy and robustness of the system significantly increases with the *integration* of individual measurements from *different subsystems* over *time*.
- When using all the methods the robot is able to move along a mostly textureless corridor.

A. Constant Distance Experiment

The goal of this first experiment was for the robot to move a distance of 20 meters along a corridor without hitting the side walls. The corridor had a width of 1.8 meters, so we instructed the robot to try to maintain a distance of 0.9 meters from the left, while moving with velocity 5 cm/sec. The initial orientation of the robot with respect to the wall varied from 0^o (parallel to the wall) to -20^o (moving away from the left wall) and $+20^o$ (moving towards the wall). We made multiple runs each time activating a different submodule with and without integrating the measurements with dead reckoning using the EKF. Finally, we performed the experiment using all the submodules together. The results are presented in Fig. 7. It is clear that each individual module in isolation performs poorly (with the exception of the line module). Integrating the measurements of a single module over time (using the EKF) greatly improves the robustness of the method. Finally, combining the measurements from different submodules, provides the most robust setting.

B. Average Distance Experiment

In this experiment we let the robot move on the corridor (still trying to maintain a distance of 0.9 meters from the left wall) with velocity 5 cm/sec, and measured the average distance traversed before the hitting the wall. We performed the experiment multiple times activating a different module or combinations of modules. The results, namely the average distance for each combination, are presented in Fig. 8. Again, we observed that a single module performs very poorly (with the exception of the line module), while combining modules together and integrating the estimates over time greatly improves the result. When the average distance is larger than 20 meters, it indicates that the robot is approaching the end of the corridor and thus we had to terminate the specific run.

IX. Conclusions

In this paper we presented two new methods for computing the 3D structure of a piece-wise planar scene from video. We also used an existing method for 3D shape estimation based on normal flow. The three methods base their estimation on complementary information. More specifically, while the normal flow technique considers individual features (i.e.

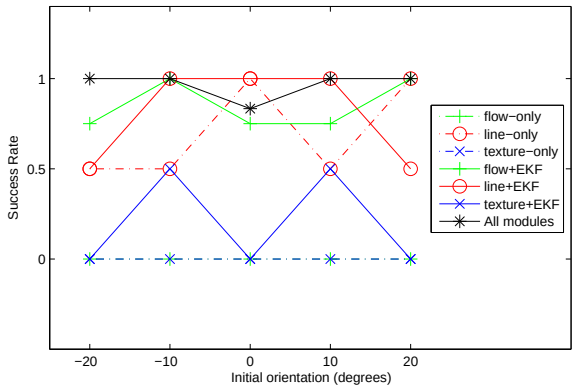


Fig. 7. Percentage of times that the robot was able to move than 20 meters without hitting the side walls.

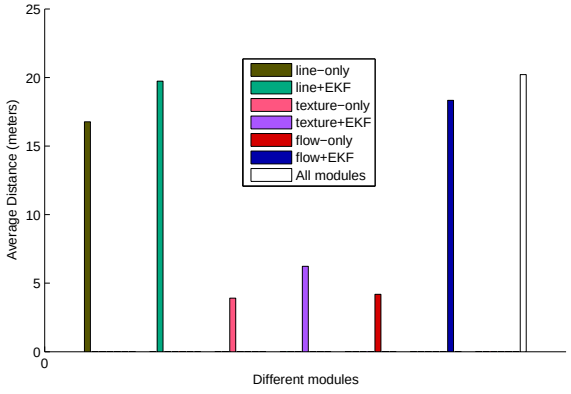


Fig. 8. Average distance that the robot was able to move using measurements from a single or multiple modules.

sharp intensity changes) within the object, the texture method considers the whole area within it. The line method, on the other hand, uses the boundary of an object. Depending on the case, we expect at least one of the methods to provide accurate measurements. For example, when we observe a mostly uniformed colored object, we anticipate that the line method will be able to accurately track the boundary of it and produce accurate results, while the remaining two modules will fail. On the other hand, when the object is highly textured, the line method might not be able to locate the boundaries accurately, but the two other methods will produce good results. For that reason, we emphasize that the integration of all three modules is the right approach, if one wants to build a robust system. For similar reasons, integration of the individual measurements over time is equally important. In this paper, we use odometry measurements from the wheels' encoders, but we might as well estimate the motion from the video (*visual odometry*, also known as *ego-motion* estimation [29],[6],[21]) or using other sensors. We present experiments in the context of visual navigation on indoor environments and verify that the combined usage of all three modules produces a robust system.

We plan a number of extensions to this work. In order for the robot to navigate in more complex environments, we need to incorporate a scene segmentation scheme into this framework. We currently develop segmentation modules based on information from motion, intensity and lines. Ultimately, the goal is to identify and track individual objects over frames, thus addressing the visual SLAM problem.

X. Acknowledgments

The authors gratefully acknowledge the contribution of National Science Foundation and the reviewers' comments.

References

[1] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards urban 3d reconstruction from video. *3DPVT*, 2006.

[2] A. Argyros and F. Bergholm. Combining central and peripheral vision for reactive robot navigation. *CVPR*, 2:646–651, 1999.

[3] A. Argyros, D.P. Tsakiris, and C. Groyer. Biomimetic centering behavior: Mobile robots with panoramic sensors. In K. Daniilides and N. Papanikolopoulos, editors, *IEEE Robotics and Automation Magazine , special issue on Panoramic Robotics*.

[4] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.

[5] B. P. Bogert, M. J. R. Healy, and J. W. Tukey. The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In M. Rosenblatt, editor, *Time Series Analysis*.

[6] T. Brodsky, C. Fermüller, and Y. Aloimonos. Structure from motion: Beyond the epipolar constraint. *Int. J. Computer Vision*, 37:231–258, 2000.

[7] D. Coombs, M. Herman, T. Hong, and M. Nashman. Real-time obstacle avoidance using central flow divergence and peripheral flow. *ICCV*, 1995.

[8] D. Coombs and K. Roberts. Centering behavior using peripheral vision. *CVPR*, 1993.

[9] K. Daniilidis. *On the Error Sensitivity in the Recovery of Object Descriptions*. PhD thesis, Department of Informatics, University of Karlsruhe, Germany, 1992. In German.

[10] A. Davison. Real-time simultaneous localization and mapping with a single camera. *ICCV*, 2003.

[11] G.K. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[12] A. Duchon and W. Warren. Robot navigation from a gibsonian viewpoint. *IEEE Conference on Systems, Man and Cybernetics*, 1994.

[13] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press.

[14] D.J. Fleet and A.D. Jepson. Computation of component image velocity from local phase information. *IJCV*, 5:77–104, 1990.

[15] H. Foroosh, J. B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Transactions on Image Processing*, 11:188–200, 2002.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

[17] Hui Ji and C. Fermüller. Noise causes slant underestimation in stereo and motion. *Vision Research*, 46(19):3105–3120, 2006.

[18] C.D. Kuglin and D.C. Hines. The phase correlation image alignment method. *IEEE Conference on Cybernetics and Society*, 1975.

[19] R. W. Schafer L. R. Rabiner. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

[20] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[21] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1:652–659, 2004.

[22] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[23] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo for robot navigation : Learning from bees. *CVPR*, 1993.

[24] E. Simoncelli. Design of multi-dimensional derivative filters. *ICIP*, 1:790–793, 1994.

[25] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *Proc. British Machine Vision Conference*, Edinburgh, 2006.

[26] B. Srinivasa and B. N. Chatterji. An fft-based technique for translation, rotation and scale-invariant image registration. *IEEE Transactions on Image Processing*, 8(8):1266–1271, 1996.

[27] M. Srinivasan, J. Chahl, K. Weber, S. Venkatesh, M. Nagle, and S. Zhang. Robot navigation inspired by principles of insect vision. *Robotics and Autonomous Systems*, 26(2):203–216, 1999.

[28] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[29] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *Trans. on PAMI*, 6:13–27, 1984.

[30] J. Weng, T. Huang, and N. Ahuja. Motion and structure from line correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336, 1992.