Design and Evaluation of a System for Vision–Based Vehicle Convoying *

Rodrigo L. Carceroni Craig Harman Christopher K. Eveland
Christopher M. Brown

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 678

January 1998

Abstract

In this paper, we discuss how to process visual information in convoying applications, using only low–cost, off–the–shelf hardware. We introduce a numerical algorithm for real–time perspective pose estimation that uses strong task–specific constraints to achieve efficiency and stability. Through extensive experiments performed with synthetic data, we show that this approach yields more accurate recovery than a general–purpose structure–and–motion recovery framework known as the Variable State Dimension Filter, even when some of its fundamental task–specific assumptions are only partially valid. In addition, we discuss efficient ways to perform low–level vision with off–the–shelf hardware, and we present a two–level control strategy that uses high–frequency odometry data to stabilize visual control. Real–world convoying experiments show that our tracking–and–control system performs quite well in the sense that it manages to keep targets in view, tolerates changes in lighting conditions, and enables vehicles to keep up with complex maneuvers performed by other members of the convoy, such as 180–degree turns.

The University of Rochester Computer Science Department supported this work.

*This material is based on work supported by CAPES process BEX-0591/95-5, NSF IIP grant CDA-94-01142, NSF grant IRI-9306454 and DARPA grant DAAB07-97-C-J027.

1 Introduction

One of the main obstacles to the practical feasibility of many computer vision techniques has been the necessity of using expensive specialized hardware for low–level image processing in order to achieve real–time performance. However, gradual improvements in the architectural design and in the manufacturing technology of general–purpose microprocessors have made their usage for low–level vision more and more attractive. In this report, we demonstrate the real–time feasibility of a tracking system for smart vehicle convoying that uses basically a 133 MHz dual Pentium board, an M68332 micro–controller and a commercial PCI frame grabber. The task at hand consists of enabling an autonomous mobile robot with a single off–the–shelf camera to follow a target placed on the posterior part of another mobile platform, controlled manually.

The key ideas used to achieve efficiency are quite traditional concepts in computer vision. In the low–level image processing front, we use multi-resolution techniques to allow the system to locate quickly the regions of interest on each scene and then to focus its attention exclusively on them, in order to obtain accurate geometrical information at relatively low computational cost. In the higher–level processes of geometrical analysis and tracking, the key idea is to use as much information available *a priori* about the target as possible, in order to develop routines that combine maximum efficiency with high precision, provided that their specialized geometry and dynamics assumptions are met. Finally, for the control itself, we use a two–level strategy that combines error signals obtained from the analysis of visual data with error signals obtained from odometry.

So, while we do introduce some novel formulations, especially in the context of geometrical analysis of the scenes, the main goal of the present work is clearly to demonstrate that by carefully putting together several well–established concepts and techniques in the area of computer vision, it is possible to tackle the challenging problem of smart vehicle convoying with low–cost equipment.

In Section 2, we discuss some of the related work in the areas of tracking and pose estimation. In Section 3 we discuss our approach for recovering three–dimensional information and exploiting the temporal coherence of sequences of images. In Section 4 we perform an empirical comparison between this approach and two possible alternatives, using synthetic data. In Section 5 we discuss the aspects related to the problem of low level image processing: how to extract and correctly identify useful features in the input images. In Section 6 we present our approach to deal with the problem of visual control, that is, how to set the speed and steering of the trailing robot so that it does not loose track of the leading mobile platform. Still in Section 6, report the results of some real–world convoying experiments. Finally, in Section 7 we present our concluding remarks.

2 Background

The task of tracking a single target can be divided in two parts: *acquisition* and *tracking proper* (below, simply *tracking*) [9]. Acquisition involves the identification and localization (possibly via motion detection and segmentation) of the target, as well as a rough initial

estimation of its pose (position and orientation), velocities and possibly some other *state variables* of interest. This phase is in some aspects quite similar to the problem of object recognition. Usually, generality is more important at this point than in the subsequent tracking phase, because in several practical applications, many different targets of interest may appear in the field–of–view of the tracking system and thus it is not possible to use techniques that work only for one particular type of target.

The information obtained in the acquisition phase is then used to initiate the tracking phase, in which the target's state variables are refined and updated at a relatively high frequency. In this report we argue that in this phase, after the target has been identified and its initial state has been properly initialized, all the specific information available about its geometry and dynamics should be exploited in the development of specialized routines that are appropriate for real–time usage and, still, require only inexpensive general–purpose hardware.

As suggested by Donald Gennery [9], the tracking phase, which is the major problem studied here, can be divided into four major subtasks:

1. **Prediction:** Given a multi-valued time–series with the history of (noisy) measurements of the target state variables performed so far, it is necessary to predict the values of these variables in the next sampling instant, so that the tracking system can always restrict its search for the target to a relatively small part of the scene. Traditionally, this extrapolation of the values of the state variables is done recursively, for efficiency reasons. In other words, at any point in time, the tracker keeps only a vector of current estimates for the *true* (as opposed to *measured*) values of the state variables and some of the higher–order moments of the multi-valued time–series (typically the covariance matrix). Then, given the new measurements, all these variables are extrapolated for some future instant and the whole process can be repeated as soon as the another set of measurements is available.

2. **Projection:** Then, given the predicted pose of the target and a certain model for the 3-D–to–2-D transformation performed by the camera, it is necessary to determine the appearance of the target in the scene. Typically, this task is formulated as the determination of positions, orientations or apparent angles, and the visibility analysis, for a set of distinctive target features.

3. **Measurement:** The next step is to search for the expected visible features in the image. The problem of identifying which image features correspond to each model features, known as the *correspondence problem*, is the hardest aspect of this task. However, this problem can be avoided (or at least ameliorated) if the features are distinctive enough to be uniquely distinguished regardless of the pose of the target (via color information, for instance). Another useful trick is to make the other three steps of the tracking phase tolerant to false matches in the solution of the correspondence problem, via the use of robust statistics.

4. **Back-projection:** Finally, it is necessary to compute the discrepancy between the actual image measurements and the image measurements that would be expected given the analysis performed in the Projection step. Of course, it is also necessary to

determine how this discrepancy affects the current estimate for the state of the target. Ultimately, some sort of back-projection from the 2-D image plane to the 3-D scene space is needed to perform this task.

In our tracking system, one of the steps in which we exploit most heavily the availability of *a priori* information about the target in order to improve efficiency and accuracy is Back-projection. More specifically, at this point (as well as in the Projection step) we make use of the fact that our target is a rigid object composed by points whose relative 3-D positions are known *a priori*. The problem of recovering the pose of a three–dimensional object from a single monocular image, given a geometrical *model* for this object, has been heavily studied in the last two decades or so. The solutions proposed in the literature can be classified, according to the nature of the imaging models and mathematical techniques employed, as: analytical perspective, affine and numerical perspective.

The general idea of the analytical solutions is to work with a fixed number of known correspondences between model and image features. Then, they express image properties such as feature positions [10; 12; 1; 8; 16], orientations [7; 26] or apparent angles [32; 29; 21] as a function of a predefined set of pose parameters. By matching the resulting expressions against the corresponding actual image measurements, one can derive a set of polynomial equations involving the pose parameters. Finally, if the number of correspondences is big enough, these equations can be combined algebraically, yielding the desired pose.

The problem with this approach is that, if the imaging process is modeled as a perspective transformation, then the equations that relate the image measurements to the three–dimensional geometrical information known *a priori* are non–linear. Because of this, all the solutions cited above work only for up to four features and can not be efficiently extended to deal with more complex geometrical shapes. Furthermore, it is known that any analytical solution based on fewer than six point correspondences is necessarily ambiguous [8], yielding multiple plausible answers for certain problem instances. In addition, due to the use of multiple non–linear constraints, most of the techniques mentioned above rely on finding the roots of polynomial equations with degree higher than four. Unfortunately, there is provably no closed–form solution for this problem itself. Finally, many of these techniques have very poor error propagation properties. In a survey performed by Haralick and Lee [10], all the techniques tested were found to produce large errors (at least 0.1% in the actual 3-D positions of the object features), just as a result of the propagation of rounding errors with single precision arithmetic. Of course, this problem becomes much more serious if we take into account the effect of quantization noise in the imaging process, for instance.

So, the source of most of these problems is the intrinsic non–linearity of the geometrical constraints that arise when the imaging process is modeled as a perspective transformation. Under certain special conditions, an imaging transformation that is actually perspective (or even more complex, if we take into account lens distortion, for instance), can be reasonably approximated with much simpler models. Techniques based on linearized camera models [2; 14; 13] are also simple, efficient, and, contrary to the analytical solutions, they work for scenes with arbitrarily many features. However, they are not able to cope with significant perspective distortion and, unfortunately, since we use a camera with a relatively wide

field–of–view to avoid losing track of the target in our application, we have to face this type of complication.

So, an ideal pose recovery algorithm should combine the generality of a perspective camera model with the robustness of the schemes based on affine approximations. Indeed, it is possible to satisfy this requirement in practice by casting the problem of pose estimation into an equivalent multivariate numerical parameter estimation problem. There are at least two distinct ways in which this can be done.

The most traditional, straightforward, and widely used numeric approach consists of defining a measure of the discrepancy between the actual image measurements and the measurements that would be expected given a perspective camera model and an arbitrary estimate for the unknown pose. Then, by replacing the chosen error measure (which is a non–linear function of the pose parameters) with a local linear approximation around the point corresponding to the current pose estimate, one can compute a correction that in general yields a better pose estimate. This process can be iterated until (ideally) the error function is locally minimized and the current pose estimate converges to the actual pose, within a predefined desired precision.

David Lowe [20; 19; 18] proposed a classic solution along this line. Given a certain pose estimate, his algorithm computes the expected values for a vector of measurements (positions or orientations) in the resulting image, using a non–linear projective model. Then, Newton's iterative gradient method is employed to minimize this error vector in a least–squares sense. Lowe's algorithm was later improved by other researchers through the use of more accurate projective models [3; 31; 15] and optimization techniques with better convergence properties [28]. Similar solutions were also proposed for the specific case of independent orientation recovery from line correspondences [28; 17; 33].

A more recent approach, suggested by DeMenthon and Davis [6], consists of computing an initial estimate for the pose with a weak perspective camera model and then refining this model numerically, in order to account for the perspective effects in the image. The key idea is to isolate the non–linearity of the perspective projection equations with a set of parameters that explicitly quantify the degree of perspective distortion in different parts of the scene. By artificially setting these parameters to zero, one can then generate an affine estimate for the pose. Then, the resulting pose parameters can be used to estimate the distortion parameters and this process can be iterated until the resulting camera model (presumably) converges to full perspective. Oberkampf *et al* [27] extend DeMenthon–Davis's original algorithm to deal with planar objects (the original formulation is not able to handle that particular case) and Horaud *et al* [11] propose a similar approach that starts with a paraperspective rather than a weak perspective camera model.

The main advantage of this kind of approach is its efficiency. Like the optimization–based techniques, each iteration of the algorithms based on initial affine approximations demands the resolution of a possibly over-constrained system of linear equations. However, in the latter methods, the coefficient matrix of this system depends only on the scene model and thus its (pseudo) inverse can be computed off–line, while the optimization–based techniques must necessarily perform this expensive operation at every single iteration. [6].

However, all the solutions mentioned so far are much more general than the application that we have in mind, namely: track and follow a target placed on the posterior position

of a non–holonomic vehicle whose motion is roughly constrained to a plane perpendicular to the image plane of the (single) camera used. Most of the model–based pose recovery algorithms available in the literature do not impose any restriction on the possible motions of the target and thus use camera models with at least six degrees of freedom, such as the *perspective*, the *weak perspective* and the *paraperspective* models.

Wiles and Brady [30] propose some simpler camera models for the important problem of smart vehicle convoying on highways. In their analysis, they assume that a camera rigidly attached to a certain trailing vehicle is used to estimate the structure of a leading vehicle, in such a way that the paths traversed by these two vehicles are composed exclusively by a series of translations and rotations parallel to a unique "ground plane". In spite of the focus of their research being the recovery of structure from motion, many of their observations and suggestions can be generalized to the analogous problem of pose estimation.

Clearly, the application–specific constraints reduce the number of DOF in the relative pose of the leading vehicle to three. Because the camera does not undergo any rotation about its optical axis, the $\mathbf{x}$ axis of the camera frame can be defined so as to be parallel to the ground plane. Furthermore, the tilt angle between the camera's optical axis and the ground plane (α) is fixed and can be measured in advance. In this situation, the general perspective camera can be specialized to a model called perspective *Ground Plane Motion* (GPM) camera, whose *extrinsic parameter matrix* is much simpler than that of a six–DOF perspective model.

In our application we take this idea to an extreme. We not only simplify even more the model proposed by Wiles and Brady with the assumption that the image plane is normal to the ground plane ($\alpha = 0$), but also use a specially–engineered symmetrical pyramidal target, in order to make the problem of inverting the perspective transformation performed by the camera as simple as possible. In addition, inspired by the work of DeMenthon and Davis [6], we adopt a solution based on the numerical refinement of an initial weak perspective pose estimate, in order to obtain accuracy at low computational cost. But rather than starting from scratch and iterating our numerical solution until it converges for each individual frame, we interleave this numerical optimization with the recursive estimation of the time series equivalent to the state of the target, as suggested by Donald Gennery [9]. So, only one iteration of the numerical pose recovery is performed per frame and the temporal coherence of the visual input stream is used in order to keep the errors in the estimates for the target state down to a sufficiently precise level.

3 Real–Time Pose Estimation

To some extent all computer vision is "engineered" to remove irrelevant variation that is difficult to deal with. Still, our engineering takes a fairly extreme form in that we track a specially–constructed three–dimensional calibration object, or target, attached to the lead robot. Our special target, placed on the back of the leading mobile robot, consists of two planes parallel with respect to each other and orthogonal to the ground plane, kept in fixed positions with respect to the mobile robot by a rigid rod, as shown in Fig. 1(a). The plane closer to the center of the leading robot (typically further away from the camera) contains a

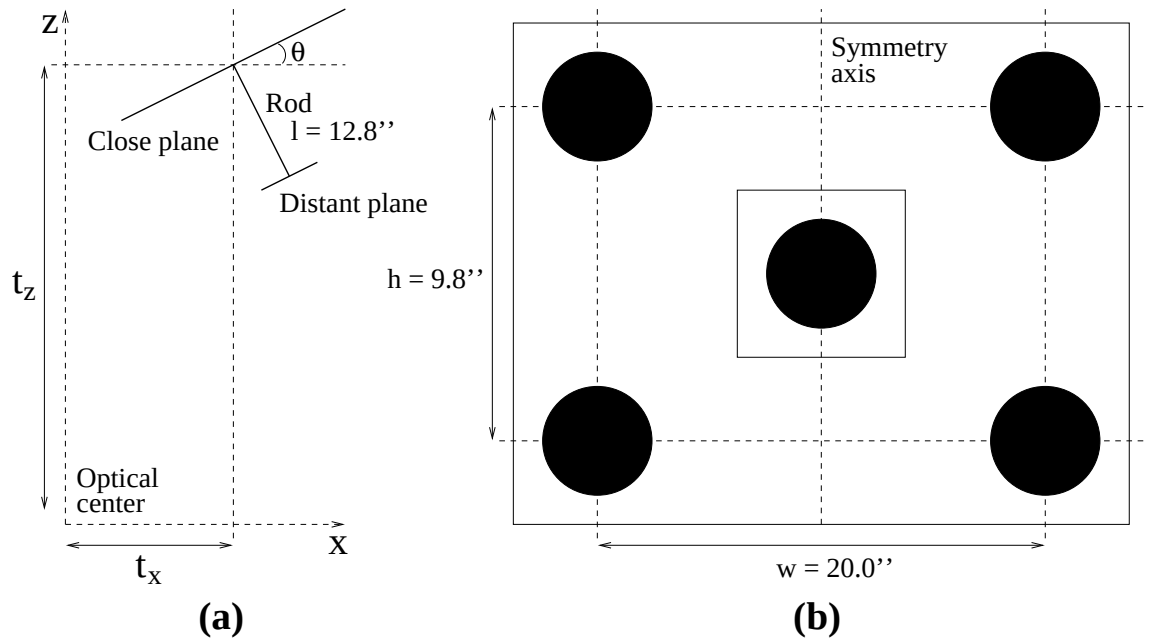

Figure 1: Geometry of the target placed on the posterior part of the leading vehicle: (a) top view; and (b) frontal view.

rectangle composed by four identical circles with a diameter of 4.0 inches each, so that two of the virtual edges defined by these points are parallel to the ground plane, as shown in Fig. 1(b). The other target plane (more distant from the leading robot) contains a unique circle with a diameter of 3.5 inches. These two planes are arranged so that the orthogonal projection of the this 3.5–inch circle on the plane closer to the leading robot lies on the axis of vertical symmetry of the rectangle composed by the other four points (Fig. 1(b)).

From the point of view of our tracking algorithm, the state of this target is described with respect to a coordinate system attached to the camera, whose **x** and **y** axes correspond to the horizontal (rightward) and vertical (downward) directions on the image plane, respectively, and whose **z** axis corresponds to the optical axis of the camera (forward). Due to the ground–motion constraint, the target has only 3 DOF with respect to the camera. The state variables used to encode these DOF are: the distances between the camera's optical center and the centroid of the target's rectangle along the **x** and **z** axes, denoted by t_x and t_z, respectively, and the counterclockwise (as seen from the top) angle between the **x** axis and the plane that contains the rectangle, denoted by θ.

At each step of the tracking phase, the tracker initially performs a *a priori Prediction* of the state of the target, based uniquely on the history of the values for the state variables. More specifically, since our mobile robots can stop and turn quite sharply, we perform this prediction with a simple velocity extrapolation for each state variable, because under these circumstances of a highly-maneuverable target and rather slow update rates, more complex filtering is impractical and destabilizing. Let $\hat{v}^{(i)}$ and $v^{(i)}$ denote the *estimated* and *measured* values for state variable v at step i, respectively, where v is one of t_x, t_z and θ. Then the

Circle	x_i	y_i	z_i
Top left	$-w/2$	$-h/2$	0
Top right	$+w/2$	$-h/2$	0
Bottom left	$-w/2$	$+h/2$	0
Bottom right	$+w/2$	$+h/2$	0
Central	0	h_c	$-l$

Table 1: Coordinates of the target centroids in the model reference frame. Here w denotes the centroid–to–centroid horizontal width of the target's rectangle, h denotes the centroid–to–centroid vertical height of the target's rectangle and l denotes the orthogonal distance between the two parallel planes that compose the target.

predictions performed by the tracker are:

$$\begin{cases} \hat{t}_x^{(i)} &= 2\,t_x^{(i-1)} - t_x^{(i-2)}, \\ \hat{t}_z^{(i)} &= 2\,t_z^{(i-1)} - t_z^{(i-2)}, \\ \hat{\theta}^{(i)} &= 2\,\theta^{(i-1)} - \theta^{(i-2)}. \end{cases} \tag{1}$$

The predicted values for the state variables are used to compute the appearances, on the image plane, expected for the five circles that compose the target. This corresponds to the *Projection* step, according to the outline presented in Section 2, and amounts to projecting the known geometry of the target, according to our simplified perspective GPM camera model. The model reference frame is defined so that its origin is the centroid of the target's rectangle, its **y** axis is aligned with the **y** axis of the camera frame and its **z** axis, when placed at the origin, points in the opposite direction of the centroid of the central circle. So, using the fact that the target has vertical symmetry, one can express the coordinates of the circle centroids in this frame according to the Table 1.

According to our imaging model, the projection equation that yields the image coordinates of an arbitrary point i, $[u_i, v_i]^T$, as a function of its coordinates on the model reference frame, $[x_i, y_i, z_i]^T$, is:

$$[u_i, v_i, 1]^T = \lambda\, M_{int}\, M_{ext}\, [x_i, y_i, z_i, 1]^T, \tag{2}$$

where the matrix of *intrinsic camera parameters*, M_{int}, (calibrated *a priori*) and the matrix of *extrinsic camera parameters*, M_{ext}, (estimated by the tracker) are given by:

$$M_{int} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

$$M_{ext} = \begin{bmatrix} \cos\hat{\theta} & 0 & -\sin\hat{\theta} & \hat{t}_x \\ 0 & 1 & 0 & h_0 \\ \sin\hat{\theta} & 0 & \cos\hat{\theta} & \hat{t}_z \end{bmatrix}. \tag{4}$$

These predicted appearances are then used in the *Measurement* phase, which corresponds to the low level processing of the current input image, as we describe in the Section 5.

Finally, the low–level image processing module returns the positions, measured in the image plane, for the apparent centroids of the target circles, which are used in the *Reprojection*, yielding the *measured* state of the target in the current step of the tracking phase. Let the apparent centroids of the top left, top right, bottom left, bottom right and central circles be denoted by $[u_{tl}, v_{tl}]^T$, $[u_{tr}, v_{tr}]^T$, $[u_{bl}, v_{bl}]^T$, $[u_{br}, v_{br}]^T$ and $[u_c, v_c]^T$, respectively. In order to simplify the derivation of the equations that yield the measured state variables t_x, t_z and θ, we define the *image measurements* m_x, m_z and m_θ, as follows:

$$m_x = \frac{u_{bl} + u_{tl} + u_{br} + u_{tr}}{4} - u_0, \tag{5}$$

$$m_z = \frac{v_{bl} - v_{tl} + v_{br} - v_{tr}}{2}, \tag{6}$$

$$m_\theta = u_c - u_0. \tag{7}$$

By replacing the predicted state variables in Eqs. (2) to (4) with their measured counterparts and substituting the resulting expressions (as well as the centroid coordinates given in Table 1) into Eqs. (5) to (7), one can express the image measurements above as a function of the state variables:

$$m_x = \frac{f_u}{2}\left(\frac{t_x - \frac{w}{2}\cos\theta}{t_z - \frac{w}{2}\sin\theta} + \frac{t_x + \frac{w}{2}\cos\theta}{t_z + \frac{w}{2}\sin\theta}\right), \tag{8}$$

$$m_z = \frac{f_v}{2}\left(\frac{h}{t_z - \frac{w}{2}\sin\theta} + \frac{h}{t_z + \frac{w}{2}\sin\theta}\right), \tag{9}$$

$$m_\theta = f_u\left(\frac{t_x + l\,\sin\theta}{t_z - l\,\cos\theta}\right). \tag{10}$$

In order to perform the *Back-projection*, we need to solve the system above for the unknown pose parameters t_x, t_z and θ. A possible approach would be to try to combine these equations analytically, but due to the nonlinearity of the camera model, this approach is likely to result in a solution with ambiguity problems and poor error propagation properties. Instead, we exploit the temporal coherence of the sequence of images through a numerical algorithm that is iterated for successive input images, in order to recover precise values of the pose parameters. We start with Eq. (9), since that is the only equation in the system above that involves only two of the three unknowns: t_z and θ. Instead of trying to solve for both unknowns at the same time, we use the measured value of θ from the previous step of the tracking process in order to get the estimated value for t_z at the current step:

$$t_z^{(i)} = \frac{f_v\,h + \sqrt{(f_v\,h)^2 + \left(m_z\,w\,\sin\theta^{(i-1)}\right)^2}}{2\,m_z}. \tag{11}$$

Now we solve Eq. (8) for the unknown t_x. Of course, the resulting expression still depends on both t_z and θ. The value of t_z in the current tracking step has just been computed according to Eq. (11) and can be used in the recovery of t_x. But the value of θ is still unknown in the current step and thus, it is obtained from the previous step:

$$t_x^{(i)} = \frac{m_x}{f_u}t_z^{(i)} + \frac{w^2\sin\theta^{(i-1)}}{4\,t_z^{(i)}}\left(\frac{m_x}{f_u}\sin\theta^{(i-1)} + \cos\theta^{(i-1)}\right). \tag{12}$$

Finally, Eq. (10) can be solved directly for θ, after the values of t_z and t_x are both known. Initially, we rewrite it as:

$$\sin\theta + \frac{f_u}{m_\theta}\,\cos\theta + \frac{f_u\,t_z}{m_\theta\,l} - \frac{t_x}{l}.$$

Notice that this expression is on the form:

$$\sin\theta + c_1\,\cos\theta + c_2 = 0, \quad \text{with:} \quad c_1 = \frac{f_u}{m_\theta}, \quad c_2 = \frac{f_u\,t_z}{m_\theta\,l} - \frac{t_x}{l}.$$

So, we rename $\sin\theta$ as a new variable and use the trigonometric identity $\cos\theta = \sqrt{1-\sin^2\theta}$ to reduce the equation above to a quadratic form. By checking the two roots of the transformed equation for consistency with the original form, we can determine a unique solution for θ:

$$\theta = \sin^{-1}\left(\frac{-c_2 - c_1\sqrt{c_1^2 - c_2^2 + 1}}{c_1^2 + 1}\right).$$

Substituting back the original expressions renamed as c_1 and c_2 and simplifying the resulting equation, we obtain the final formula for θ:

$$\theta^{(i)} = \sin^{-1}\left(\frac{f_u\,k_2 - m_\theta\sqrt{k_1 - k_2^2}}{k_1}\right), \quad \text{where:} \quad k_1 = f_u^2 + m_\theta^2, \quad k_2 = \frac{m_\theta\,t_z^{(i)} - f_u\,t_x^{(i)}}{l}. \quad (13)$$

Eqs. (11) to (13) allow one to perform pose recovery recursively, using the solution found in the previous step as an initial guess for the unknown pose at the current step. However, we still need an initial guess for θ at the first time that Eqs. (11) and (12) are to be used. Our choice, in this case, is to set $\theta^{(0)} = 0$, reducing the equations mentioned above to:

$$t_z^{(1)} = \frac{f_v\,h}{m_z}, \quad (14)$$

$$t_x^{(1)} = \frac{m_x\,t_z^{(1)}}{f_u}. \quad (15)$$

Notice that this amounts to a weak perspective approximation, since $\theta = 0$ implies that all the four vertices of the target rectangle that is used to recover t_z and t_x are at the same depth with respect to the camera. So, in this sense, our pose recovery algorithm is inspired in the scheme proposed by DeMenthon and Davis [6], because it starts with a weak perspective approximation and then refines the projective model iteratively, in order to recover a fully perspective pose. As we mentioned in Section 2, the basic differences are that we use a much more specialized camera model, with only three DOF (as opposed to six in DeMenthon–Davis's algorithm), and we embed the refinement of the projective model in successive steps of the tracking phase, rather than starting all over from scratch and iterating our algorithm until it converges for each frame. This is a way of exploiting the temporal coherence of the input images to achieve relatively precise pose estimates at low computational cost. Finally, one last difference between our work and DeMenthon–Davis's method is that our initial solution is not completely affine, since the equation for θ takes into

account the distortion caused by the fact that the central circle in the target is located at a different plane than the other four circles. In fact the three–dimensionality of the target is very important: for small variations of heading, the three–dimensional target exhibits first order effects (proportional to the sine of the angle near zero) but a two-dimensional target exhibits only second-order effects proportional to the cosine of the angle near zero.

4 Comparing Our Approach to the VSDF

In order to evaluate the benefits that the use of strong application–specific constraints brings to the formulation introduced in the previous section, we used synthetic data to compare it against a more generic motion recovery tool, known as the Variable State Dimension Filter (VSDF) [23; 24; 25]. Contrary to our technique, the VSDF does not make any prior assumptions about the nature of the rigid motion in the input scenes. We also included in the comparison a simpler pose recovery algorithm that uses weak perspective rather than perspective as its imaging model. This algorithm, that also assumes the existence of a unique ground plane, consists of using Eqs. (14) and (15) for translation recovery; and a formula derived from Eq. (10) — with the approximation $\cos\theta \approx 1$ — for rotation recovery.

The VSDF is a framework for optimal recursive estimation of structure and motion based on the assumption that image measurement errors are independent and Gaussian distributed. In other words, the VSDF allows new measurements to be used to refine, iteratively, estimates for the values of a set of unknown structure and motion parameters, initially obtained through a batch algorithm. The VSDF achieves this goal by linearizing the (typically non–linear) imaging equations relating the measurements to the unknown parameters around the current estimates for the values of these parameters. Different measurement equations corresponding to distinct camera models (such as perspective [24], affine [23] or projective [24]) can be used within this general framework. The VSDF only requires some of the unknown parameters (typically the structure parameters) to be *local*, in the sense that they can only be related to a unique feature and must remain constant across different scenes. The remaining parameters (typically the motion parameters) are assumed to be *global*, in the sense that they are typically related to the whole set of measurements and they may vary across different scenes. No prior assumption is made about the dynamics of these global parameters.

Among the several working modes of the VSDF, we chose one that uses an affine camera model. We ruled out the perspective mode for two reasons: it requires very accurate prior camera calibration (sometimes does not converge at all when the calibration this requirement is not met); and as of the date of publication of this report, the public distribution of the VSDF available within the vision package Horatio [22] supports this mode only in a batch form that is not feasible for real–time usage. The projective mode, on the other hand, is not able to filter out image noise unless at least six features can be tracked, which does not happen in the our set–up.

Since the type of structure recovered by the affine mode of the VSDF is non–metric, the motion recovered with it is not necessarily rigid, contrary to the other two techniques considered in the comparison. However, we do have reasonable prior estimates of the calibration parameters. So, we initially pre–multiply all the image measurements by the inverse

of the estimated calibration matrix (Eq. (3)), so as to remove the intrinsic projective transformation performed by the camera. Then we apply the VSDF to the pre–processed data and recover a 3×4 projection matrix for each frame. If the calibration parameters remain fixed and the motion is truly rigid, the first three columns of this matrix are ideally equivalent to a matrix describing the target–camera rotation, up to an arbitrary scaling factor; and the last column is ideally equivalent to the target–camera translation, up to the same scaling factor. In practice, this does not happen because there are some errors in the calibration and the VSDF can't always eliminate all the noise from the input images. So, we post–process the 3×4 matrix returned by the VSDF in order to ensure the orthonormality of its first three columns and then use the result to obtain the three desired planar motion parameters.

4.1 Experimental Methodology

In order to be able to perform a reasonably realistic simulation of our intended application, we initially measured the maximum linear and angular speeds that can be attained by the vehicles used, when fully loaded with equipment. These speeds were found to be approximately equal to 60 inches/s and 90 degrees/s, respectively (or 2 inches/frame and 3 degrees/frame, at a frame rate of 30 Hz). Since our mobile platforms are propelled by two independent motors, one for each active wheel, rotations are obtained by applying a different torque on each wheel and thus it is not possible to reach both top linear and angular speeds at the same time.

In our simulations, we assume that the linear and angular speeds at a single instant in time are linearly related, so that the increasing the former induces a decrease in the latter and vice–versa. We also rule out the cases in which the motion of the leading vehicle with respect to the ground is entirely or almost entirely rotational, because the non–holonomic constraints in the trailing platform usually make the convoying task physically infeasible in such situations. Having this in mind, we assume that, at any given frame t, the linear and angular velocities of the leading platform are given by:

$$\begin{aligned} v_t &= (2 - \lambda_t) \text{ inches/frame}, \\ \omega_t &= 2\lambda_t \text{ degrees/frame}, \end{aligned}$$

where λ_t is a scalar in the interval $[0, 1)$, used to quantify "how curved" is the trajectory at frame t.

Each motion sequence used in the experiments presented in this section consists of 1,800 consecutive frames (generated at a sampling rate of 30 Hz). The motion of the leading platform is modeled as a sequence of multi–frame maneuvers, such that during each maneuver the value of the parameter λ, drawn from a uniform distribution over its domain $[0, 1)$, is kept constant. The duration of each maneuver is drawn from a uniform distribution on the interval $[15, 30)$ frames. The motion of the trailing platform is modeled as being generated by an "ideal" controller, so that the it repeats the same sequence of poses (positions and orientations) of the leading platform with respect to the ground plane, with a delay of Δt frames — where Δt is a simulation parameter that is varied in some experiments and kept fixed in others.

Several types of imprecisions occurring in practice are taken into account in the simulation. The structure of the target placed in the posterior part of the leading platform was measured with imprecisions of about 0.1 inches in the distances between pairs of features. So, in the generation of each motion sequence, Gaussian noise with zero mean and standard deviation equal to 0.1 inches is added to the measured structure shown in Fig. 1, but the pose recovery (using any of the three alternatives under consideration) is always performed with the original measurements.

The imaging transformation is modeled approximately according to Eq. (4). However, in practice it is impossible to align the optical axis exactly with respect to the ground plane, as assumed in Eq. (4). So, at each frame t, the "ideal" image generated by that equation is perturbed by a random camera rotation whose axis is uniformly sampled in a sphere with unit norm and whose magnitude is obtained from a Gaussian distribution with zero mean and standard deviation controlled by a parameter m_a. Different values of m_a are used to model varying degrees of misalignment of the camera with respect to the ground plane. Notice that in the type of domain that we have in mind (navigation in indoor environments such as rooms and hallways) the imprecision caused by this type of misalignment largely subsumes that caused by violations on the assumption that the ground plane is unique.

Furthermore, in practice it is impossible to determine the values of the intrinsic parameters f_u, f_v, u_0 and v_0 exactly. In all the experiments the *true* values of these parameters were assumed to be equal to 320, 240, 160 and 120 pixels, respectively. However, independent Gaussian biases (fixed throughout each sequence of 1,800 frames) with zero mean and standard deviation equal to $c_b f_u$ (for a certain constant c_b) are added to the values of f_u and u_0. Similarly, biases with standard deviation equal to $c_b f_v$ are added to f_v and v_0. Different values chosen for the multiplicative constant c_b model varying degrees of calibration inaccuracy, as we will describe later. Finally, independent Gaussian noise with zero mean and standard deviation controlled by a parameter i_n is added to the image coordinates of each feature, after the application of the imaging transformation. This noise is used to model both the effects of spatial quantization and errors in the localization of the features in the image plane.

4.2 Accuracy in the General Case

Initially, we ran some experiments to determine which of the three methods under consideration is the most accurate, in "general" cases. Throughout the experiments reported in this subsection, the standard deviation of the misalignment angle (m_a), the multiplicative calibration bias (c_b) and the level of image noise (i_n) are kept fixed at 2.0 degrees, 1% and 0.5 pixels, respectively. These values approximately agree with our practical implementation described in Section 6. So, during this initial experiment, only the simulation parameter Δt is varied. A different sequence with 1,800 frames was generated for each of the following values of Δt: 30, 45, 60, 75 and 90 frames. The reason why these particular values were chosen is that, with less than 30 frames of delay, the distance between the two vehicles often becomes so small that several target features exit the field of view of the trailing vehicle's camera. On the other hand, with more than 90 frames of delay, the difference in orientation between the two vehicles becomes so big that the target often becomes occluded by other parts leading vehicle.

For each frame on each sequence, the values of the three planar pose parameters (t_x, t_z and θ) are estimated with each of the techniques available. The averages and standard deviations (per sequence) of the absolute differences between these estimates and the true values of the parameters, for each method, are plotted in Fig. 2. In this and all the other graphs exhibited in this section, solid lines, dash–dotted lines and dashed lines are used to represent, respectively, results obtained with our planar perspective pose recovery scheme, with a weak–perspective pose recovery algorithm also based on the assumption of planar motion and with the affine mode of the Variable State Dimension Filter (VSDF).

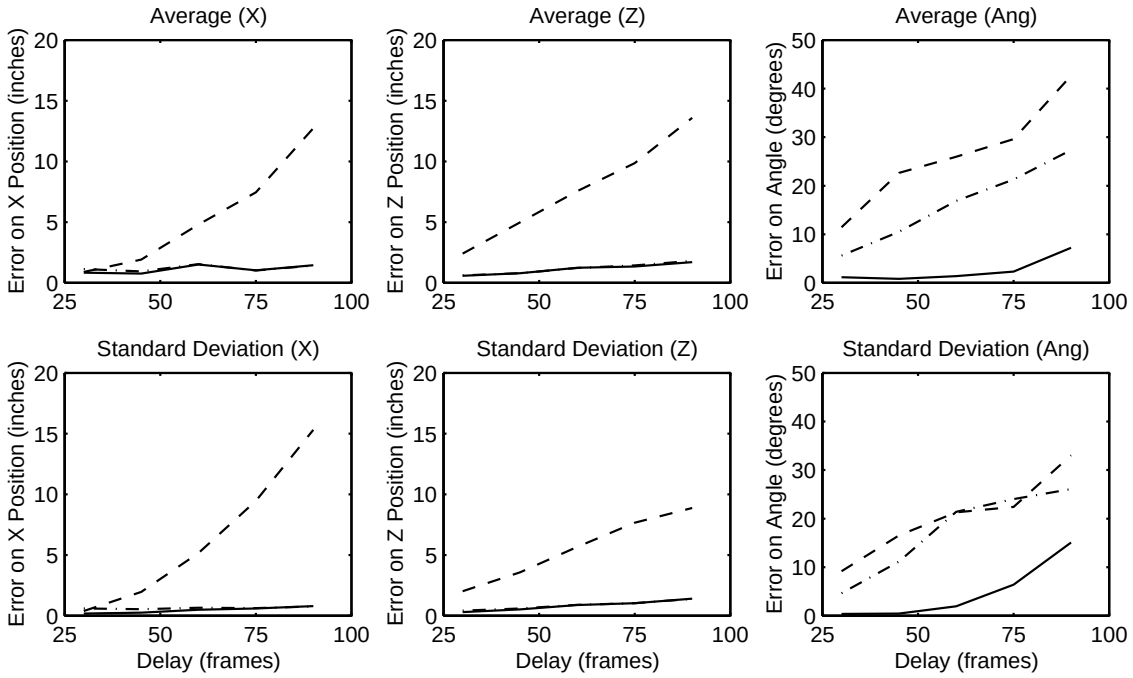

Figure 2: Sensitivity in errors of the estimated planar pose parameters with respect to the delay between the leading and the trailing platforms. Solid, dash–dotted and dashed lines represent, respectively, planar perspective pose recovery, planar weak–perspective pose recovery and affine pose recovery with the VSDF.

It can be observed that both schemes based on the assumption of planar motion (perspective and weak–perspective) are roughly equivalent for the estimation of the translational parameters t_x and t_z (the solid and dash–dotted lines are almost indistinguishable in the two first columns of Fig. 2). However, the perspective algorithm that we propose is clearly superior for rotation estimation. Furthermore, both planar algorithms are clearly more accurate than the affine VSDF estimation.

4.3 Sensitivity with Respect to Biases and Noise

Of course, one might object that the results reported in the previous subsection are not exactly surprising, since by fixing the simulation parameters m_a, c_b and i_n, we established

a priori a certain "degree of validity" for the crucial assumptions that make our proposed method more accurate than the competing techniques. So, we also ran some experiments in which Δt was kept fixed and the other parameters were changed, in order to determine the sensitivity of the different techniques with respect to different violations of the task–specific assumptions that we impose.

The value $\Delta t = 45$ frames was chosen because with it the length of the path that the trailing platform has to traverse to reach the current position of the leading platform, at any given instant of time, is constrained to the interval $[45, 90)$ inches, which agrees with the real–world experiments reported in Section 6. In each of the remaining experiments described in this section, one of the three parameters m_a, c_b and i_n is varied and the other two are kept fixed at half of their values in the first experiment (so as to make the impact of the varying parameter more clear).

Initially, we evaluated how well the three alternative pose recovery schemes perform if the assumption of alignment between camera and ground plane is violated. In this experiment, m_a is varied exponentially between 0.25 and 32 degrees. The averages and standard deviations of the absolute errors in the individual parameters, shown in Fig. 3, indicate that our planar perspective technique is the one whose accuracy is most sensitive with respect to variations in this simulation parameter.

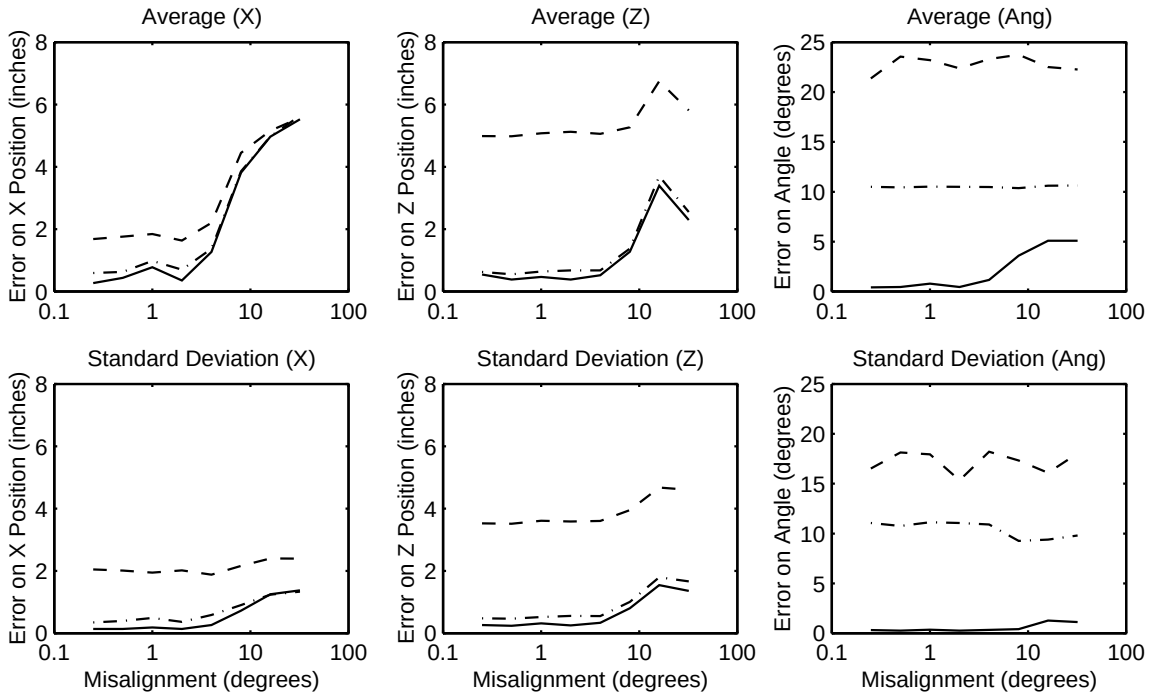

Figure 3: Sensitivity in errors of the estimated planar pose parameters with respect to the standard deviation in the angle between the optical axis and the ground plane. Solid, dash–dotted and dashed lines represent, respectively, planar perspective pose recovery, planar weak–perspective pose recovery and affine pose recovery with the VSDF.

This is quite natural, since the VSDF does not rely on this assumption of alignment at

all and the precision of the weak–perspective estimation is already bound by the fact that the perspective distortion in the input scenes is ignored. However, even with gross alignment errors (32 degrees), the technique that we propose remains significantly more precise than the competing alternatives. With misalignments of up to 4 degrees, in particular, it yields very precise estimates for all three pose parameters, unlike the other two methods.

We also ran a similar experiment in which c_b is exponentially varied between 0.125% and 8% (and the remaining simulation parameters are kept constant), in order to check the sensitivity of the techniques in study with respect to the precision of the camera calibration. The results (based on the same error metrics as the previous experiments) are displayed in Fig. 4.

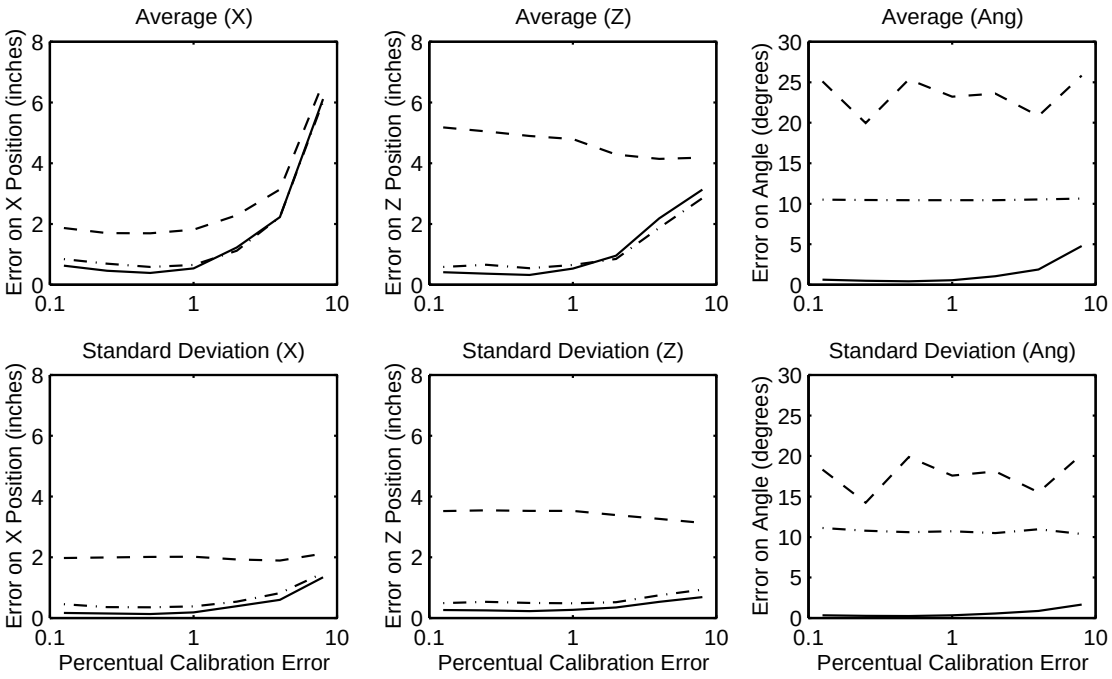

Figure 4: Sensitivity in errors of the estimated planar pose parameters with respect to the level of bias in the camera's intrinsic parameters. Solid, dash–dotted and dashed lines represent, respectively, planar perspective pose recovery, planar weak–perspective pose recovery and affine pose recovery with the VSDF.

These results are qualitatively very similar to those obtained in the previous experiments, or in other words: our proposed planar perspective algorithm yields again more accurate results than the competing techniques, especially in the recovery of the rotation about the normal to the ground plane (θ). As the calibration bias is increased, this accuracy gap is gradually reduced, but even with quite imprecise calibration (errors of the order of 8% of the focal length), there is still a significant advantage in using our technique. Below the threshold of 1% imprecision, our technique yields very accurate recovery.

Finally, we checked the behavior of the different techniques when the image noise (i_n) is increased exponentially from 0.125 pixels to 8 pixels. The results (with the usual error

metrics) are plotted in Fig. 5. Once more, the superiority of our suggested approach is clear. For very noisy images (8 pixels of standard deviation in positional errors on a 320 $\times$ 240 image plane), the accuracy of the planar techniques for translation recovery becomes slightly worse than that of the VSDF. However, even in this case the planar algorithms, especially the perspective one, are still much more accurate for rotation recovery. An ideal noise threshold to achieve very precise recovery with our technique is 0.5 pixels.

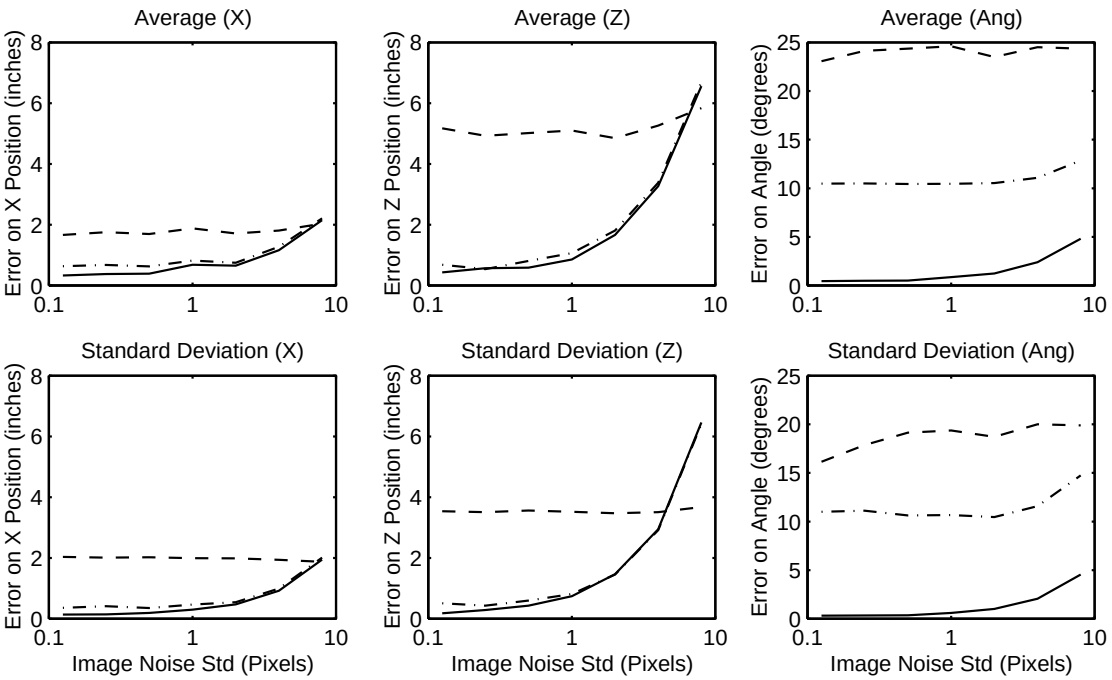

Figure 5: Sensitivity in errors of the estimated planar pose parameters with respect to the standard deviation of the additive Gaussian noise on the image plane. Solid, dash–dotted and dashed lines represent, respectively, planar perspective pose recovery, planar weak–perspective pose recovery and affine pose recovery with the VSDF.

4.4 Discussion

So, to summarize the results presented so far: the planar perspective technique that we suggest seems to be much more accurate than the affine mode of the Variable State Dimension Filter (VSDF) in the specific domain of our intended application, both for translation and for rotation recovery. This is not surprising, since the VSDF is a much more generic technique that does not exploit domain–specific constraints to achieve greater stability. A more interesting result is the fact that this superiority of our approach can still be verified even when some of its domain–specific assumptions are partially violated. In addition, when compared to a simpler approach that does not take into account the effects of perspective distortion, our technique yields significantly more accurate rotation estimates. The two approaches are roughly equivalent for translation estimation.

We also measured averages and standard deviations of execution times for all the experiments performed (the simulations were run in the same twin Pentium processors used in the real–world implementation and all the code was compiled at the optimization level defined by the flag -O2). The planar perspective and planar weak–perspective techniques have minimal computational requirements, with average execution times per frame of 19 and 22 microseconds, respectively, and standard deviations of 3 microseconds (in both cases). The affine mode of the VSDF, on the other hand, can barely be used in real–time since it takes on average 51 milliseconds per frame to execute, with a standard deviation of 3 milliseconds. In practice, since the same computational resources used for pose estimation are also support low–level vision and control in our implementation, the use of the VSDF would constrain the frame rate of our system to something in the order of 10 Hz, at best.

5 Efficient Low Level Image Processing

The image acquisition is performed with a Matrox Meteor frame grabber. In order to achieve maximum efficiency, this device is used in a mode that reads the images directly to the memory physically addressed by the Pentium processors, using multiple preallocated buffers to store successive frames. This way, the digitized images can be processed directly in the memory location where they are originally stored, while the following frames are written to different locations.

The initial step of the low–level image processing is the construction of a multi-resolution pyramid. In the current implementation, we start with digitized images of size 180×280. On each of the lower resolution levels, each image is obtained by convolving the corresponding image in the immediately higher resolution level with a Gaussian kernel and subsampling by a factor of two. This operation was implemented in a very careful way, in order to guarantee the desired real–time feasibility. Instead of using some general convolution routine that works with arbitrary kernels, we implemented a hand–optimized function that convolves images with a specific 3×3 blurring kernel, corresponding to a bivariate Gaussian distribution with standard deviation approximately equal to 0.8493 on both axes. The use of a single predefined kernel eliminates the need to keep its elements either in specially–allocated registers or in memory, speeding up the critical inner loop of the convolution. The criterion used in the choice of the particular kernel shown in Fig. 6 is the fact that its elements are all powers of two, and thus it can be implemented with integer additions only. With a careful subexpression factorization, the resulting convolution and subsampling can be implemented with only 1.5 memory accesses with pointer increment, 2.25 integer additions, 0.25 pointer comparisons and 0.25 shift–right operations per element of the original image, on average.

The next step is the segmentation of the target in the image. In order to obtain some robustness with respect to variations in the illumination and in the background of the scene, we use a target composed of black circles printed on white paper and perform a histogram analysis to determine an ideal threshold to binarize the monochromatic images grabbed by the Matrox Meteor in the trailing robot, so that the black dots can be told apart from the white paper. For efficiency purposes, the grey–level frequency information needed to

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 6: Discrete Gaussian kernel used in the subsampling process; $\sigma \approx 0.8493$

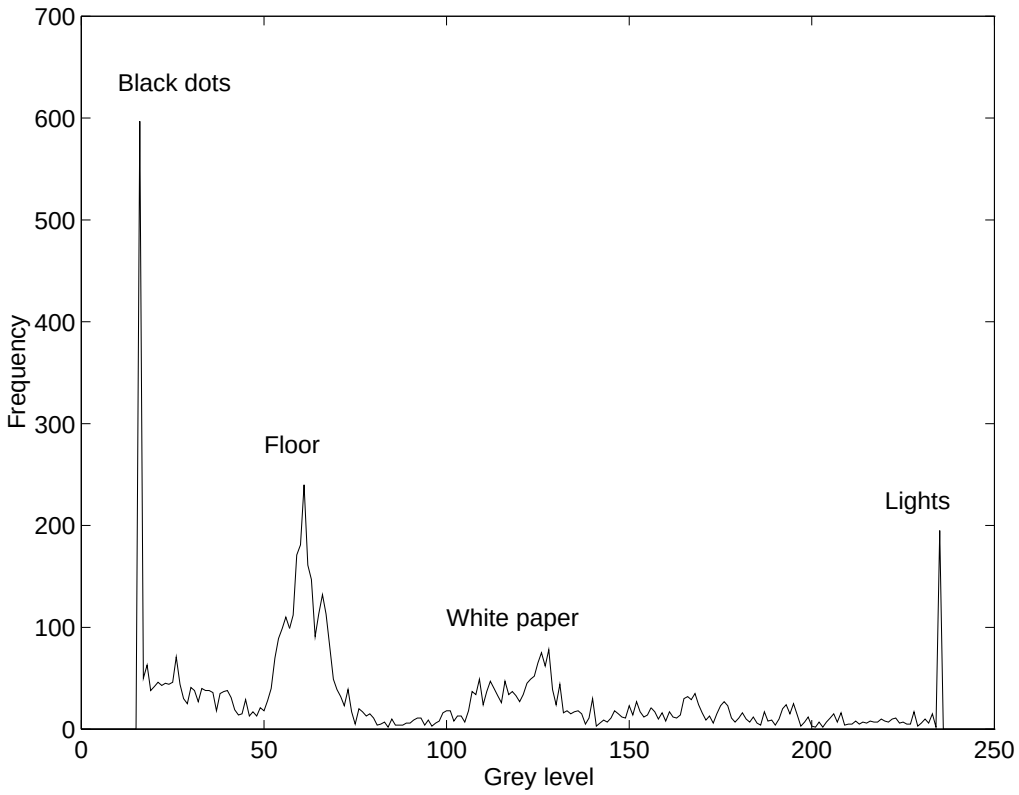

Figure 7: Grey level histogram in a scene with the target at about 10 feet from the camera.

generate the histograms is gathered on–the–fly, during the subsampling process, at the extra cost of 1 memory access with pointer increment, per pixel.

A typical low–resolution grey–level histogram for an image grabbed in the environment used in the tests, with the chair relatively far away from the camera (about 10 feet), is shown in Fig. 7. It was verified that there is a satisfactory contrast between the black dots (corresponding to the darkest end of the histogram) and the white paper (corresponding to a peak of intermediate intensity), for a wide range of illumination conditions. Unfortunately, it was also verified that in general there are many different parts of the scene background that have roughly the same intensity than either one of these two parts of the target.

In order to determine the ideal thresholding point, we initially smooth the histogram, by convolving it with an unidimensional discrete low–pass (*average*) filter with a kernel of size eight. Then we scan the histogram from the darkest to the lightest grey level, looking for the first valley that appears after a peak, in order to try to separate the black dots from

anything else but a few black spots in the scene. A peak is defined as a grey level whose frequency is strictly higher than the following eight levels in the histogram at least, and a valley is defined as a grey level that is at least eight levels apart from the previous peak and has frequency strictly lower than the following eight levels. Actually, for improved efficiency, the smoothing is performed on–the–fly, during the scanning process, and it is interrupted as soon as the desired valley is found. This valley is then used as the binarization threshold.

The next step is to detect and label all the connected regions of low intensity (according to the selected threshold) in the image. This is done using the local blob coloring algorithm described in Ballard and Brown [4]. Again, the criterion that determined the selection of this technique was its efficiency, since it scans the entire input image just once, from the left to the right and from the top to the bottom.

Initially, this algorithm is used to detect all the dark regions in a level of low resolution in the pyramid. In this phase, in addition to labeling all the connected components in the image, we also compute, on–the–fly, their bounding boxes, centroids and masses. The dark regions detected in the image are compared against the appearances predicted for the target's black circles by the tracker that we describe in Section 3. For each predicted appearance (converted to the appropriate level of resolution), we initially label as matching candidates all the detected regions with similar mass and aspect ratio. Among these, the detected region whose centroid is closest to the position predicted by the tracker is selected as the final match for the corresponding circle in the target.

The selected bounding boxes are then converted to a level of high resolution, and the blob coloring algorithm is used on each resulting window, in order to refine the precision of the estimates for the centroids in the image. The resulting image positions are used as inputs to the tracker, that recovers the 3-D pose of the target, predicts how this pose will evolve over time, and then reprojects the 3-D predictions into the 2-D image plane, in order to calculate new predicted appearances for the black dots, which are used on the next step of the low level digital image processing.

6 Visual Control and Real–World Experiments

In addition to the tracking of the leading platform, the problem of smart convoying also requires the motion of the trailing robot to be properly controlled, so that the target to be followed never disappears from its field–of–view (or alternatively, it is reacquired whenever it disappears). In our system, this control is based on the 30 Hz error signal corresponding to the values recovered for t_x and t_z (θ is used only in the prediction of the appearance of the target on the next frame), and also on odometry data.

The reason why the use of odometry is important is that the true dynamics of our mobile platforms is quite complex. The two motors are directly driven by PCM signals generated by a proprietary controller which interprets the output of either an Onset M68332 micro–controller, or a joystick (for manual control). So, all that the M68332 sees is an abstraction of the motors. It can issue commands to change the velocity or the steering direction of the platform, but it can not control the wheels individually. The true effects of the issued commands depend on a number of factors that are difficult to be modeled exactly, such as

differences in the calibration of the motor torques, differences in the pressure of the two tires and the relative orientations between the two wheels and a set of three passive casters that are used to create a stable set of contact points between the platform and the ground. Because of these imprecisions, an open–loop sequence of "go–straight" commands issued by the M68332 can actually make the platform move along a curved trajectory, for instance.

In order to overcome this difficulty, we use the data obtained by two bi–directional hollow shaft encoders [5] (one for each wheel) to close the loop so as to guarantee that the angular velocities on the two wheel axes actually correspond to the desired motion patterns. Each encoder generates two square waves, with a ninety–degree phase lag. These output signals are then decoded by customized hardware and used to decrement or increment a specific register in the M68332 each time the corresponding wheel rotates back or forth, respectively, by an arc equivalent to the precision of the shaft encoder. Variations in these registers are then compared with the desired values for the angular velocities of the wheels, so as to create error signals that are fed back to the M68332 controller.

On the other hand, the M68332 by itself does not provide enough computational power to process high–bandwidth signals, such as visual data, in real–time. So, we augmented the system with twin 133 MHz Pentium processors, that are used to process the digitized image sequences so as to extract the desired visual measurements and estimate motion (as explained in Sections 4 and 5).

This set–up naturally leads to a two–level control strategy. In the Pentium processors, a higher level composed by two low–frequency (30 Hz) PID controllers (with *proportional*, *integral* and *derivative* gains empirically set) converts the t_z and t_x signals, respectively, into ideal speed and steering commands for the platform. The goal of one of this controllers is to keep t_z equal to convenient predefined value, while the other aims to keeping t_x equal to zero. These commands are then passed down to a lower level, that runs at 100 Hz in the M68332. This level also uses two PID controllers with empirically–set gains. One of them uses differences in the rates of change of the tick counters associated with the two wheels to stabilize steering, while the other uses the average of these rates of change to stabilize the velocity. So, from the point–of–view of the higher level, this lower level creates an abstraction for the dynamics of the mobile platform that is much simpler than reality, since the unpredictable effects of several imprecisions are compensated through the use of odometry. The communication between the Pentium board and the M68332 is performed trough a specialized serial protocol whose design and implementation are described in [5]. A block diagram illustrating our control approach is exhibited in Fig. 8.

In order to evaluate our approach for convoying, we ran some experiments with real data. Basically, we used the methodology described so far to try to make one of our two identical mobile platforms follow the other (manually driven) at a roughly constant distance of about 5 feet. These experiments were performed in indoor environments with varying lighting conditions. It was verified that the controller performs quite well in the sense that it manages to keep the leading platform in view, actually keeps the distance roughly constant, tolerates changes in lighting conditions, and can reliably track turns of up to 180 degrees without losing target features, as illustrated by the sequence of Fig. 9.

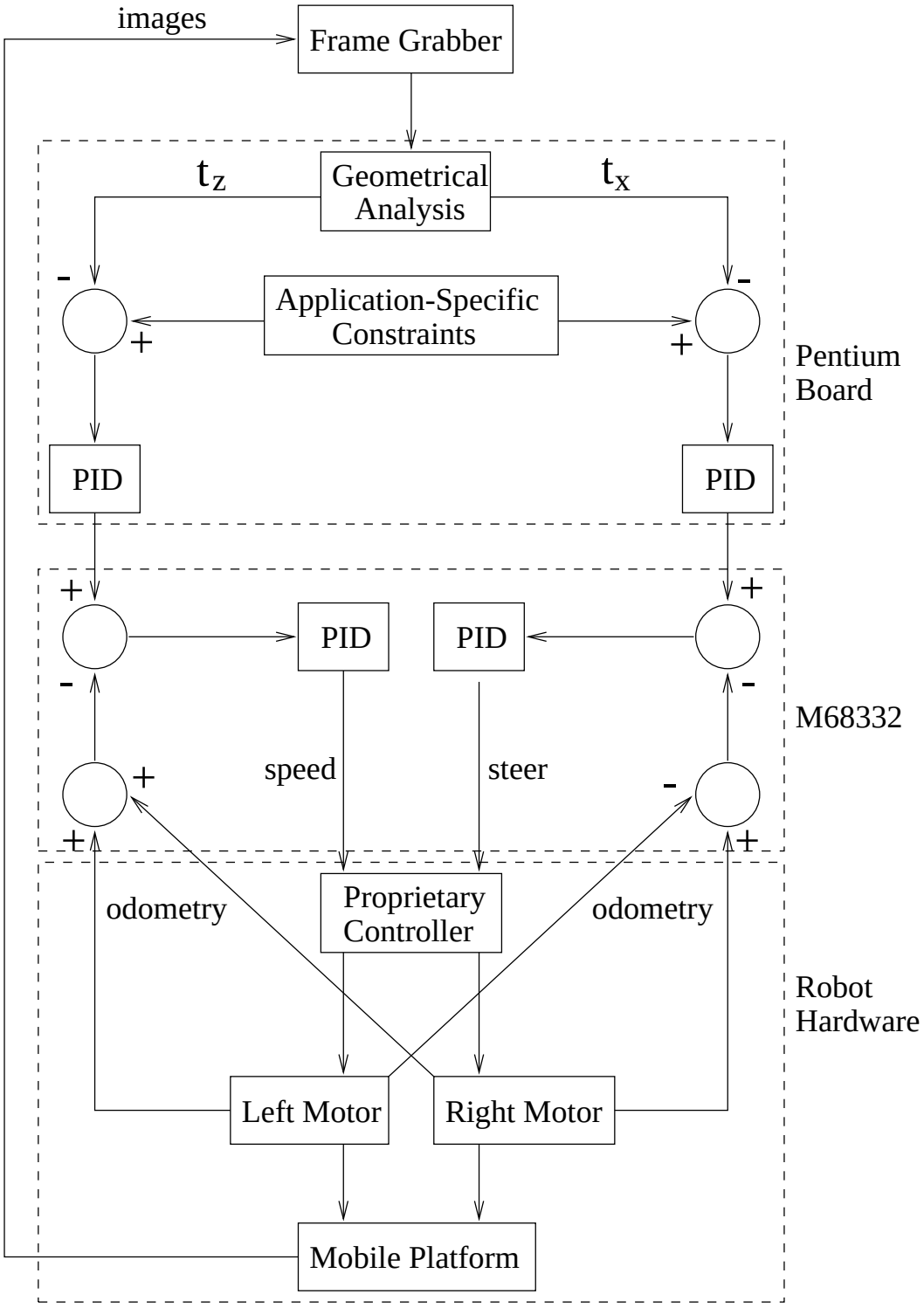

Figure 8: Block diagram showing the main components of our tracking–and–control system.

7 Conclusions

This results support our position that by putting together traditional computer vision techniques carefully customized to the to meet application–specific needs, it is possible to tackle challenging problems with low–cost off–the–shelf hardware. In the specific case of convoying, we have shown, in a careful evaluation with synthetic data, that specialized motion analysis algorithms that take into account domain–specific constraints such as the existence of a unique ground plane often yield more accurate and stable results than totally generic techniques, even when these assumptions are only partially met. We have also shown how careful engeneering of the problem can eliminate the need for expensive vectorial hardware in low–level vision. Finally, we suggested a two–level approach for control, in which high–frequency odometry data can be used to stabilize visual control.

This paper describes work that is still in progress and we stress the fact that some of the issues raised here need further investigation. In our opinion one of the most interesting directions in which this work must be continued is with a deeper investigation of which is the best control strategy for the application at hand. Our current controller assumes "off road" conditions: it is permissible always to head directly at the lead vehicle, thus not necessarily following its path. Its goal is roughly to keep the target centered in the field of view and correctly sized. If vehicles must stay "on road", then what is desired is that:

$$\mathbf{s}_f^{(t)} = \mathbf{s}_l^{(t-\Delta t)}, \tag{16}$$

where $\mathbf{s}_f$ and $\mathbf{s}_l$ are the state vectors of the follower and leader, respectively, and Δt is some desired delay. In words, the follower should re–trace the trajectory of the leader precisely. The criterion that the follower maintain a constant distance from the lead vehicle and stay on its path can be disastrous since it could, for example, induce the follower to take a sharp corner too fast while trying to keep up with a leader who speeds up once through the corner. Eq. (16) raises a number of interesting issues on itself: state estimation of the leader's heading (global steering angle, say) as well as speed (or accelerations) are ultimately needed, to be duplicated for local control. Vision becomes harder since the follower cannot always aim itself at the leader. The desired trajectory is known, which turns the problem into one that can perhaps more usefully be related to optimal control than to simple feedback control.

Figure 9: Twelve frames from a sequence that shows a completely autonomous mobile platform following a manually–driven platform around a cluster of tables. The temporal sequence of the frames corresponds to a row–major order.

References

[1] M. A. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE Trans. PAMI*, 17(5):534–538, 1995.

[2] T. D. Alter. 3-D pose from 3 points using weak–perspective. *IEEE Trans. PAMI*, 16(8):802–808, 1994.

[3] H. Araujo, R. L. Carceroni, and C. M. Brown. A fully projective formulation for Lowe's tracking algorithm. Technical Report 641, U. Rochester Comp. Sci. Dept., 1996.

[4] D. H. Ballard and C. M. Brown. *Computer Vision.* Prentice–Hall, 1982.

[5] J. D. Bayliss, , C. M. Brown, R. L. Carceroni, C. K. Eveland, C. Harman, A. Singhal, and M. Van Wie. Mobile robotics 1997. Technical Report 661, U. Rochester Comp. Sci. Dept., 1997.

[6] D. F. DeMenthon and L. S. Davis. Model–based object pose in 25 lines of code. *Int. J. of Comp. Vis.*, 15:123–141, 1995.

[7] M. Dhome, M. Richetin, J-T. Lapresté, and G. Rives. 3-D pose from 3 points using weak–perspective. *IEEE Trans. PAMI*, 11(12):1265–1278, 1989.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analisys and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[9] D. B. Gennery. Visual tracking of known three–dimensional objects. *Int. J. Comp. Vis.*, 7(3):243–270, 1992.

[10] R. M. Haralick and C. Lee. Analysis and solutions of the three point perspective pose estimation problem. In *Proc. IEEE Conf. CVPR*, pages 592–598, 1991.

[11] R. Horaud, S. Christy, F. Dornaika, and B. Lamiroy. Object pose: Links between paraperspective and perspective. In *Proc. Int. Conf. Comp. Vis.*, pages 426–433, 1995.

[12] R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle. An analytic solution for the perspective 4-point problem. *CVGIP*, 47:33–44, 1989.

[13] D. P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. Int. Conf. on Comp. Vis.*, pages 102–111, 1987.

[14] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. J. of Comp. Vis.*, 5(2):195–212, 1990.

[15] M. Ishii, S. Sakane, M. Kakikura, and Y. Mikami. A 3-D sensor system for teaching robot paths and environments. *Int. J. Rob. Res.*, 6(2):45–59, 1987.

[16] S. Linnainmaa, D. Harwood, and L. S. Davis. Pose determination of a three–dimensional object using triangle pairs. *IEEE Trans. PAMI*, 10(5):634–647, 1988.

[17] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-D to 3-D line and point correspondences. *IEEE Trans. PAMI*, 12(1):28–37, 1990.

[18] D. G. Lowe. Solving for the parameters of object models from image descriptions. In *Proc. ARPA IU Workshop*, pages 121–127, 1980.

[19] D. G. Lowe. Three–dimensional object recognition from single two–dimensional images. *Artif. Intell.*, 31(3):355–395, 1987.

[20] D. G. Lowe. Fitting parameterized three–dimensional models to images. *IEEE Trans. PAMI*, 13(5):441–450, 1991.

[21] C. B. Madsen. Viewpoint variation in the noise sensitivity of pose estimation. In *Proc. IEEE Conf. CVPR*, pages 41–46, 1996.

[22] P. F. McLauchlan. Horatio: libraries for vision applications. Technical report, Dept. Engin. Sci. U. Oxford, 1992.

[23] P. F. McLauchlan and D. W. Murray. Recursive affine structure and motion from image sequences. In *Proc. European Conf. on Comp. Vis.*, pages 217–224, 1994.

[24] P. F. McLauchlan and D. W. Murray. A unifying framework for structure and motion recovery from image sequences. In *Proc. IEEE Int. Conf. on Comp. Vis.*, pages 314–320, 1995.

[25] P. F. McLauchlan and D. W. Murray. Active camera calibration for a head–eye platform using the variable state–dimension filter. *IEEE Trans. PAMI*, 18(1):15–21, 1996.

[26] N. Navab and O. Faugeras. Monocular pose determination from lines: Critical sets and maximum number of solutions. In *Proc. IEEE Conf. CVPR*, pages 254–260, 1993.

[27] D. Oberkampf, D. F. DeMenthon, and L. S. Davis. Iterative pose estimation using coplanar feature points. *Comp. Vis. Image Understanding*, 63(3):495–511, 1996.

[28] T. Q. Phong, R. Horaud, and P. D. Tao. Object pose from 2-D to 3-D point and line correspondences. *Int. J. Comp. Vis.*, 15:225–243, 1995.

[29] T. Shakunaga and H. Kaneko. Perspective angle transform: Principle of shape from angles. *Int. J. Comp. Vis.*, 3:239–254, 1989.

[30] C. Wiles and M. Brady. Ground plane motion camera models. In *Proc. European Conf. on Comp. Vis.*, volume 2, pages 238–247, 1996.

[31] A. D. Worrall, K. D. Baker, and G. D. Sullivan. Model based perspective inversion. *Image Vis. Comp.*, 7(1):17–23, 1989.

[32] Y. Wu, S. S. Iyengar, R. Jain, and S. Bose. A new generalized computational framework for finding object orientation using perspective trihedral angle constraint. *IEEE Trans. PAMI*, 16(10):961–975, 1994.

[33] J. S.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Trans. Rob. Aut.*, 5(2):129–142, 1989.