Natural Feature Tracking for Augmented Reality

Ulrich Neumann and Suya You

Abstract— **Natural scene features stabilize and extend the tracking range of augmented reality (AR) pose-tracking systems. We develop robust computer vision methods to detect and track natural features in video images. Point and region features are automatically and adaptively selected for properties that lead to robust tracking. A multistage tracking algorithm produces accurate motion estimates, and the entire system operates in a closed-loop that stabilizes its performance and accuracy. We present demonstrations of the benefits of using tracked natural features for AR applications that illustrate direct scene annotation, pose stabilization, and extendible tracking range. Our system represents a step toward integrating vision with graphics to produce robust wide-area augmented realities.**

Index Terms— **Motion estimation, natural feature tracking, optical flow, (3-VAR) augmented reality.**

I. Introduction

A. Purpose and Motivation

AUGMENTED reality (AR) is an advanced technology for enhancing or augmenting a person's view of the real world with computer generated graphics. Enhancements could include label annotations, virtual object overlays, or shading modifications. An enhanced view of the real world also offers a compelling technology for navigating and working in the real world. The AR metaphor of displaying information in the spatial context of the real world has a wide range of potential applications in multimedia computing and human-computer interaction [2], [5], [7], [21], [24], [26].

Maintaining accurate registration between real and computer generated objects is one of the most critical requirements for creating an augmented reality. As the user moves his or her head and viewpoint, the computer-generated objects must remain aligned with the three-dimensional (3-D) locations and orientations of real objects. Alignment is dependent on tracking (or measuring) the real-world viewing pose accurately. The viewing pose is a six-degree of freedom (6DOF) measurement: three degrees of freedom for position and three for orientation. The tracked viewing pose defines the projection of 3-D graphics into the real-world image so tracking accuracy determines the accuracy of alignment.

General tracking technologies include mechanical arms and linkages: accelerometers and gyroscopes, magnetic fields, radio frequency signals, and acoustics [9], [10], [17]. Tracking measurements are subject to signal noise, degradation with distance, and interference sources. Active tracking systems require calibrated sensors and signal sources in a prepared and calibrated environment [2], [10], [22]. Among passive tracking approaches, computer vision methods can determine pose as well as detect, measure, and reduce posetracking errors derived by other technologies [15], [19], [21], [22], [24]. The combined abilities to both track pose and manage residual errors are unique to vision-based approaches. Vision methods offer a potential for accurate, passive, and low-cost pose tracking. However, they suffer from a notorious lack of robustness. This paper presents our efforts at addressing some of the robustness issues through the detection and tracking of natural features in video images.

The term "tracking" is in common use for describing both 6DOF pose measurement and 2-D feature correspondence in image sequences. We use the term for both purposes in this paper and clarify its meaning by context.

B. Optical Tracking in Augmented Reality

Optical tracking systems often rely upon easily detected artificial features (fiducials) or active light sources (beacons) in proximity to the annotated object(s). The positions of three or more known features in an image determine the viewing pose relative to the observed features [27]. These approaches are applied in many AR applications prototypes [13], [19], [20], [21], [22], [24]. Since the tracking measurements are made with the same camera used to view the scene, the measurement error is minimized for the view direction and scaled relative to the size of the object(s) in the image [2], [20]. These tracking methods require that scene images contain natural or intentionally placed features (fiducials) whose positions are known *a priori*. The dependence upon known feature positions inherently limits a vision-based pose tracking system in several ways.

- Operating regions are limited to areas that offer unobstructed views of at least three known features.
- The stability of the pose estimate diminishes with fewer visible features.
- Known features do not necessarily correspond to the desired points or regions of annotation.

The work presented in this paper take a step toward alleviating the above limitations by making use of natural features, with *a priori unknown* positions. The use of such natural features in AR pose-tracking systems is novel, and we demonstrate its utility.

We define natural feature tracking as computing the motion of a point, locus of points, or region in a scene. These feature

Manuscript received August 19, 1998; revised November 20, 1998. This work was supported by the Defense Advanced Research Project Agency (DARPA) "Geospatial Registration of Information for Dismounted Soldiers" and an NSF CAREER Grant. The associate editor coordinating the review of this paper and approving it for publication was Dr. Chung-Sheng Li.

The authors are with the Computer Science Department, Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089-0781 USA (e-mail: uneumann@graphics.usc.edu; suyay@graphics.usc.edu).

Publisher Item Identifier S 1520-9210(99)01585-0.

classes correspond, respectively, to zero-dimensional (0-D), one-dimensional (1-D), and two-dimensional (2-D) subsets of an image. A 1-D locus of points arises from an edge or silhouette, which can vary abruptly with pose and whose motion along the edge is ambiguous. Since our goal is to estimate the motion of a camera from 2-D feature motions, we limit ourselves to 0-D points and 2-D regions as the feature classes to track.

C. Approach

We develop an architecture for robust tracking of naturally occurring features in unprepared environments and demonstrate how such tracking enhances vision-based AR systems. The architecture integrates three functions—feature selection, motion estimation, and evaluation in a closed-loop cooperative manner that achieves robust 2-D tracking. The main points are summarized in two categories:

Natural feature tracking

—natural feature (points and regions) detection and selection

—multi-stage motion estimation integrating point and region tracking

—evaluation feedback for stabilized detection and tracking

AR applications

—direct annotation of 2-D image sequences

—extendible tracking ranges

—pose stabilization against occlusions and noise.

D. Paper Organization

Section II presents an overview of the closed-loop motion tracking architecture. Section III describes the adaptive feature selection and detection strategy used for identifying the most reliable 0-D features (points) and 2-D features (regions). Section IV describes the integrated point and region tracking method and the closed-loop evaluation feedback. Sections V and VI present test results that illustrate the advantages of our approach and example AR applications. We conclude with remarks and discussions of future work in Section VII.

II. Closed-Loop Motion Tracking Architecture

Fig. 1 depicts the overall tracking system architecture. It integrates three main functions—feature selection, feature tracking, and evaluation feedback, in a closed-loop cooperative manner.

The feature selection stage identifies 0-D and 2-D features (points and regions) with characteristics that promote stable tracking. The selection criteria also include dynamic evaluations fed back from the feature tracking stage. The tracking stage uses multiscale optical-flow for region tracking and a multiscale correlation-peak search for point tracking. Region and point tracking results are fit to an affine motion model, and an evaluation metric assesses the tracking error. High error evaluations cause iterative refinement until the error converges.

Large motions and temporal aliasing are addressed by coarse-to-fine multiscale tracking. The affine motion model allows for local geometric distortions due to large view variations and long-sequence tracking. The affine parameters also facilitate tracking evaluations by modeling region and point motions. A comparison of the modeled motion and the observed motion evaluates the model error, and this information enables the feature detection stage to continuously select the best features to track. This closed-loop control of the tracking system is inspired by the use of feedback for stabilizing errors in nonlinear control systems. The process acts as a "selection-hypothesis-verification-correction" strategy that makes it possible to discriminate between good and poor tracking features, thereby producing motion estimates with consistent quality.

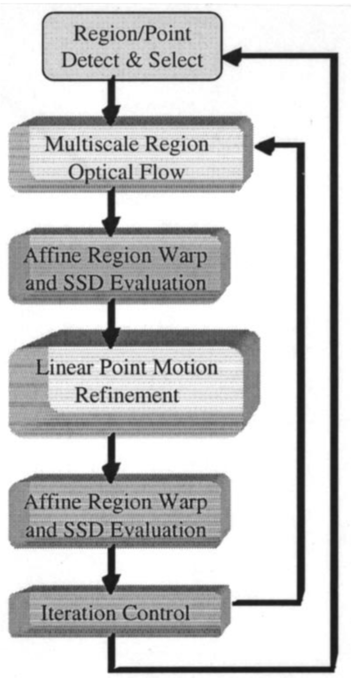

Region/Point Detect & Select
Multiscale Region Optical Flow
Affine Region Warp and SSD Evaluation
Linear Point Motion Refinement
Affine Region Warp and SSD Evaluation
Iteration Control

Fig. 1. Functional blocks for closed-loop motion tracking.

III. Feature Selection and Evaluation

A. Integrating Point and Region Features

Robust 2-D motion tracking depends on both the structures of the selected features and the methods used to track them. Because of their complementary tracking qualities, 0-D point features and 2-D region features are combined in our method. In general, region features are easier to track because the whole region participates in the temporal matching computation. However, region features are prone to significant imaging distortions that arise from variations of view, occlusion, and illumination. For example, a region that includes a foreground fence against a background hillside creates difficulties under camera translation because of the different motions within the region. Is the region motion defined by the fence or the hillside motions? Our philosophic approach to this question is that it does not matter which one is tracked, as long as the region motion tracks one of them consistently. Region tracking requires strong constraints to compensate for these conditions. Unfortunately, the scene geometry needed to model these constraints is usually unknown, so region features often only recover approximate image motion. Our approach constrains each region to track a planar part of the scene. During evaluation, regions are rejected if their motions do not approximate a planar scene motion model. The actual plane orientation is not significant, and each region is free to approximate a different planar orientation.

Accurate 2-D feature motions are required to estimate egomotion. Small-scale point features have the advantage that motion measurements are often possible to at least pixel resolution. The related disadvantage of point tracking is that it becomes difficult in complex scenes, especially under large camera motions. If many point features are detected and tracked reliably, they produce a sparse but accurate motion field suitable for computing egomotion. Observations from methods using large-scale features or dense motion fields indicate that the most reliable measurements often occur near feature points [4].

Considering the complementary strengths and weaknesses of point and region tracking, an integration of both features may attain our goal of an accurate and robust motion field. The feature selection stage identifies good points for tracking (as described in Section III-C) and then identifies regions that encompass clusters of these points. The region tracking process maintains the global relationships between the points in a region, and provides an estimate of point motions. Region motion is coarse but relatively robust for large camera motions, partial occlusions, and long tracking sequences. The approximate point motions defined by a region are refined by correlation to produce an accurate motion field.

General feature detection is a nontrivial problem. For motion tracking, features should demonstrate reliability and stability with the tracking method, even if they do not have any physical correspondence to real-world structure. In another words, the design of feature detection methods should also consider the tracking method used for these features, and vice-versa. (In Sections III-C and III-D, we detail our integrated method for point and region feature detection.) Our detection and selection methods are adaptive and fully data-driven, based on a prediction of the feature's suitability for tracking and an evaluation of its actual tracking performance. To help derive our selection metrics, we first introduce the equations used for optical flow computing and region tracking.

B. Motion Estimate Equations

As a camera moves, image intensity patterns change as a function of three variables $I(x, y, t)$. However, images taken at near time instants are usually strongly related to each other. Formally, this means that the function $I(x, y, t)$ is not arbitrary, but satisfies an intensity conservation constraint that leads to the principal relationship between intensity derivatives and image motion (optical flow): the *optical flow constraint equation* [12]

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{v} + I_t(\mathbf{x}, t) = \mathbf{0} \tag{1}$$

where $\mathbf{v}$ is the feature motion vector, $I_t(\mathbf{x}, t)$ denotes the partial time derivative of $I(\mathbf{x}, t)$, $\nabla I(\mathbf{x}, t) = [I_x(\mathbf{x}, t), I_y(\mathbf{x}, t)]$, and $\nabla I \cdot \mathbf{v}$ denotes the usual dot product.

Motion estimation based on (1) relies on the spatial-temporal gradients of image intensity. This formulation is an ill-posed problem requiring additional constraints. A global model does not typically describe unconstrained general flow fields. Different local models facilitate the estimation process, including constant flow with a local window and locally smooth or continuous flow [12], [16]. The former constraint facilitates direct local estimation, whereas the latter model requires iterative relaxation techniques. We use the local constant model because the results compare favorably with other methods [4], and it is efficient to compute. In this approach, optical flow is constrained to a constant in each small spatial neighborhood. Motion estimates are computed by minimizing the weighted least-square fit

$$E(\mathbf{x}) = \sum_{\mathbf{x} \in \Omega} W^2(\mathbf{x})[\nabla I(\mathbf{x}, t) \cdot \mathbf{v} + I_t(\mathbf{x}, t)]^2 \tag{2}$$

where $W(\mathbf{x})$ denotes a window function that gives more influence to pixels at the center of the neighborhood than those at the periphery. Minimizing this fitting error with respect to $\mathbf{v}$ leads to the equation $\nabla E(\mathbf{v}) = \mathbf{0}$, from which the optical flow field is computed

$$\mathbf{v} = \mathbf{A}^{-1}\mathbf{B} \tag{3}$$

and

$$\mathbf{A} = \sum_{\mathbf{x} \in \Omega} W^2(\mathbf{x}) \begin{bmatrix} I_x(\mathbf{x}, t)^2 & I_x(\mathbf{x}, t) I_y(\mathbf{x}, t) \\ I_y(\mathbf{x}, t) I_x(\mathbf{x}, t) & I_y(\mathbf{x}, t)^2 \end{bmatrix}$$

$$\mathbf{B} = \sum_{\mathbf{x} \in \Omega} W^2(\mathbf{x}) \begin{bmatrix} I_x(\mathbf{x}, t) I_t(\mathbf{x}, t) \\ I_y(\mathbf{x}, t) I_t(\mathbf{x}, t) \end{bmatrix}.$$

Solving for inter-frame motion $\mathbf{v}$ at each pixel or feature and integrating $\mathbf{v}$ over a sequence of images, estimates a feature's motion over an aggregate time interval.

The above equations assume linear or translation motion for the spatial extent of the window function $W(\mathbf{x})$. While this assumption is adequate for small image regions undergoing small inter-frame motions, large image motions, large image regions, or motion discontinuities at foreground and background silhouettes often violate the assumption. To compensate for the geometric deformations caused by large motions and regions we apply a more general *affine* motion constraint to the motion of a whole region [11], [23]. Motion discontinuities still cause problems for region tracking; however, these are addressed by our integration of region tracking with point tracking and a verification process (as described below).

C. Point Feature Selection

Consider the motion-estimation equation (3), given above. The system has a closed-form solution when the 2×2 matrix $\mathbf{A}$ is nonsingular. The optical flow at a point is only reliable if $\mathbf{A}$ is constructed from image measurements that allow its inversion at that point. The rank of $\mathbf{A}$ is full unless the directions of gradient vectors everywhere within the window are similar. $\mathbf{A}$ must be well conditioned, meaning its eignvalues are significant and similar in magnitude. The matrix $\mathbf{A}$ is a covariance matrix of image derivatives, which indicates the distribution of image structure over a small patch [4], [23]. Small eignvalues of $\mathbf{A}$ correspond to a relatively constant intensity within a region. One large and one small eignvalue arise from a unidirectional texture pattern. Two large eignvalues represent corners, salt-and-pepper textures, or uncorrelated intensity patterns.

The eigendistribution of covariance matrix $\mathbf{A}$ predicts the confidence of the optical flow computation at a point and is therefore useful as a metric for selecting point features. Image points with both eignvalues above a threshold are accepted as candidate point features. (Our implementation uses a 7 $\times$ 7 patch to define a point feature.)

$$\min(\lambda_1,, \lambda_2) > TH. \tag{4}$$

Candidate features have a predicted tracking confidence based on their minimum eignvalue $\lambda = \min(\lambda_1, \lambda_2)$. The predicted confidence is combined with a measured tracking evaluation δ fed back from the tracking stage. The final confidence value assigned to a point feature is defined as

$$C = k_1\lambda + k_2\delta \tag{5}$$

where k_1, k_2 are weighting coefficients. Ranked by their confidence values C, the best candidate features are selected as the final point feature set $\{PF_i\}$. (The number selected is an application parameter, but 10–50 is typical for our tests.)

$$\{PF_i = \mathbf{x}_i(C) \mid i\lfloor \textit{candidate set}, \, C > \textit{threshold}\}. \tag{6}$$

The point feature set is updated dynamically. No updates are needed while the system tracks a sufficient number of points and regions. New features are added to the set to replace features whose confidence values fall below an acceptance threshold or features that move off-screen. Since feature confidence derives from both the information that determines the tracking algorithm's stability (λ) and an evaluation of the algorithm's tracking performance (δ), the total system automatically adapts to a scene, locating and tracking the best features available.

D. Region Feature Selection

Region features provide global guidance for accurate point motion estimation so regions are deemed reliable for tracking if they include a sufficient number of point features. In our implementation, the image is divided into nonoverlapping (31 $\times$ 31-pixel) candidate regions R_i. The number of points in each candidate region is tabulated, and the regions with the most features are selected as final region features $\{RF_i\}$

$$\{RF_i = R_i(S) \mid i\lfloor \textit{candidate region}, \, S > \textit{threshold}\} \tag{7}$$

where the quality metric is S, given by

$$S = \frac{N_p}{N_T} \tag{8}$$

where N_p is the number of point features within the region, defined by (6), and N_T is the total number of pixels in the region. The number of region features is arbitrary, depending on the complexity of the scene structure and the application. (Three to six regions are typical for our applications.) Processing time is approximately linear in the number of regions.

IV. Feature Tracking and Feedback

Imaging distortions, especially in the natural environment, can significantly alter feature appearance and cause unreliable tracking. A tracking system cannot prevent these effects, and their variety and complexity make it difficult for any algorithm to track accurate motions in their presence. Our algorithm attempts to detect and purposefully ignore scene features that suffer from distortions. With feedback from the tracking stage, our algorithm detects poor tracking of point and region features. The system automatically rejects point and region motions that disagree or fail to match the piece-wise planar scene assumption. This strategy assumes that the scene contains regions with point features that are approximately planar, a fairly general assumption for natural scenes. Even at the silhouettes of different foreground and background motions, our method tracks points in one or the other scene plane. Where severe conditions cause tracking to fail, the points (and regions) are automatically rejected and do not corrupt the tracking system output.

A. Tracking Algorithm Design

The *optical flow constraint equation* (3) is ill-posed because there are two unknown components of velocity $\mathbf{v}$ constrained by only one linear equation. Only the motion component in the direction of the local image gradient may directly be estimated. This phenomenon is commonly known as the *aperture problem* [28]. The motion can be fully estimated at image locations with sufficient intensity structure. Constraints in addition to (1) are necessary to solve for both motion components at a given point.

A tracking evaluation or confidence measure is an important consideration for optical flow computing and tracking. It is almost impossible to estimate accurate motion for every image pixel, due to the aperture problem, imaging distortions, and occlusions. Observations with many methods attempting to recover full motion fields show that the most reliable measurements often occur near significant feature points, and it is commonly realized that appropriate confidence measures are necessary to filter the estimated motion field. Confidence measures in current optical flow algorithms make use of local image gradient, principal curvature, condition number of solution, and eignvalues of covariance matrix. In these methods, however, the measures are often employed as a post-process to threshold the optical flow field at every pixel. Recent work on image motion estimation focuses on finding a balance between local dense motion estimates and global approaches. In our method, the confidence measure is a dynamic measure of a feature's tracking stability. We do not attempt to perform a global computation, favoring instead the dynamic properties of region and point motion estimates.

B. Multistage Tracking Iterations

The multistage strategy includes three basic steps: a) image warping, b) motion residual estimation, and c) motion model refinement. Let $R_{t_0}(\mathbf{x}, t_0)$ be a region selected for tracking in the frame t_0. $R_t(\mathbf{x}, t)$ is the corresponding target region at time t. A parameter vector $\mathbf{v} = [v_1, v_2, \ldots, v_6]$ describes

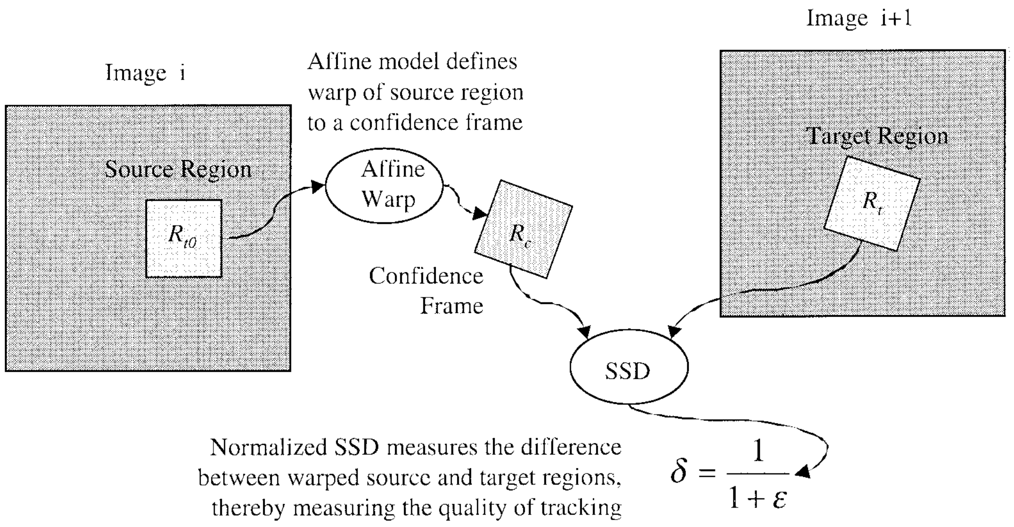


Image i
Source Region
R_{t0}
Affine model defines warp of source region to a confidence frame
Affine Warp
R_c
Confidence Frame
Image i+1
Target Region
R_t
SSD
Normalized SSD measures the difference between warped source and target regions, thereby measuring the quality of tracking
$\delta = \frac{1}{1+\varepsilon}$

Fig. 2. Tracking evaluation compares the motion predicted by the current affine parameters to the observed motion.

the translation motion of the region (at its the center) and its affine deformation parameters. As shown in Fig. 2, a new region $R_c(\mathbf{x}, t)$ can be reconstructed, based on the parameters, by warping the region $R_{t_0}(\mathbf{x}, t_0)$ toward $R_t(\mathbf{x}, t)$

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} v_1 x_{t_0} + v_2 y_{t_0} + v_3 \\ v_4 x_{t_0} + v_5 y_{t_0} + v_6 \end{bmatrix}. \tag{9}$$

The newly constructed region $R_c(\mathbf{x}, t)$ is called a *confidence frame*. The new region, derived from the motion estimate parameters, facilitates an evaluation of how well the parameters model the observed motion. The error of the motion estimate is computed as the least-squares distance between the confidence frame $R_c(\mathbf{x}, t)$ and its target $R_t(\mathbf{x}, t)$

$$\varepsilon = \frac{\|R_t(\mathbf{x}, t) - R_c(\mathbf{x}, t)\|^2}{\max\{\|R_t(\mathbf{x}, t)\|^2, \|R_c(\mathbf{x}, t)\|^2\}}. \tag{10}$$

Region and feature motion estimates are computed at multiple image scales to handle large inter-frame motion and temporal aliasing effects. (Three scales are typical for our implementation and tests.) Gradient-based optical flow methods are sensitive to numerical differentiation, and the coarse-to-fine process keep the images sufficiently well registered at each scale for numerical differentiation. Starting at the highest scale (coarse), region motion is estimated by optical flow (3), and the resulting field determines (by least-squares fit) a set of affine motion parameters $\mathbf{v_r}$. These parameters are evaluated by the confidence frame method described above, producing an error ε_r. Point motions within the region are estimated from the region parameters $\mathbf{v_r}$. Point motion estimates are refined by local correlation searches to subpixel resolution. The refined point motions determine a new affine parameter set $\mathbf{v_p}$ for the whole region. These parameters are evaluated, producing error ε_p. If $\varepsilon_r > \varepsilon_p$, the $\mathbf{v_p}$ parameters are used to determine a new region for an optical flow calculation, and the region motion is estimated again to start another iteration. If $\varepsilon_r \approx \varepsilon_p$, the next iteration is performed at a lower scale, and once the lowest scale is reached, the iterations terminate. This iterative multi-stage tracking procedure is summarized in the following pseudocode:

```
from coarse to fine image scale levels do {
    compute $\mathbf{v_r}$ from region optical flow
    $\varepsilon_r$ = confidence frame evaluation of $\mathbf{v_r}$
    refine point motions and compute $\mathbf{v_p}$
    $\varepsilon_p$ = confidence frame evaluation of $\mathbf{v_p}$
} while (($\varepsilon_r > \varepsilon_p$) && (iterations < limit))
```

If the residual error diverges or remains above a threshold after a preset number of iterations, the region points have their tracking confidence δ reduced to eliminate them from the point feature list. If the number of point features or regions drops below a threshold, a reselection process identifies new regions and points for tracking.

The integration of region and point tracking is related to multiscale methods [25]. Our approach, however, tracks regions and points differently, and their agreement or disagreement provides the additional information about scene motion that facilitates tracking evaluation.

C. Tracking Evaluation and Feedback

Tracking evaluations are fed back to the feature selection stage to dynamically "optimize" the system for tracking the most reliable features. In (5), the tracking confidence is employed to select and rank features according to their dynamic reliability. This allows the system to respond gracefully as features become occluded or distorted over time. Tracking confidence δ is derived from the evaluation processes

$$\delta = \frac{1}{1+\varepsilon} \tag{11}$$

where ε is the motion residual defined in (10).

V. Test Results and Comparisons

Our tracking system implementation is tested on a number of synthetic image sequences (for which the true motion fields are known) and real video sequences. To quantify accuracy, we use the angle error measure [4] and standard RMS error

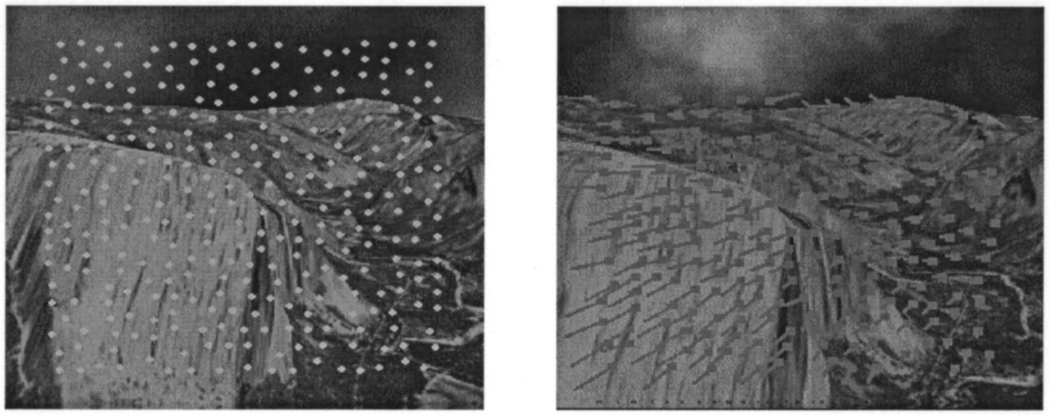

(a) (b)

Fig. 3. Synthetic image sequence (Yosemite-Fly-Through) for accuracy comparison (a) detected tracking features (b) estimated motion field.

measure. The angle error measure treats image velocity as a spatio-temporal vector $\mathbf{v} = (u, v, 1)$ in units of (pixel, pixel, frame). The angular error between the correct velocity $\mathbf{v}_c$ and the estimate $\mathbf{v}_e$ is defined as

$$Error_{\text{angle}} = \arccos(\mathbf{v}_c \cdot \mathbf{v}_e) \tag{12}$$

where $\mathbf{v}_i = \frac{(u,v,1)^T}{\sqrt{u+v+1}}$. This angle error measure is convenient because it handles large and small speeds without the amplifications inherent in a relative measure of vector differences. The measure also has a potential bias, for example, directional errors at a small velocity do not give as large an angular error as a similar directional error at large velocity. For these reasons, we also use the RMS error measure

$$Error_{\text{rms}} = \sqrt{\frac{\sum_{\mathbf{x}\in\Omega}(I_c(\mathbf{x},t) - I_e(\mathbf{x},t))^2}{MN}} \tag{13}$$

where $I_c(\mathbf{x}, t)$ is a size $M \times N$ region of a real image sequence at time t, and $I_e(\mathbf{x}, t)$ is the reconstructed region based on the estimated motion field. Note that this error measure is similar to the tracking evaluation measure we use in Section IV.

A. Optical Flow Tracking Comparison

Extensive experiments have been conducted to evaluate and compare our multi-stage technique with traditional optical flow methods. Fig. 3 illustrates an experimental result for the Yosemite-Fly-Through sequence. The motion of a camera along its view axis toward the mountain and valley generates diverging motion flow around the upper right of the mountain, producing one pixel-per-frame translation motion in the cloud area and about four pixels per-frame of motion in the lower-left area. For this test, only one image region is selected as a tracking region with its size equal to the original image size (256 × 256). In the region, the top 50% of the pixel evaluation values are selected as point features. We chose these numbers for performance comparisons with other optical flow approaches that compute motion estimates for full images. Fig. 3(a) shows the selected tracking points, and Fig. 3(b) illustrates the final tracking results after fifteen frames. In this test, about 3% of the initially selected features were declared as unreliable due to low tracking confidence (with 0.7 as the feature evaluation threshold and a 15 × 15 point-feature window size). The resulting average angle error is 2.84, and the RMS measure is 7.31.

TABLE I
ACCURACY COMPARISON FOR VARIOUS OPTICAL FLOW METHODS

Technique	Average Angle Error	Standard Deviation
Horn and Schunck	11.26	16.41
Lucas and Kanade	4.10	9.58
Anandan	15.84	13.46
Fleet and Jepson	4.29	11.24
Closed-loop	2.84	7.69

Fig. 4 illustrates a similar experiment on a real video sequence from the NASA training scene described in Section VI-A. The scene undergoes significant changes in viewing pose, lighting and occlusions. As in the Yosemite test, the image region is also set equal to the original image size of 320 × 240, and the top 50% of the pixel evaluation values are selected as tracking features. Fig. 4(a) shows the computed motion fields that produced a 4.21 RMS error measure after 30 frames with our multistage approach. Fig. 4(b) shows the results of the same sequence computed with Lucas and Kanades's different-based optical flow method [16]. Lucas's approach is a typical local different-based optical flow technique, in which the optical flow field is fitted to a constant model in each small spatial neighborhood, and the optical flow estimates are computed by directly minimizing the weighted least-squared fitting to (2). To select the most reliable estimates, the eignvalues of the image covariance matrix are used as a post-processed confidence measure to filter the estimated flow field at every pixel [4]. This approach performs a global computation and results in an uncontrolled estimate distribution, so that in many cases a single scene feature can not be tracked consistently. The RMS estimate error for this sequence is 58.11 with a 1.0 eigen-confidence threshold.

As a relative performance comparison, we include our results with those of other published optical flow methods, including Horn and Schunck's global regularization algorithm

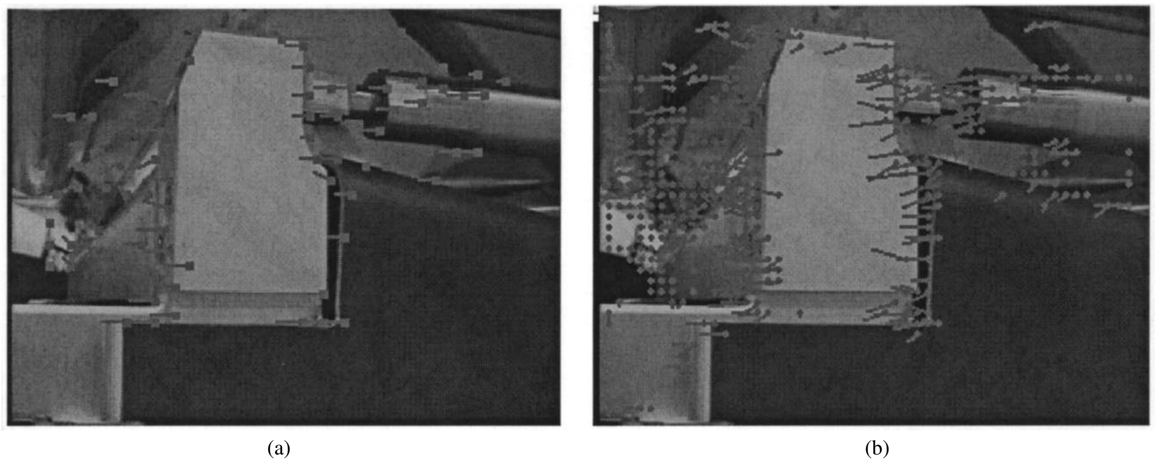


Fig. 4. Comparison of (a) our closed-loop method with (b) filtered optical flow motion estimates. The images are from the application described in Section VI-A.

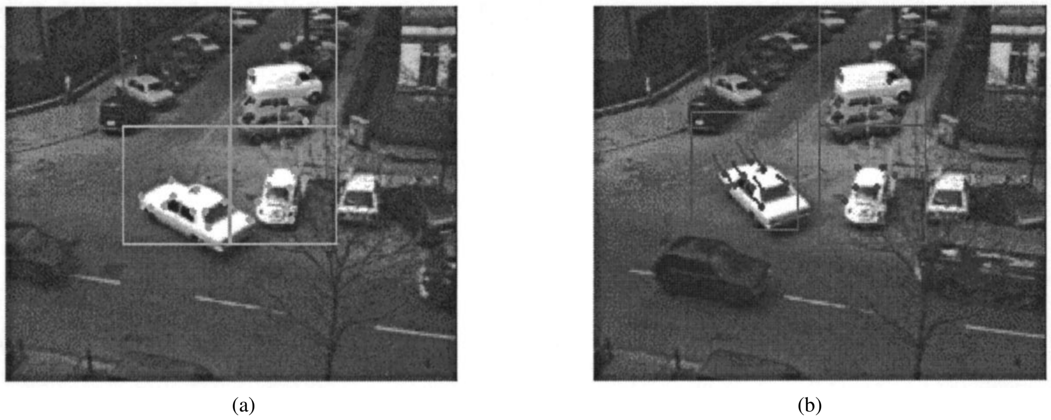


Fig. 5. Real scene sequence (Hamburg Taxi): (a) first frame with detected region and point features, and (b) motion results at the twentieth frame.

[12], Lucas and Kanade's local differential method [16], Anandan's matching correlation algorithm [1], and Fleet and Jepson's frequency-based method [8]. Table I summarizes the results for this sequence. The data show superior accuracy for our multistage approach.

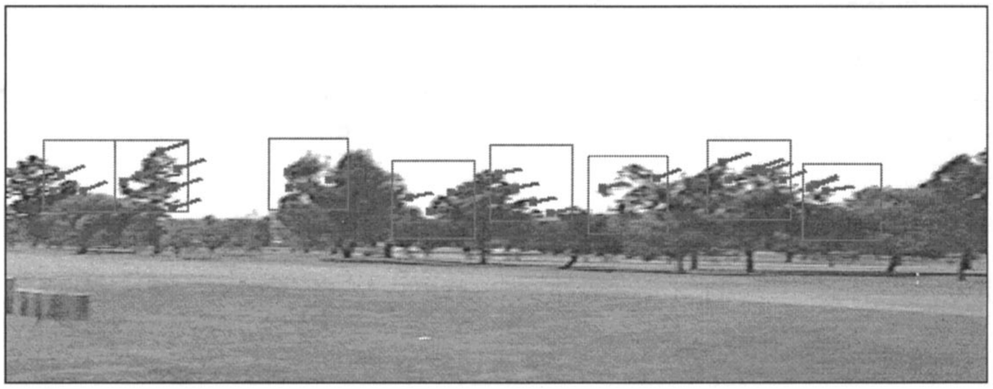
B. Tracking System Experiments

These experiments test and evaluate our whole tracking system with real image sequences captured in different settings and under different imaging conditions. This sequence of Fig. 5 (Hamburg Taxi) contains four cars moving with different motion directions and velocities against a static street background. In this test, three regions are automatically selected for motion estimation. The sizes of these regions are 61 × 61, and in each region, about ten top-ranked points are automatically selected for motion tracking [Fig. 5(a)]. It is worth noting that automatically detected features cluster around the significant physical features in the scene such as object corners and edges. Normally these types of physical features are expected to be reliable for tracking, as noted in many publications, but our approach selects them based on the tracking metrics, as well as their spatial characteristics. Fig. 5(a) shows a feature that is on the left moving white car detected in the middle region. Apparently, this feature's motion is inconsistent with the other motions within that region. The feature is correctly rejected by the tracking evaluation feedback that controls dynamic feature selection. This example illustrates the behavior of integrated region and point tracking under complex imaging conditions.

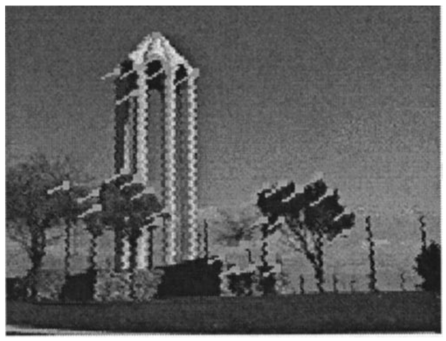
TABLE II
RMS ERRORS FOR MOTION ESTIMATION OF DIFFERENT IMAGE SEQUENCES BY OUR CLOSED-LOOP METHOD

Sequences	RMS
Yosemite-Fly-Through	7.31
Hamburg Taxi	4.54
Park sequence	11.03

Table II gives the RMS estimate error produced by our tracking system for several test sequences, including the Park sequence shown in Fig. 6(a). This latter sequence shows high RMS error, which we believe is due to imaging distortions that occur in the trees as a result of the camera translation. These errors do not preclude the algorithm from automatically detecting and robustly tracking the features marked along the tree-sky silhouette. Both sequences in Fig. 6 were captured



(a)



(b)

Fig. 6. Tracking result for an outdoor natural scene show the selection of what the algorithm autmatically selects as the best points and regions to track. The park sequence (a) illustrates the selection of features and regions along the tree-sky slihouette. The towersequence (b) shows features selected on the forground trees, background fence, and structure. Both sequences are obtained from amoving vehicle by a hand-held 8 mm camcorder.

with an 8-mm camcorder from a moving vehicle while viewing to the right of the motion direction and panning the camera. Both contain irregular natural objects such as trees and grass. It is almost impossible to predict what kind features should be adopted for detection and tracking in scenes like these.

VI. Applications to Augmented Reality

In this section, we present examples that show the benefits of using our natural tracking system for AR applications. We show the capability of direct scene annotation, extendible AR tracking range, and pose stabilization with natural features.

A. Direct Scene Annotation

The addition of virtual annotations for task guidance is a typical AR application, and in many cases, the annotations appear on objects whose positions in the world may vary freely without impact on the AR media linked them. For example, AR annotation can identify specific components on a subassembly or portion of structure that moves throughout an assembly facility [2], [19], [29]. A full 6DOF camera pose is often not needed to maintain this simple form of annotation. In this example, we use our 2-D tracking method to directly track structure features that are annotated as a camera moves to provide more detail and context.

The scenario is developed in collaboration with Dr. A. Majoros (The Boeing Company) for a NASA astronaut training application. Space station astronauts may shoot a video sequence to illustrate a problem that requires assistance from ground-station experts. Ground-based experts use an AR workstation to process the video and interactively place annotation. The experts select keyframes and link text and images (annotation) to structural features in the image. The tracking system then automatically keeps the annotation linked to the features as the camera moves in the following frames to show additional structure or views that clarify the context and extent of the problem. In our scenario tests, the tracking system must do its best in response to hand-held camera motion. Fig. 7(a) shows three keyframe images taken from camcorder video sequences. Features in the keyframes are interactively identified and annotated with text banners. Fig. 7(b) shows later images from the three sequences. The numbers between the image pairs specify how many frames (70 or 75) transpired between the initial and final annotated images. These sequences demonstrate feature tracking under significant changes in viewing pose and lighting. Note that these features are manually selected so the algorithm has no choice about the features to track. However, even in the presence of considerable background, lighting, scale, and view direction changes, the method succeeds in tracking the selected points.

B. Extendible Tracking and Pose Stabilization

A second application of natural feature tracking is the automatic extension of an AR system's workspace. As noted

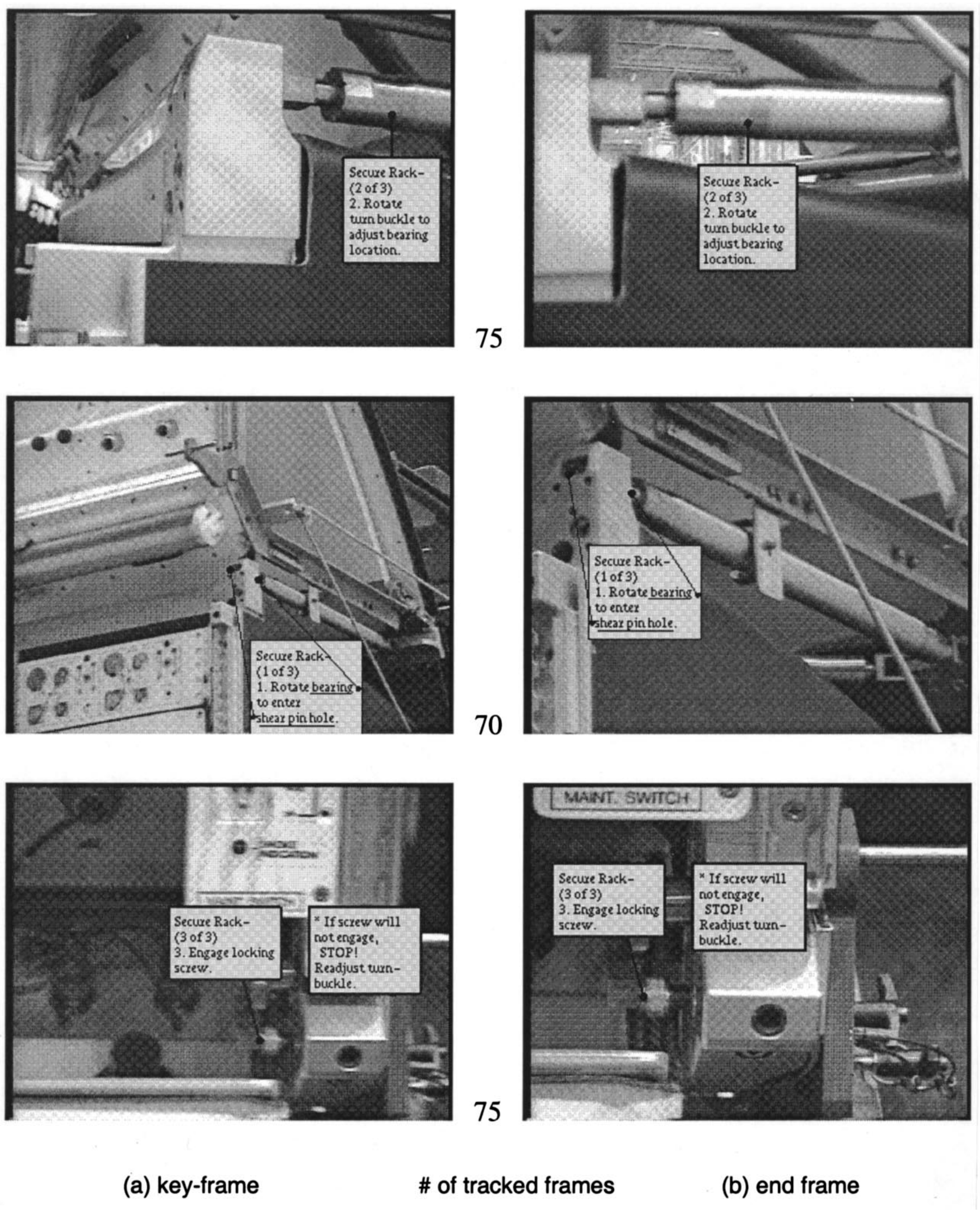


Fig. 7. Direct scene annotation (a) initial frames used to interactively place annotations, (b) later frames in the same sequences showing the automatic tracking of the selected features. Numerals between image pairs indicate how many frames are tracked. (a) Keyframe and (b) endframe.

previously, vision-based AR systems often rely on artificial landmarks (fiducials), or a priori known models to perform dynamic tracking and alignment between the real and virtual camera. These approaches are appropriate in situations where known and recognizable features are always in view. The dependence upon known feature positions inherently limits the tracked range of camera poses to a bounded working space. If the camera moves beyond these bounds, the image no longer supports tracking unless additional information is available to the system. A means of providing this new information is to track the naturally occurring features and dynamically calibrate them so they can be used as additional fiducials. In this way, naturally occurring scene features extend the AR tracking range.

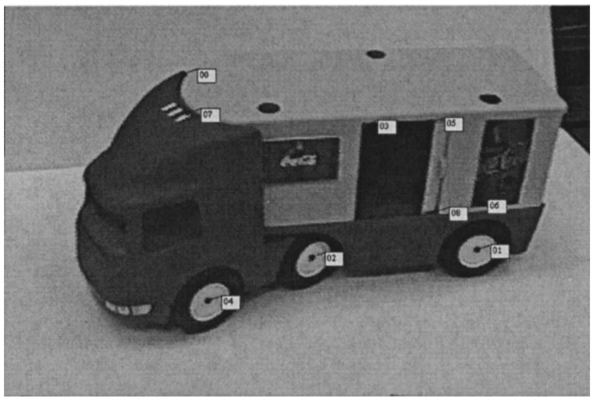
We developed an extendible AR tracking system by incorporating our tracking approach with an Extended Kalman Filter (EKF) that estimates the 3-D positions of natural features [20]. The 6DOF camera pose is derived from three visible features, as in many other systems. Initially, the camera pose is based on the known fiducials, and our method automatically selects and tracks natural features. As the camera moves and its pose is tracked over multiple frames, the recursive filter EKF automatically estimates the 3-D positions of the tracked natural features. Once the 3-D positions of the natural features are known within an accuracy threshold, these features facilitate continued camera pose computation in the absence of visible fiducials. This approach allows a system to automatically extend it AR's tracking range during the course of its use. In principle, an AR system may increase its robustness as it is used.

Fig. 8 illustrates the first of two extendible tracking experiments. Approximately 300 video frames were acquired by a handheld camera and digitized. In automatic processing of the sequence, ten features were detected, and nine were automatically selected for tracking [marked with tags in Fig. 8(a)]. One feature was automatically rejected for being too close to another selected feature. The annotations and colored circle fiducials are at known calibrated 3-D positions. Fig. 8(b) shows the initial frame with fiducial-based camera tracking, and Fig. 8(c) shows the 295th frame with camera tracking derived from automatically calibrated natural features. The calibration convergence of the natural features is illustrated in the lower row of Fig. 10. The features converge rapidly to their final position values.

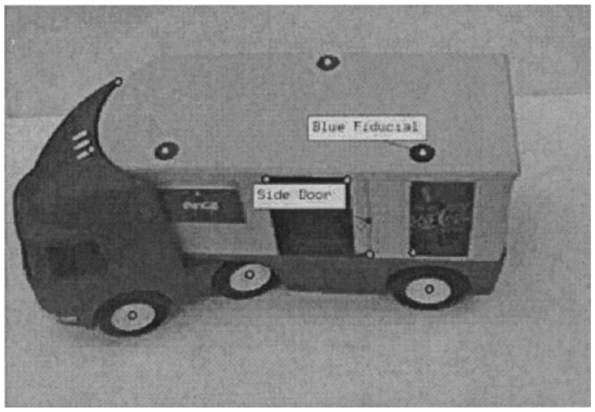
A similar second experiment is illustrated in Fig. 9. A 250-frame video sequence of a rack model was digitized from a mockup of the NASA application described in Section VI-A. The annotation and colored circle fiducials [on the right side of Fig. 9(a)] are at known calibrated positions. Twenty natural features were detected in the first frame, and 12 (shown as white dots) were selected for tracking; the others were rejected for being too close to the already-selected features. Fig. 9(a) shows the 124th frame: the first frame for which camera pose is computed from calibrated natural features (marked with yellow crosses). Fig. 9(b) shows a later frame with the fiducials completely off screen, leaving only the calibrated features to support camera tracking. The upper row of Fig. 10 shows the convergence of the natural feature's X, Y, and Z 3-D position coordinates. Convergence takes about 90 frames and remains stable. Note that the initial Z coordinate estimates are less accurate than the X and Y coordinates. This is largely due to the camera image-plane being predominantly aligned with the X-Y plane.

Extendible tracking stabilizes the pose calculation against occlusions and noise. In many cases, where users interact with world object, their hands or tools can easily occlude the fiducials or features needed to support tracking. As shown in the examples above, when insufficient fiducials are visible for computing camera pose, for any reason, a system can automatically switch to the highest confidence natural features available.

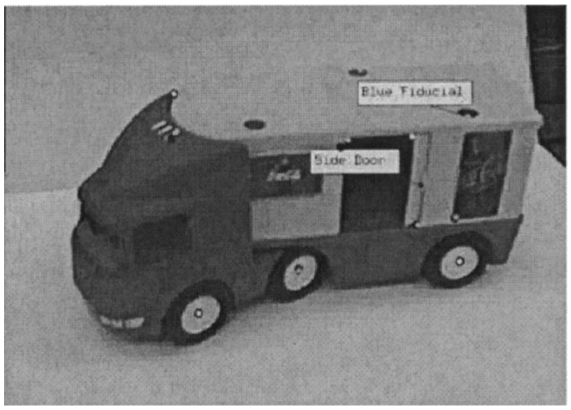
The above applications use real-time video capture and off-line processing. Natural feature tracking (for 640 $\times$ 480 images) takes approximately 0.15 ss per image on an SGI O2 workstation. To simulate a real-time application, the offline processing is completely automatic with no user intervention. We anticipate that optimizations and near-term DSP or custom hardware systems will provide the factor of 5–10 increase in processing power needed for real-time interactive operation.



(a)



(b)

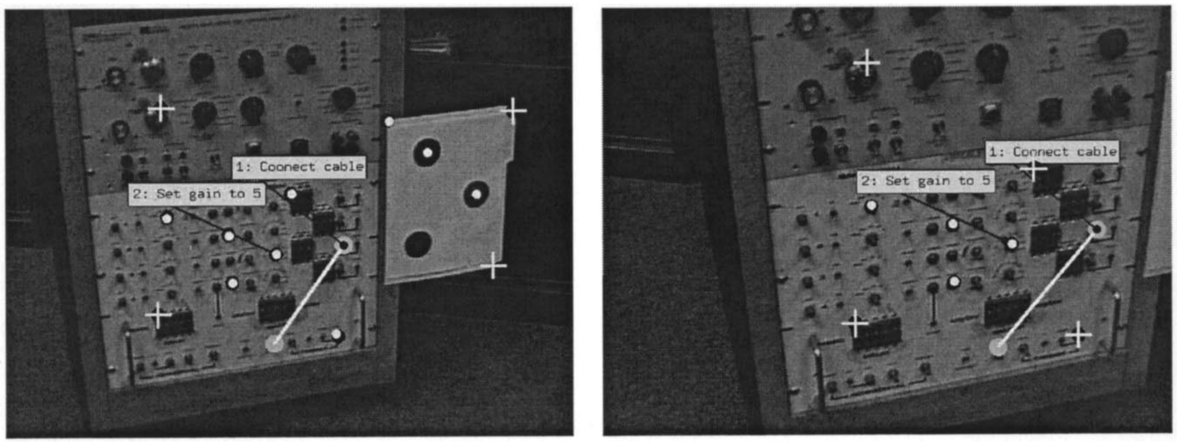


(c)

Fig. 8. This sequence starts by tracking camera pose from fiducials, while natural features are automatically detected and calibrated. Note the annotation indicating the blue fiducial and the side door of the truck. As the camera drops low to the ground at the end of thesequence, the fiducials are no longer usable for tracking since theiraspect is extreme, and the now-calibrated natural featuresautomatically support continued tracking. (a) Tags show positions of automatically detected, tracked, and calibrated natural features. (b) Initial image with fiducial camera tracking (c) Frame 295 with camera tracking based on tracked and calibrated natural features.

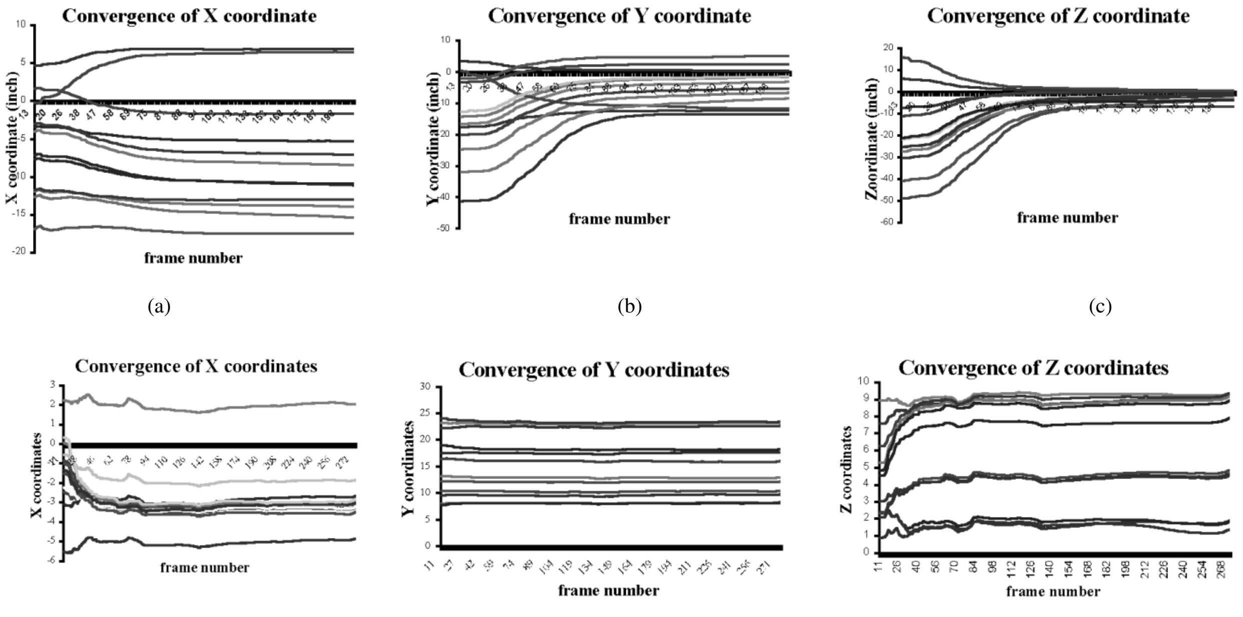
VII. Summary and Conclusion

Natural scene features can stabilize and extend the tracking ranges of augmented reality pose-tracking systems. This paper presents an architecture for robust detection and tracking of naturally occurring features in unprepared environments. Demonstration applications illustrate how such tracking ben-



(a) (b)

Fig. 9. Equipment rack annotation experiment starts with fiducial-based tracking as shown at left side of (a). Camera pan and zoom isolates the lower rack section for a detailed view (b) where calibrated natural features continue to support tracking. The text and line annotations indicate where cable connections are needed and which gain control to set. (a) The 124th frame and (b) the 249th frame.



(a) (b) (c)

(d) (e) (f)

Fig. 10. Convergence of natural feature points in rack and truck model tests.

efits vision-based AR tracking systems. The architecture integrates three motion-analysis functions: feature selection, motion tracking, and estimate evaluation in a closed-loop cooperative manner. Both 0-D point and 2-D region features are automatically and adaptively selected for properties that lead to robust tracking.

The biggest single obstacle to building an effective AR system is the lack of accurate, long-range sensors and trackers that report the locations of the user and the surrounding objects in the environment. Active tracking approaches cannot provide the flexibility and portability needed in wide-area and mobile tracking environments. Vision-based tracking can potentially recognize and locate objects in an environment by measuring the locations of visual features in the natural world and tracking them over time. Furthermore, since vision-based approaches do not rely on any active transmitters, they offer flexibility when dealing with diverse environments. We feel that it is possible to develop more economic and practical AR systems based on vision tracking methods, and our work represents a step toward this goal.

ACKNOWLEDGMENT

The authors acknowledge support from the Defense Advanced Research Project Agency (DARPA) and an NSF CAREER Grant. The NSF-funded “Integrated Media Systems Center” (IMSC) provided support and facilities. Dr. A. Majoros of The Boeing Company, Long Beach, CA, provided invaluable assistance in defining the application scenario presented in Section VI-A. The research members of the AR

Tracking Group at the University of Southern California provided ideas and suggestions (www.usc.edu/dept/CGIT).

References

[1] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *Int. J. Comput. Vis.*, vol. 2, pp. 283–310, 1989.

[2] R. Azuma, "A survey of augmented reality," *SIGGRAPH'95* course notes, Aug. 1995.

[3] M. Bajura and U. Neumann, "Dynamic registration correction in augmented reality systems," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1995, pp. 189–196.

[4] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995.

[5] T. P. Caudell and D. M. Mizell, "Augmented reality: An application of heads-up display technology to manual manufacturing processes," in *Proc. Hawaii Int. Conf. Systems Sciences*, 1992, pp. 659–669.

[6] E. C. Hildreth, "Computation underlying the measurement of visual motion," *Artif. Intell.*, vol. 23, pp. 309–354, 1984.

[7] S. Feiner, B. MacIntyre, and D. Seligmann, "Knowledge-based augmented reality," *Commun. ACM*, vol. 36, no. 7, pp. 52–62, July 1993.

[8] D. J. Fleet and A. D. Jeson, "Computation of component image velocity from local phase information," *Int. J. Comput. Vis.*, vol. 5, pp. 77–104, 1990.

[9] E. Foxlin, "Inertial head-tracker sensor fusion by a complementary separate-bias Kalman filter," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1996, pp. 184–194.

[10] M. Ghazisadedy, D. Adamczyk, D. J. Sandlin, R. V. Kenyon, and T. A. DeFanti, "Ultrasonic calibration of a magnetic tracker in a virtual reality space," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1995, pp. 179–188.

[11] G. D. Hager and P. N. Belhumeur, "Real-time tracking of image regions with changes in geometry and illumination," in *Proc. IEEE CVPR*, 1996.

[12] B. K. P. Horn and B. G. Schunk, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1984.

[13] D. Kim, S. W. Richards, and T. P. Caudell, "An optical tracker for augmented reality and wearable computers," in *Proc. of IEEE Virtual Reality Annual International Symposium*, 1997, pp. 146–150.

[14] G. Klinker, K. Ahlers, D. Breem, P. Chevalier, C. Crampton, D. Greer, D. Koller, A. Kramer, E. Rose, M. Tuceryan, and R. Whitaker, "Confluence of computer vision and interactive graphics for augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 433–451, Aug. 1997.

[15] K. Kutulakos and J. Vallino, "Affine object representations for calibration-free augmented reality," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1996, pp. 25–36.

[16] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA IU Workshop*, 1981, pp. 121–130.

[17] K. Meyer, H. L. Applewhite, and F. A. Biocca, "A survey of position trackers," *Presence: Teleoperators and Virtual Environments*, vol. 1, no. 2, pp. 173–200, 1992.

[18] H. H. Nagel, "On a constraint equation for the estimation of displacement rates in image sequences," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 1, pp. 13–30, 1989.

[19] U. Neumann and Y. Cho, "A self-tracking augmented reality system," in *Proc. ACM Virtual Reality Software and Technology*, 1996, pp. 109–115.

[20] U. Neumann and J. Park, "Extendible object-centric tracking for augmented reality," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1998, pp. 148–155.

[21] R. Sharma and J. Molineros, "Computer vision-based augmented reality for guiding manual assembly," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 3, pp. 292–317, June 1997.

[22] A. State, G. Hirota, D. T. Chen, B. Garrett, and M. Livingston, "Superior augmented reality registration by integrating landmark tracking and magnetic tracking," in *Proc. SIGGRAPH'96*, 1996, pp. 429–438.

[23] Tomasi and T. Kanade, "Shape and motion from image streams: A factorization method," Tech. Rep., Carnegie Mellon Univ., Pittsburgh, PA, Sept. 1990.

[24] M. Uenohara and T. Kanade, "Vision-based object registration for real-time image overlay," in *Proc. Computer Vision, Virtual Reality, and Robotics in Medicine*, 1995, pp. 13–22.

[25] J. R. Bergen and E. H. Adelson, "Hierarchical, computationally efficient motion estimation algorithm," *J. Opt. Soc. Amer.*, vol. 4, no. 35, 1987.

[26] U. Neumann and A. Majoros, "Cognitive, performance, and systems issues for augmented reality applications in manufacturing and maintenance," in *Proc. IEEE Virtual Reality Annu. Int. Symp.*, 1998, pp. 4–11.

[27] R. Haralick, C. Lee, K. Ottenberg, and M. Nolle, "Review and analysis of solutions of the three point perspective pose estimation problem," *IJCV*, vol. 13, no. 3, pp. 331–356, 1994.

[28] S. Ullman, *The Interpretation of Visual Motion*. Cambridge, MA: MIT Press, 1979.

Ulrich Neumann received the M.S.E.E. degree from the State University of New York at Buffalo and the Ph.D. degree in computer science from the University of North Carolina, Chapel Hill, in 1993.

He is an Assistant Professor of Computer Science at the University of Southern California (USC), Los Angeles. His research relates to interactive visual media, including AR tracking systems, 3-D modeling, and animation systems. He has published over 25 conference and journal papers. He is an Associate Research Director for Computer Interfaces in the USC Integrated Media Systems Center (IMSC), an NSF Engineering Research Center. He directs the Computer Graphics and Immersive Technologies (CGIT) Laboratory at USC. In his commercial career, he designed multiprocessor graphics and DSP systems, cofounded a video game corporation, and independently developed and licensed electronic products.

Suya You received the B.Sc. degree in electrical engineering from the Huazhong University of Science and Technology, China, in 1985, the M.Sc. degree in 1988, and the Ph.D. degree in 1994.

From 1994 to 1996, he was a Post-doctoral Research Fellow in computer science at the Tsinghua University, China, and a Research Associate in the Computer Science Department at the State University of New York, Stony Brook, from 1996 to 1997. He joined the University of Southern California, Los Angeles, in 1997 as a Research Associate with the Computer Science Department and the Integrated Media Systems Center. His professional interests lie in visual information processing including computer vision and pattern recognition, computer graphics and visualization, multimedia computing, and advanced human-computer interaction. His current research involves the integrated applications of efficient techniques and algorithms of computer vision for augmented reality and graphics problems.