IEEE VR2003, March 22-26, Los Angeles

# VIS-Tracker: A Wearable Vision-Inertial Self-Tracker

Eric Foxlin & Leonid Naimark
InterSense Inc.
{ericf/leonidn}@isense.com

## Abstract

*We present a demonstrated and commercially viable self-tracker, using robust software that fuses data from inertial and vision sensors. Compared to infrastructure-based trackers, self-trackers have the advantage that objects can be tracked over an extremely wide area, without the prohibitive cost of an extensive network of sensors or emitters to track them. So far most AR research has focused on the long-term goal of a purely vision-based tracker that can operate in arbitrary unprepared environments, even outdoors. We instead chose to start with artificial fiducials, in order to quickly develop the first self-tracker which is small enough to wear on a belt, low cost, easy to install and self-calibrate, and low enough latency to achieve AR registration. We also present a roadmap for how we plan to migrate from artificial fiducials to natural ones. By designing to the requirements of AR, our system can easily handle the less challenging applications of wearable VR systems and robot navigation.*

## 1. Introduction

When you think about computer graphics applications that require motion tracking today, you probably think first of VR games or character animation driven by full-body motion capture. However, in the future computer graphics is likely to make an impact on some industries less glamorous than entertainment, but larger and more indispensable, such as manufacturing, maintenance, construction, transportation, medicine, and the military. One technology that seems poised to make radical improvements in efficiency of operations for all of these industries is augmented reality (AR). It is widely recognized that the motion tracking requirements for AR are more challenging than for VR: the tracking needs to be faster and more accurate, and for large-scale applications like aircraft manufacturing and maintenance or battlefield AR, it needs to work over very large areas. Researchers have been trying to develop a tracking solution that can meet these difficult requirements, with dozens of papers appearing in such conferences as ISAR, ISMR, and IEEE VR. In order to provide tracking over wide areas or outdoors, many of these papers have described various approaches to self-tracking.

The concept of a "self-tracker" has a long and distinguished history, introduced to the CG world by Gary Bishop's 1984 dissertation [3], but practiced even earlier by ancient mariners who navigated the seas using a sextant to measure the bearing angles to celestial landmarks. Most researchers have pursued the lofty goal of a self-tracker that works with absolutely no infrastructure by observing natural features, even outdoors. While we agree that there is a burning need for a self-tracker that can work in an unstructured dynamic outdoor environment, the practical and robust implementation of such a tracker are years away due to the computer vision challenges of finding suitable natural features to track under such diverse conditions. In the meantime, there are still significant applications for a tracker to be used in and around buildings and vehicles, where one can assist the tracker by installing "fiducial" marks that can be extracted quickly and reliably using today's machine vision hardware. In Section 5 we present a roadmap for weaning our system away from reliance on these artificial fiducials. Part of this strategy is to capitalize on the work others are now doing on natural feature recognition, which we can rapidly integrate into our system due to its open architectural framework summarized briefly in Section 2.

Indeed many researchers have worked on self-tracking systems using printed paper landmarks or fiducials [e.g. 2,5,8,9], but they are still not robust, because when the camera moves quickly, the image processor has to search the whole image to find candidate fiducial marks, and this is too slow and unreliable. The system easily becomes disoriented and takes a long time to recover. In order to build a robust and fast self-tracker, there is growing consensus that you need to combine inertial and computer vision technologies using sophisticated sensor fusion algorithms [e.g. 1,6,10,11,12]. This paper describes the first implementation of such a system which has all of the attributes required to be commercially viable:

- It is small, lightweight, and low power, so the processor can be comfortably worn on a belt while the sensor combination is attached to an HMD or handheld device.
- It is reproducibly manufacturable using just three low-cost off-the-shelf components, and automatic self-calibration software.

- It is self-installing, in the sense that after sticking some paper fiducials to the ceiling or walls where tracking is needed, one can auto-map the locations of these fiducials by walking through the area with the tracker itself, and no other surveying is needed.
- It is robust in the face of difficult lighting conditions, fast motions, occlusions, and has very low latency.
- It uses a novel 2-D barcode system to recognize a large number of unique fiducial codes so as to initialize its location over a wide area.

In our search for solutions to the AR problem we turned to the robotics and inertial navigation fields, from which we learned much about computer vision, simultaneous tracking and map-building, and inertial sensors and their modeling, but we had to meet much more difficult accuracy and size requirements than most robots impose. Now that we have met them, we have a device which is smaller and more capable than most commercially-available technology for robots and automatic guided vehicles (AGVs). Compared to laser scanners which are gaining popularity for AGVs, our solution offers full 6-DOF tracking instead of 2-DOF, quicker installation through auto-mapping, many uniquely coded fiducials so it can self-initialize anywhere, reduced reliance on line-of-sight, and no moving parts or lasers. This may turn out to be the near-term application for the device until wearable AR and VR software applications are developed and improved HMDs are available.

## 2. System architecture

The following figure shows the basic architecture of the system. Each self-tracker consists of an inertial measurement unit (IMU), a sensor fusion core (SFC), and one or more "smart cameras".

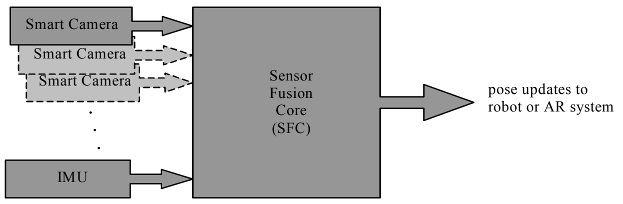

**Figure 1: Basic self-tracking system architecture.**

In the implementation described below, the smart camera is a CCD with an attached DSP that extracts the centroids of artificial fiducials in the image. However, at the architectural level, the smart cameras are very general sensors, and may even include lasers, sonars or radar units. Self-tracking or navigation systems designed for use on mining machines or undersea vehicles may require very different types of sensors than head trackers used in an office building, and the architecture is modular enough to allow substitution. Also, note that Figure 1 is a logical, not physical, diagram of the architecture. The image processing algorithms that make the smart camera smart may be implemented as hardware inside the smart camera package, or as software running on the same CPU as the SFC, and the SFC may have a dedicated processor or run as a driver on the host. However, the interfaces will be standardized to allow easy switching between these very different implementations.

The sensor fusion core (SFC) has three major functions:

1. **Tracking**: During tracking, the SFC uses optimal Kalman filtering algorithms to fuse together the high-rate integration of the IMU with the low-rate absolute environment-referenced measurements of the smart cameras when available. This hybrid approach to tracking is one of the greatest strengths of the system, combining the robustness, low jitter, low latency, interference immunity and predictive capabilities of an inertial tracking system with the absolute registration accuracy of an optical tracker.
2. **Auto-Calibration**: During factory calibration (or user re-calibration after customizing the sensor arrangement) the SFC performs identification algorithms to estimate, and thereafter compensate, all the sensor instrinsic parameters (such as lens focal length and distortion) and extrinsic parameters (position and orientation of each camera relative to the IMU). These auto-calibration algorithms are a major factor in getting high accuracy from combinations of low-cost sensors, without expensive calibration equipment.
3. **Auto-Mapping**: Before using the system for tracking in a new space (and after installing artificial targets if they are needed), the user walks through the space with the tracker and it simultaneously tracks itself relative to the fiducials it already has in its map, acquires new fiducials, adds them to the map, and successively refines their position estimates as new measurements are made. Because the auto-mapping algorithm is based on an augmented Kalman filter formulation for simultaneous localization and map-building, it is not necessary to turn it off to enter the tracking mode, but doing so can speed the update rate.

Figure 2 shows more detail of the sensor fusion core. It shows three complementary Kalman filter blocks, which perform the three functions described above, and a fusion filter that allows them to be used individually or in any combination, based on a novel distributed Kalman filtering algorithm described in [4]. The measurement management unit (MMU) is responsible for acquiring pose and initializing the dynamics reference unit (DRU), which carries out the high-rate dead-reckoning integration of the inertial sensors and continuously updates the pose output between vision measurements. The DRU also provides time-varying linearized process-model matrices to the three complementary Kalman filter blocks, so that they can be designed generically, and any changes in the interoceptive sensors (inertial or encoder-based dead-reckoning) can be encapsulated in the DRU. During tracking, the MMU is responsible for scheduling measurements, performing data association on returned results, and feeding the measurement innovations with their associated linearized measurement model matrices to the complementary filter bank. To do this it receives information about the current pose from the DRU, the sensor and target configurations from the vehicle and environment map managers, and status and characteristics of various sensors from the sensor/target meta-driver.

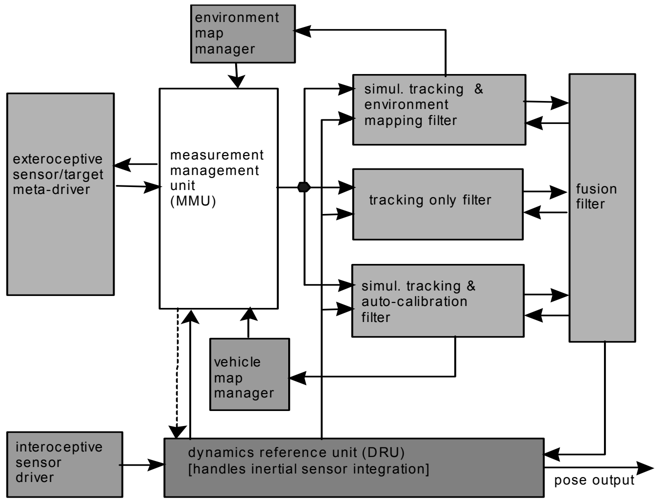
environment map manager
exteroceptive sensor/target meta-driver
measurement management unit (MMU)
simul. tracking & environment mapping filter
tracking only filter
simul. tracking & auto-calibration filter
fusion filter
vehicle map manager
interoceptive sensor driver
dynamics reference unit (DRU) [handles inertial sensor integration]
pose output

**Figure 2: Block diagram of the sensor fusion core (SFC). The key to the extensibility is the standardized interface between the sensor/target meta-driver and the measurement management unit. By hooking up a new type of sensor and placing a driver in a certain directory, the meta-driver will automatically allow the MMU to schedule and process it.**

Much more detail about this highly generalized sensor fusion architecture and the distributed Kalman filtering algorithms that make it possible can be found in [4].

## 3. Implementation

We have built an initial prototype of a self-tracking system based on this architecture. A major focus while building the prototype was to use off-the-shelf hardware. However, in selecting the hardware we were careful to keep the size and power small enough for use on wearable AR systems and mobile robots. As a result, the system can be brought to market extremely quickly (because all three hardware components are already in volume production), and can be further miniaturized or tailored to meet customer needs by swapping COTS hardware subsystems.

Figure 3 shows a physical block diagram of the implemented system. It corresponds quite closely to Figure 1, with a Smart Camera comprised of the CCD and lens and a dedicated image processor board, plus a software driver running on the main CPU board to compensate for lens distortion.

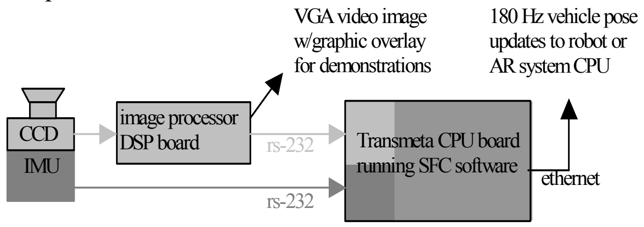
VGA video image w/graphic overlay for demonstrations
180 Hz vehicle pose updates to robot or AR system CPU
CCD
IMU
image processor DSP board
rs-232
rs-232
Transmeta CPU board running SFC software
ethernet

**Figure 3: Prototype implementation block diagram.**

### 3.1. IMU

For an IMU, we used the InertiaCube2, shown in Figure 4. This new InterSense product replaces the IS-300 with a smaller sensor assembly and no base processor unit. Because it is the world's smallest IMU, runs on just 0.5W of power, and has an rs-232 output, this was a natural choice for easy integration in the system of Figure 3. This MEMS-based device measures three axes of angular rate, linear acceleration, and magnetic field (not used in this product) in a package less than 1 inch tall.

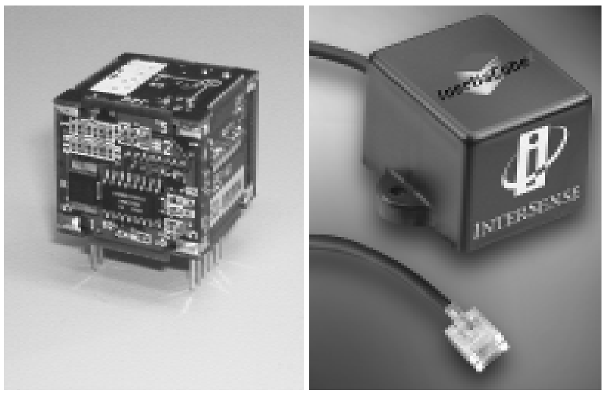

**Figure 4: InertiaCube2 inertial measurement unit contains 3 gyros, 3 accelerometers, and 3 magnetometers in a 1 cubic inch package.**

### 3.2. Smart camera

For the smart camera, we use an image processing board based on the ADSP 2185 chip, and a board camera based on a Sony 640x480 progressive-scan black & white CCD with a micro-video lens having 108° diagonal field of view. This 16-bit DSP board was found to be inexpensive, low power, and sufficiently fast to find our artificial fiducials once we developed a clever and efficient set of image processing stages. At 70mm X 120 mm, it is an acceptable size, and a new version measuring just 25 mm X 50 mm is in development. Because the inertial sensor predicts within several pixels where each fiducial will show up in the image, the image processor only needs to process a sub-area of the image a little bigger than the expected size of the fiducial, typically under 40 x 40 pixels. This is about 200 times faster than processing a 640 x 480 frame, and even a modest DSP chip can do the job in a few milliseconds (see Section 3.4 for more details).

We programmed the camera to perform three main functions in response to serial port commands from the SFC. The first command, acquisition, causes the camera to search the entire image for objects that might be fiducials, and then attempt to read the barcodes of each candidate and report the centroid locations and barcodes of the best-positioned 4 targets that were successfully read, with which the SFC performs a pose-recovery algorithm to establish its initial position. The second function, tracking, is called about 20 times per second after the SFC initializes pose and starts the inertial integrators running. The command includes a search box area within which the SFC predicts that there is a fiducial. The smart camera only processes this sub-image, and returns the centroid of the found fiducial as quickly as possible, without the need to even read the barcode since the SFC already knows which fiducial it is expecting. The third command, "tracquisition", is a hybrid of these two used during automapping. It is sent a few times per second during tracking mode, and only a small area is searched so as not to produce enough delay to throw off the tracking process. However, the goal is to discover new fiducials to add to the map, so within the search box, the camera performs essentially the same operations as acquisition mode, and returns the location and barcode of a fiducial if one is found. All three commands as well as the fiducial design and image processing steps are explicitly described in [7].

## 3.3. Sensor fusion core

We have chosen to implement the SFC software using the Mathworks' Simulink and DSP Blockset, and then convert it to C automatically with Real Time Workshop (RTW). The DSP Blockset provides a very high-level language for rapidly constructing advanced signal processing applications that involve intensive use of matrix operations, linear algebra, and filtering. The greatest time savings were realized by using RTW to automatically convert the Simulink models into production-quality C-code, which saved months of coding and even more months of debugging. Another advantage is that because RTW was designed for developing real-time embedded firmware, it provides us with a greater level of control over the timing and execution order of our subsystems than we could achieve by hand-coding in C, which is critical for a tracking system which needs to have minimal and precisely deterministic latency. A third advantage is portability. The system can be quickly re-targeted to various microprocessors and real-time operating systems using customized or off-the-shelf target description files.

For hardware, we currently use a 100 X 150 mm embedded PC board with a 667 MHz Transmeta Crusoe processor, running Windows NT Embedded off a Compact Flash card. At 9W, it is considerably more efficient than a comparably fast Pentium, and smaller and less expensive than a packaged wearable computer or mini-laptop.

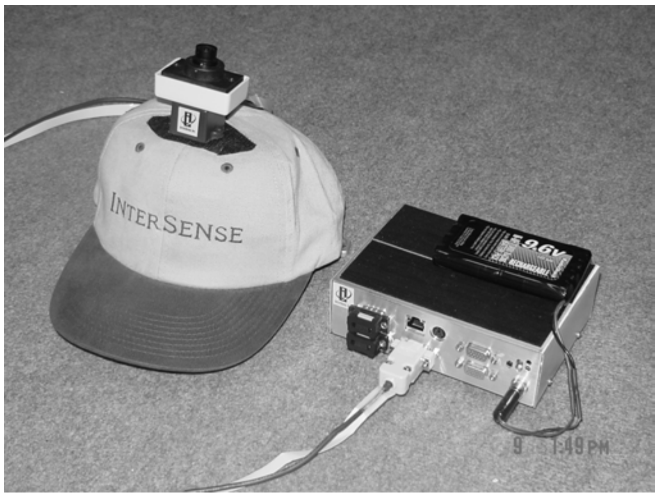

**Figure 5: Photograph of prototype self-tracking system.**

Figure 5 shows a photograph of our complete prototype system, including the head-mounted sensor assembly of camera and InertiaCube, and the belt-mounted electronics unit that houses the CPU board and image processor board. The electronics unit is about 110 x 150 x 50 mm, and weighs 580 g. It consumes 13 W of power, which adds about 100 g of Lithium-ion batteries per hour of desired battery life. The sensor head is about 50 mm tall and weighs 70 g. The embedded operating system has been configured to auto-login on power up and run the SFC program, which communicates with the InertiaCube and Smart Camera, and broadcasts its cooked position and orientation data as UDP packets out the built-in Ethernet port of the CPU board.

## 3.4. Timing and latency mitigation

Figure 6 shows the timing of communication between the SFC and the InertiaCube and smart camera in tracking mode. The inertial data are received every 8.3ms (although each data packet represents a pre-integrated batch of samples internally taken in the IMU at a 1920 Hz rate). Integrated 6-DOF pose data is sent to the external computer/graphical processor every 16.6 ms to support the 60Hz rates of graphics typically required for AR/VR.

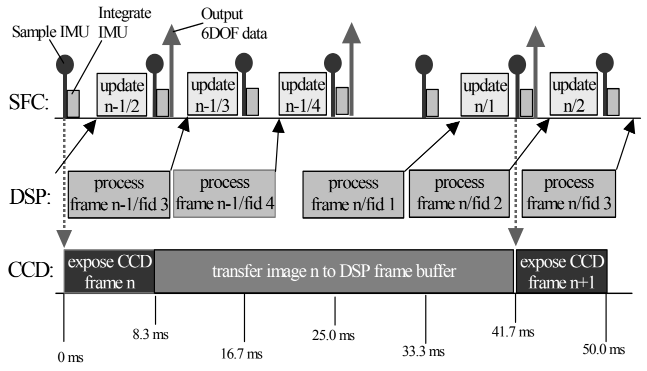

**Figure 6: System timing diagram.**

In Figure 6 one can see a 41.7 ms camera cycle, or 24 frames/sec rate. This frame rate cannot be increased any further because the CCD needs about 8 ms for exposure, plus about 33 ms for image transfer. We tried a CMOS image sensor, which reduces the image transfer time by downloading only the areas of interest which will be processed, but it's light sensitivity was not good enough to track in dark rooms. Our next version of the smart camera will use a CCD with 18 ms image transfer and ~40 fps.

After taking a picture, it is simultaneously transferred to a frame buffer while leftover fiducials from the previous frame (in memory) are processed. Once part of the current frame has been transferred, the DSP may begin processing fiducials from the current image, if they are located in the top part of the image, which has already been transferred. As Figure 6 shows, this mixed strategy results in the SFC receiving vision measurements having varying latencies from about 20 ms for fiducials near the top of the image to >50 ms for those at the bottom. However, these optical measurement latencies do *not* directly affect the tracking system latency, which is calculated as the time to get an inertial data record from the IMU, integrate it, and transmit the 6 DOF pose record out the Ethernet. This tracking system latency works out to a steady 4 ms, and prediction based on the inertially-measured motion derivatives can be used to compensate for this as well as subsequent rendering delays. If incorporated by the Kalman filter when received, the delayed measurements would drag the pose estimate back towards the pose when the images were captured. To prevent this, we retroactively incorporate the vision measurements into the Kalman filter at the point in the past when the vision measurement was fresh, using the stored recent history of the filter states and matrices. Thus after the measurement is received, we make corrections to the state and covariance equal to the corrections that would have been made if the measurement had been incorporated when the picture was taken plus an adjustment to compensate for their evolution since that time.

## 4. Demonstrations & results

A previous version of the system running on a laptop was first publicly demonstrated at the International Symposium on Augmented Reality (New York, Oct. 2001). To demonstrate the AR capability, we interfaced it into the backpack-based Battlefield Augmented Reality System (BARS) developed at the Naval Research Lab. The coordinates of the major corners of the room were surveyed along with the tracker fiducials, and used by the BARS software to overlay lines on the room edges. Attendees wore the backpack and wandered around the room looking through a Sony Glasstron optical see-through display that refreshed the wireframe overlays at a 60 Hz update rate. Unfortunately, we did not have means to capture any through-the-lens video of the demo, or collect any data at the conference, but our subjective impression was that the lines seemed to be registered to the actual room edges within about a centimeter most of the time, even as the person was walking or turning. We do not know if this is primarily tracking error, HMD calibration error, or both.

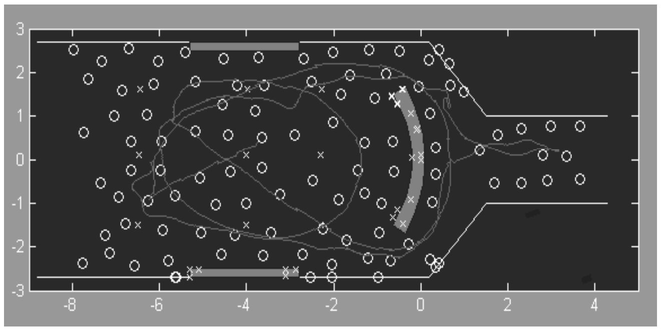

**Figure 7: Floor plan showing dimensions of lobby and adjoining tracked corridor in meters. Circles represent fiducials and crosses are certain architectural corner points we surveyed for future testing. The faint wiggly line is the path of the walk-through described below.**

In order to get some further experience with the system, we installed a constellation of 103 fiducials in the lobby and adjoining corridors of our building. The tracked area includes a large open space, and at one end there is a curved divider that hides a narrow hallway going back to the bathrooms. The space provided an opportunity to test the system with a wide variety of lighting conditions, from bright sun at the front of the lobby to near darkness in the rear corridor with the lights off, and a mixture of many different ceiling heights and wide open spaces as well as narrow maze-like corridors which are impossible to track with other technologies. Figure 7 shows the layout of the space, and the surveyed fiducial positions. A pole with a flat disc on the end was used for putting up the fiducials with double-sided tape on the high ceiling with no ladder in about 90 minutes. 90% of the fiducials are on the ceiling, because we are tracking with an upward-looking camera, but we added a few on the walls to demonstrate that their arrangement is flexible and up to the discretion of the user.

Figure 8 shows a person wearing the VIS-Tracker on an HMD and walking around the lobby test area with fiducials installed. As can be seen, we typically tend to place most of the fiducials on the ceiling when the goal is to allow a user to walk around throughout a large facility with continuous tracking, but they could also be placed exclusively on the walls, with the head-mounted camera facing forwards, or for the ultimate performance, one could have a mixture of wall- and ceiling-mounted fiducials, and a system which incorporates two cameras facing up and forwards.

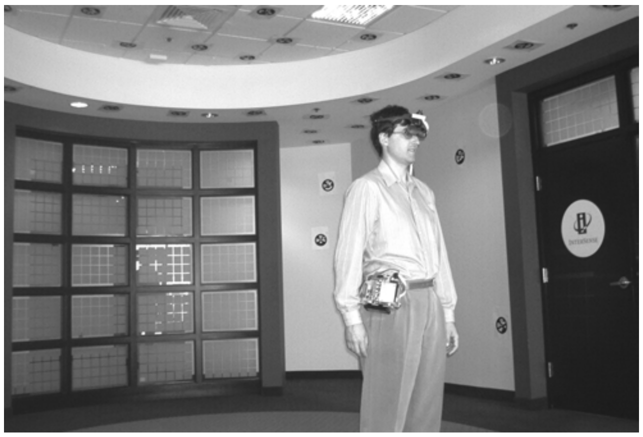

**Figure 8: Using VIS-Tracker in lobby test area.**

The main results of interest from this demo are shown in an accompanying videotape[1] captured from the image processing board which shows what the upward-looking tracking camera sees, overlaid with graphics that show the search box in which it searches for a fiducial on each frame, and a cross showing the location and size of the fiducial it actually found, if any. Unfortunately, the stills included here simply can't convey how the system works, because they only overlay one fiducial per frame, so please see the video for a better understanding of this section.

## 4.1. Tracking experiment 1: wide-area walk-through

Once the fiducials were installed and surveyed, we turned on the tracker and walked all around the large multi-room space observing performance. Figure 8 shows one of us partaking in this novel experience, and the squiggly line in Figure 7 shows the path he followed, as logged by the tracker. Subjectively, we found excellent coverage with no trouble spots. While all the "mobile" AR demos we've seen before involved donning large backpack contraptions, this one was so light you could forget about it and run around freely through the space. If you were to add a virtual environment application to run on the Transmeta CPU board (or on a separate wearable computer), there would be almost no limit to the number of players who could co-habitate the virtual space, since the tracking infrastructure is entirely passive. We have not yet integrated a 3-D graphics application in our system, but plan to shortly. For now, the HMD is used to mount the tracker on the subject's head, and to display output of the smart camera, which shows how the tracker is working.

If you watch the video of this experiment, you will see that the person started running laps around the lobby, sometimes going behind the frosted glass screen where there are only a small number of fiducials that could be surveyed, and yet the tracking continued to operate. Most computer vision systems need to track the fiducials from one frame to the next, which is impossible if they move all the way across the frame or even leave it. In our hybrid system, we instead track the 6-DOF pose of the camera using inertial sensors, and then predict the locations where fiducials will show up in the new frame, their size (based on distance), and the uncertainty in the 2-D image location as a function of the uncertainties in 6-DOF pose states maintained by the Kalman filters in the SFC. Based on these factors, the smart camera driver computes a search box which is just big enough to contain the expected fiducial. In the video, you can see the search boxes growing and shrinking as the camera moves closer and further from the fiducials, and growing to accommodate greater prediction uncertainty after a very active maneuver, but shrinking quickly after the maneuver stops and several measurements are made. There is no lower limit on the image size of a fiducial for it to be used in tracking, but it must be at least 20 x 20 pixels to read the barcode during acquisition. Figure 9 shows some video stills taken from floor level to 3 meters high, to illustrate how the size of the search box adapts to the expected size of the fiducial.

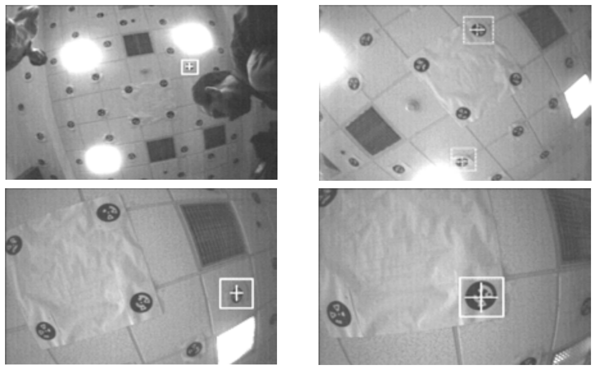

**Figure 9: Stills from the video: Tracking at different heights.**

[1] www.isense.com/support/downloads/vistracker.zip

During the walk-through, the tracker logged data about position, orientation, estimation uncertainties, angular rates, linear accelerations, and fiducial prediction errors. The AR registration accuracy for the system would be about 1-2 pixels r.m.s. when the person is walking and a little more once they start running. This statement assumes a video see-through AR system with the overlays added to the same camera that does the tracking.

### 4.2. Tracking experiment 2: varied lighting conditions

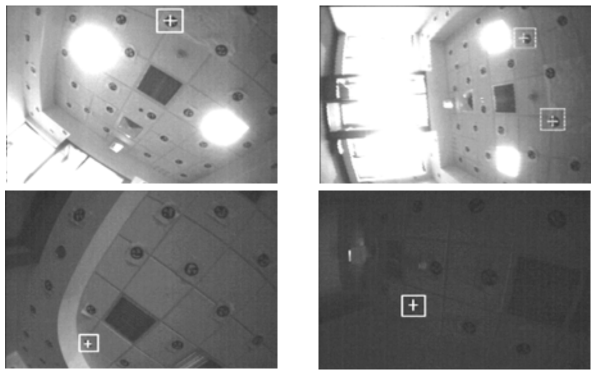
Figure 10: Stills from the video: Different lighting conditions.

A great deal of our development effort was invested in making sure the image processing algorithms for the smart camera are robust to a wide range of lighting conditions that one might encounter in practical applications [7]. To test the system's robustness, we did an experiment where we walked through the space with the lights switching on and off, and made a point to aim the camera towards the sunny glass wall in the front of the lobby, and then to go into the back corridor with the lights off, which was very dark. It kept on tracking through this entire procedure. Figure 10 shows a few stills from this experiment. The final still illustrates that even when the image captured by the CCD is so dark that a human can't see anything but black, the local image processing can still find and accurately report the centroid of the fiducial.

### 4.3. Tracking experiment 3: occlusions

Because of the hybrid design, the tracker is relatively robust to line-of-sight occlusions. Figure 11 shows some stills from a video segment which demonstrates that if you block most but not all of the fiducials, it will continue to try all the potential fiducials, and use measurements from any that are not blocked.

When it looks for a fiducial that is partially occluded, as in the second still, it recognizes that the fiducial is not whole, so it does not report a potentially corrupted value that could de-stabilize the tracking. As long as there are at least two fiducials not blocked, it can keep tracking indefinitely, and with just one it can last a few minutes. Even when the entire lens is covered, as in the third still, it can keep tracking using the inertial sensor for about 5 seconds, although the position drifts during that time.

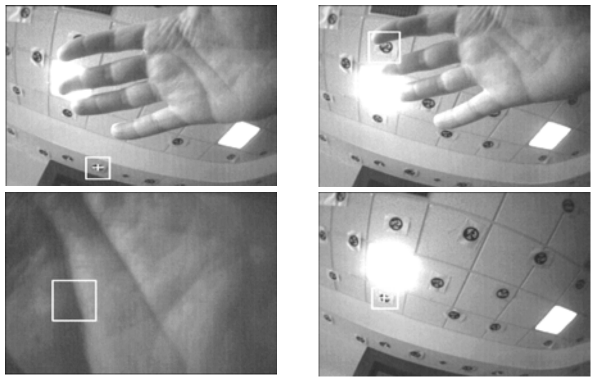
Figure 11: Stills from the video: Tracking during occlusion.

### 4.4. Tracking experiment 4: rapid camera motions

To evaluate the robustness of the tracker, we made another video featuring several quick rotations and translations with the sensor held in the hand, which are even more abrupt than with it mounted on an HMD. Data logged during the experiment shows that the tracking can continue through peak rotation rates over 1000 deg/s and peak translational accelerations over 25 m/s$^2$.

### 4.5. Auto-mapping experiment

We demonstrated the auto-mapping process running in real-time on the above described hardware at the International Conference on Robotics and Automation (ICRA 2002, Washington, D.C, May,2002). We are currently limited to N=100 fiducials at a time, because this adds 3*N beacon position states to the simultaneous tracking and environment mapping filter, which is about all the CPU can process in real time. The procedure is to start with a seed constellation of four known fiducials and begin tracking by pointing the camera towards this seed area. Once it starts tracking, the operator then points the central region of the camera towards other fiducials that are not yet in the map (you can tell because they don't flash overlays). After about a second, you will see the new fiducial gets highlighted and decoded, then automatically enters the map with an initial position estimate which is calculated based on the u and v bearing angles from the camera and a crude estimated distance based on size. Thereafter, it gets scheduled and used like all the other fiducials in the map, and the Kalman filter rapidly improves on its initial rough position estimate each time a new measurement is made. We are working on further automating this process so that the operator does not need to point the camera at the new fiducials, and on developing distributed data fusion algorithms to accommodate large map sizes consistently and seamlessly.

## 5. Discussion

We have described a unique hybrid vision/inertial self-tracking system that is nearing readiness for productization. Compared to vision-only trackers such as the popular AR Toolkit [2], it has several important distinctions. First, it tracks its pose relative to a unified world coordinate system, fusing together measurements from all the fiducials that are visible (AR Toolkit separately determines the camera pose relative to each visible fiducial). Second, it performs real-time auto-mapping of the fiducial constellation while it tracks. Third, it provides 120 pose updates per second (or 180 using a faster CPU), genlocked to the video rendering rate, with guaranteed constant latency and inertial-quality prediction. Fourth, it uses a novel fiducial design and image processing strategy [7] which allows it to quickly decode over 30,000 different markers under more varied lighting conditions and with higher centroid accuracy. Fifth, it relies primarily on inertial tracking, so it can handle rapid camera motions and occlusions much more robustly.

There are two avenues for improvement under consideration. The first concerns further miniaturization and accuracy improvement. The system as implemented with a single smart camera can overlay objects within the field of view of that camera within a few pixels, but if it were used to generate overlays in an HMD or television camera pointing in a different direction from the tracking camera, the errors would be larger. We can make some improvements by better compensating the single lens, but we expect much higher absolute pose accuracy to be obtained by using two smart cameras pointed in orthogonal directions. Fortunately, we think we can do this and also reduce the size of both the sensor head and the belt unit by using a much smaller smart camera now in development.

The second project is reducing the reliance on artificial fiducials. Figure 7 shows a very high density of fiducials – about 1.7 per square meter. The only reason we used so many is so that at any location the system can find and read 4 barcodes and re-acquire. For tracking purposes, far fewer would be sufficient. As a first step, we are trying to get the system to acquire from one fiducial, so that we can install a much sparser constellation.

As a second step, we are planning to make the system use naturally occurring corners, which are plentiful in buildings, as fiducials during tracking. Then it will only be necessary to put one fiducial per location where the system needs to be able to initialize. If the tracking almost never gets lost, this might be as few as one per large room.

In the long run, we hope to eliminate the artificial fiducials altogether. This poses relatively minor difficulties for tracking (mainly data association algorithms to handle clusters of natural features that are closely spaced in the image), but adds considerable complexity to the initial acquisition process. After processing two images taken from different locations to find all the corners in the frame, labeled with any size, type or color characteristics that can be measured, the software would establish point correspondences between the two images, triangulate the two rays to each point to determine all the 3-D point locations relative to the second camera pose, then search through the entire feature database looking for the best match of this 3-D point cloud to a subset of points from the database, using an iterative closest point (ICP) algorithm.

To facilitate natural feature tracking outdoors and in other specialized environments, we are developing an open sensor driver API to allow researchers to interface their own computer vision hardware and image processing algorithms into the sensor fusion core.

## 6. Bibliography

[1] Azuma, R., Hoff, B., Neely, H. and Sarfaty, R. (1999). A Motion Stabilized Outdoor Augmented Reality System. Proc. of IEEE VR-1999, pp. 252-259.

[2] Billinghurst, M. and Kato, H. (1999). Collaborative Mixed Reality. Proc. of ISMR-1999, pp. 261-284.

[3] Bishop, G. (1984). Self-Tracker: A Smart Sensor on Silicon. Unpublished PhD. Dissertation, Dept. of C.S., UNC-Chapel Hill.

[4] Foxlin, E. (2002). Generalized Architecture for Simultaneous Localization, Auto-Calibration, and Map-building. Proc. of IEEE/RSJ Intl. Conf. On Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland.

[5] Hoff, W., Nguyen, K. and Lyon, T. (1996). Computer Vision-Based Registration Techniques for Augmented Reality. Proc. of IRCV, SPIE Vol. 2904, pp. 538-548.

[6] Kanbara, M., Fujii H., Takemura, H. and Yokoya, N. (2000). A Stereo Vision-Based Augmented Reality System with Inertial Sensor. Proc. of ISAR-2000, pp 97-100.

[7] Naimark, L. and Foxlin, E. (2002). Circular Data Matrix Fiducial System and Robust Image Processing for a Wearable Vision-Inertial Self-Tracker. Proc. of Intl. Symposium on Mixed and Augmented Reality (ISMAR-2002), Darmstadt, Germany.

[8] Neumann, U. and Cho, Y. (1996). A Self-Tracking Augmented Reality System. Proc. of ACM VRST 96.

[9] Stricker, D., Klinker, G. and Reiners, D. (1998). A fast and robust line-based optical tracker for augmented reality applications. Proc. of IWAR-1998.

[10] Welch, G. (1995). Hybrid Self-Tracker: An Inertial/Optical Hybrid Three-Dimensional Tracking System, UNC- Chapel Hill, Dept. of C. S., NC, USA TR95-048, 1995.

[11] Yokokohji, Y., Eto, D. and Yoshikawa, T. (2001). It's Really Sticking! –Dynamically Accurate Image Overlay Through Hybrid Vision/Inertial Tracking, Proc. of ISMR-2001, pp. 196-197.

[12] You, S. and Neumann, U. (2001). Fusion of Vision and Gyro Tracking for Robust Augmented Reality Applications. Proc. of IEEE VR2001, pp. 71-78.