Proceedings of IEEE and ACM International Symposium on Augmented Reality 2000 (Munich, Germany, 5-6 October 2000), 159-164.

Autocalibration of an Electronic Compass in an Outdoor Augmented Reality System

Bruce Hoff [§] Ronald Azuma
BioDiscovery, Inc. *HRL Laboratories, LLC*
hoff@biodiscovery.com azuma@HRL.com
[§] work done while employed at HRL Laboratories

Abstract

Accurate registration in an Augmented Reality system requires accurate trackers. An electronic compass can be a valuable sensor in an outdoor Augmented Reality system because it provides absolute heading estimates. However, compasses are vulnerable to distortion caused by environmental disturbances to Earth's magnetic field. These disturbances vary with geographic location and are not trivial to model. Static calibration methods exist but these require an explicit initial calibration step and do not adapt to changing distortion patterns. This paper describes in detail an autocalibration method that compensates for changing compass distortions. With minimal user input, it automatically measures and adjusts the calibration table used to correct the compass output. Autocalibration uses redundant heading information computed from rate gyroscopes. We demonstrate that autocalibration converges to solutions similar to a distortion table that was manually measured with a mechanical turntable. With autocalibration, an electronic compass can provide useful measurements even as the user walks around through areas of varying magnetic distortion.

1. Motivation

A basic objective of any Augmented Reality (AR) system is the accurate alignment of virtual objects over their real counterparts. Achieving this goal requires accurate real-time sensors to measure the user's position and orientation. Ideally, an AR system would support accurate registration in any operating environment, including outdoors.

In an outdoor situation, an electronic compass can be an important part of an effective hybrid tracking system for an AR display. The compass yields an absolute measurement of the yaw component of orientation by sensing Earth's magnetic field. In an indoor situation, the AR system may have many absolute references available, perhaps through fiducial markers placed at known locations. But such absolute references are harder to provide in an outdoor situation, since the system designer has little control over the environment. Therefore, a sensor like an electronic compass that can provide absolute information in an outdoor situation without requiring any modification of the environment is potentially valuable.

However, compasses are vulnerable to distortion, because Earth's magnetic field is a weak signal. Besides magnetic declination (an offset caused by the geographical separation of Earth's true north pole and magnetic north pole), distortions can be introduced by any nearby substance that disturbs the magnetic field. Empirical measurements taken at several different locations around Malibu suggest that the peak-to-peak distortions can be several degrees, and the distortion pattern can appear as complicated curves that are not easily modeled.

These distortions will exist even with a perfect compass. They are endemic properties of the environment. Although electronic compasses vary in sensitivity and quality, buying a perfect compass (if such a device existed) would not solve this problem.

Furthermore, these distortions vary with geographic location. A user walking around outdoors would experience different distortion patterns at different locations. This problem is especially acute in areas with large magnetic disturbances, such as an urban environment. Therefore, calibrating the system for a single location is insufficient to solve the entire problem.

This paper describes in detail an approach to autocalibrate an electronic compass. The method is designed to automatically adjust, with minimal user input, the calibration table used to correct the compass output. Thus, as the user walks around, this method can automatically compensate for the changing distortions in the magnetic field. With this method, an electronic compass can still be a useful source of information to an outdoor AR system even as the user walks around.

2. Previous work and contribution

Standard approaches to calibrating electronic compasses involve either building a mathematical model of the distortion and finding appropriate parameters to fit that model to the experienced distortion, or building a calibration table of the distortion and applying that to correct the distortion. These approaches involve an explicit calibration step and measure the distortion at one location. While they may work well for vehicles such as aircraft, since those operate far away from the ground-based sources of magnetic distortion, these approaches are not well-suited for a walking user on the ground who will be exposed to many changing distortion patterns.

Some outdoor AR systems have been built and have used electronic compasses as part of their tracking system. An earlier system at HRL [1] uses an electronic compass as part of a gyro-compass hybrid tracker, but the compass calibration in that system was an explicit, manual measurement of the distortion at individual locations, and it had no ability to compensate for changing distortions. [2], [4] and [6] also have outdoor AR tracking systems, some of which use a compass, but none explicitly describe a compass calibration routine.

Our work is related to previous autocalibration efforts for virtual environment systems. Their general approach is to use multiple sensors to provide redundant measurements of a particular property and use the redundancy to correct for parameters that control the output of the tracker. [3] describes approaches for autocalibration involving magnetic and optical trackers. The Single Constraint at a Time (SCAAT) filter [7] includes autocalibration as a feature and has been demonstrated on a custom optical tracker at UNC Chapel Hill. While the basic idea behind autocalibration may be simple, finding effective methods of employing it and demonstrating it on working systems remains a nontrivial problem. Our work differs by designing an autocalibration method for an electronic compass rather than 6-DOF magnetic and optical trackers, requiring a different algorithm and approach.

The contribution of this paper is in the description of a new method to autocalibrate an electronic compass, to make such a tracker useful for outdoor AR applications. This calibration approach avoids an explicit calibration step in advance and adapts to changing magnetic distortions as the user walks around, while previous methods are intended for constant distortion patterns.

3. Method

3.1. Overview

Effective autocalibration requires sources of redundant information, either through multiple sensors or multiple measurements of a known invariant. In our system, we use two types of sensors to measure head orientation: three rate gyroscopes (Systron Donner GyroChip II, model QRS14-500-103) and one electronic compass orientation sensor (Precision Navigation TCM2-50). The rate gyroscopes measure the angular velocity of orientation, while the TCM2 measures absolute orientation through a combination of an electronic compass (heading) and two tilt sensors (pitch and yaw).

We state the problem as follows: with these sensors, the system must recover an accurate estimation of the head orientation. The sensors will have some noise in their outputs. There is also a significant bias in the electronic compass, due to magnetic disturbances in the environment. In reality, there will be biases in the gyroscopes as well and other types of distortions but for the purposes of this autocalibration routine we assume those other errors are much smaller and can be ignored. Therefore, the estimator must simultaneously compute the orientation and the bias for the electronic compass.

The compass bias varies with head orientation. The bias is not a scalar value, but rather takes the form of a calibration table. The table is indexed by the reported compass heading. To correct the compass output, the estimator looks up the associated bias for that heading and adds that to the output. Since the table has a limited number of entries, the computed bias will generally be interpolated between the two nearest table entries.

Our general approach is to modify the Kalman-like estimator used in [1]. We add an explicit bias table as part of the estimator. The method must automatically update the values in the bias table. We define a cost function that has minimum value when the bias estimate matches the true bias. A gradient descent strategy tells the estimator how to adjust the bias values to reduce the cost. With time, the estimator eventually computes good estimates for the bias table. To reduce the computational complexity, we make a significant simplification that empirically works, as demonstrated on real test data.

The rest of this section is organized as follows: Section 3.2 defines the estimator and the relevant variables. The reader can then read the derivation of the method in section 3.3 or skip directly to section 3.4 for the explanation of the actual algorithm. Section 3.5 describes an important pitfall in the method and how to compensate for that weakness.

3.2. Definitions

The estimator is based upon a discrete Kalman filter [5] and is a modification of the estimator used in [1]. Each discrete step i is one millisecond of time, so the estimator updates at 1 kHz. The estimator maintains a state vector $\mathbf{x}_i$ which is a 6 by 1 matrix:

$$\mathbf{x}_i = \begin{bmatrix} r_C & p_C & h_C & r_g & p_g & h_g \end{bmatrix}^{\mathbf{T}}$$

where r, p, and h represent the roll, pitch and heading values at that discrete step i. The subscripts c and g denote compass (TCM2) and gyroscope, respectively. The first three values are absolute angles in degrees and the last three are angular rates in degrees per second. $\mathbf{x}_i$ is the true value of the state vector, which of course is not known. We can only estimate the state vector through our sensor measurements, and this estimate is $\hat{\mathbf{x}}_i$.

There are two processes that update the state. The first is the model of system dynamics, represented by:

$$\mathbf{x}_{i+1} = \mathbf{A}_i \mathbf{x}_i + \mathbf{w}_i$$

where $\mathbf{A}_i$ is a 6 by 6 matrix representing how the state vector changes in the absence of any measurements, and

$\mathbf{w}_i$ represents a 6 by 1 matrix of white noise sources. We define $\mathbf{A}_i$ later on in this section. The other process shows the relationship of measurements to the state vector:

$$\mathbf{z}_i = \mathbf{H}\mathbf{x}_i + \mathbf{B}(\mathbf{x}_i) + \mathbf{v}_i$$

where $\mathbf{z}_i$ is a 6 by 1 matrix holding the measurements, $\mathbf{H}$ is a 6 by 6 matrix converting the state vector into the measurements, $\mathbf{B}(\mathbf{x}_i)$ is the state-dependent sensor bias, and $\mathbf{v}_i$ is another white noise vector. In our case, $\mathbf{z}_i$ has the same definition as $\mathbf{x}_i$, so $\mathbf{H}$ is the 6 by 6 identity matrix. Note: in the realtime system there is a time delay between the compass and gyro measurements, which we ignore here for simplicity. See [1] for how that is accounted for in the estimator. We define $\mathbf{B}$ later on in this section.

The estimator combines both the dynamics and the measurement processes into one update. Given a measurement and a previous estimated state, the new estimated state is computed as follows:

$$\hat{\mathbf{x}}_{i+1} = \mathbf{A}_i\,\hat{\mathbf{x}}_i + \mathbf{K}\left(\mathbf{z}_{i+1} - \hat{\mathbf{B}}(\mathbf{A}_i\,\hat{\mathbf{x}}_i) - \mathbf{H}\mathbf{A}_i\,\hat{\mathbf{x}}_i\right)$$

This update is a simplification of the Kalman filter, where we use a constant 6 by 6 Kalman gain matrix $\mathbf{K}$, defined as:

$$\mathbf{K} = \begin{bmatrix} g_c\,\mathbf{I}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & g_g\,\mathbf{I}_{3x3} \end{bmatrix}$$

where $\mathbf{I}_{3x3}$ is the 3 by 3 identity matrix, $\mathbf{0}_{3x3}$ is a 3 by 3 matrix filled with zeroes, and the empirically-determined gain constants are $g_c = 0.05$ and $g_g = 1.0$. Using a constant $\mathbf{K}$ reduces the computation requirements, effectively operating the filter in a steady-state condition.

The derivation of $\mathbf{A}_i$ is lengthy, so we only include the definition here:

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{I}_{3x3} & \Delta t\,\mathbf{A}_{12}(\hat{\mathbf{x}}_i) \\ \mathbf{0}_{3x3} & \mathbf{I}_{3x3} \end{bmatrix}$$

where Δt is the timestep (0.001 seconds) and $\mathbf{A}_{12}$ is a 6 by 6 matrix that translates small rotations in the sensor suite's frame to small changes in the TCM2 variables:

$$\mathbf{A}_{12}(\hat{\mathbf{x}}_i) = \begin{bmatrix} cp\,c^2r/a^2 & a\,sr\,cr\,sp(t^2r + 2/c^2p) & a\,tp\,c^2r/c^2p \\ 0 & a/cp & -a\,tr \\ 0 & a\,tr/cp & a/c^2p \end{bmatrix}$$

where cθ = cos(θ), sθ = sin(θ), tθ = tan(θ). For example, $cp = \cos(p)$ and $t^2r = \tan^2(r)$. r and p are the TCM2 roll and pitch variables in the estimated state vector (i.e. r_c and p_c). Variable a is defined as:

$$a = \frac{1}{\sqrt{1 + t^2p + t^2r}}$$

Our goal is to determine the sensor bias $\mathbf{B}()$. Of course, we can only estimate this, so we compute the estimated bias $\hat{\mathbf{B}}()$. The bias takes the form of a lookup table, and the entry depends on the current estimated state. The lookup table has n entries, where each represents one patch (p_1 to p_n) of heading space. When the heading h_c in the estimated state is in patch p_j, we use the estimated bias vector $\hat{\mathbf{B}}_j$ from the lookup table:

$$\hat{\mathbf{B}}(\mathbf{A}_i\,\hat{\mathbf{x}}_i) = \hat{\mathbf{B}}_j, \text{if } \mathbf{A}_i\,\hat{\mathbf{x}}_i \in p_j$$

Bias vector $\hat{\mathbf{B}}_j$ is a 6 by 1 matrix with only one non-zero setting, since the only bias we are computing is the bias for the compass heading (yaw), b_h.

$$\hat{\mathbf{B}}_j = \begin{bmatrix} 0 & 0 & b_h & 0 & 0 & 0 \end{bmatrix}^{\mathbf{T}}$$

3.3. Derivation

The goal is to compute the entries in the bias table, which should contain the differences between the measured and the true values in the state:

$$\mathbf{B}(\mathbf{x}_i) = \mathrm{E}[\mathbf{z}_i - \mathbf{H}\mathbf{x}_i \,|\, \mathbf{x}_i]$$

This problem would be easy if we knew the true state $\mathbf{x}_i$. Of course, we can only estimate the state, so we must employ an indirect estimation method. Another way to state the problem is to subtract the sensor bias estimate and tune the result so the expected value is zero. We use $\hat{\mathbf{B}}_j$ as the sensor bias vector for the current patch:

$$\mathrm{E}\left[\mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\mathbf{x}_i \,|\, \mathbf{x}_i \in p_j\right] = 0$$

One approach to solving such a parameter estimation problem is to construct an energy function where $\hat{\mathbf{B}}_j$ is the parameter. The value of the energy function is everywhere nonnegative and is minimized when $\hat{\mathbf{B}}_j$ is optimal. We then use a gradient descent search strategy to search for the optimal value of $\hat{\mathbf{B}}_j$ by taking the gradient of this function with respect to $\hat{\mathbf{B}}_j$ and incrementally changing the value of $\hat{\mathbf{B}}_j$ along the direction of the gradient to improve our estimate. One such energy function is:

$$\mathrm{E}\left[\left(\mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\mathbf{x}_i\right)^2 \,|\, \mathbf{x}_i \in p_j\right] = 0$$

It can be shown that this energy function is minimized when $\hat{\mathbf{B}}_j = \mathrm{E}\left[\mathbf{B}(\mathbf{x}_i) \,|\, \mathbf{x}_i \in p_j\right]$, i.e. when $\hat{\mathbf{B}}_j$ is an optimal bias estimate.

However, this candidate energy function requires that we know the true state of the system, $\mathbf{x}_i$, which is unavailable. Instead, we must substitute the estimated state $\hat{\mathbf{x}}_i$, so the energy function becomes:

$$\mathrm{E}\left[\left(\mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\hat{\mathbf{x}}_i\right)^2 \,|\, \hat{\mathbf{x}}_i \in p_j\right] = 0$$

There is a significant pitfall created by this substitution, which is discussed in detail in section 3.5.

Now define:

$$\mathbf{d}_i = \mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\hat{\mathbf{x}}_i$$

Our energy function E_j is the statistical variance of $\mathbf{d}_i$ over all "visits" the algorithm makes to patch p_j:

$$E_j = \tfrac{1}{2}\frac{1}{|p_j|}\sum_{\hat{\mathbf{x}}_i\in p_j}\mathbf{d}_i^{\mathrm{T}}\,\mathbf{d}_i$$

where $|p_j|$ is the number of "visits" to patch p_j and $\tfrac{1}{2}$ is an arbitrary positive constant added for convenience.

The gradient descent algorithm is expressed by the parameter update:

$$\Delta\hat{\mathbf{B}}_j = -\gamma\left(\nabla_{\hat{\mathbf{B}}_j}E_j\right)^{\mathrm{T}}$$

where $\Delta\hat{\mathbf{B}}_j$ is the change to the sensor bias estimate at the patch p_j, γ is a small positive constant controlling the learning rate, and $\nabla_{\hat{\mathbf{B}}_j}E_j$ is the gradient of E_j with respect to $\hat{\mathbf{B}}_j$. This gradient is:

$$\nabla_{\hat{\mathbf{B}}_j}E_j = \frac{1}{|p_j|}\sum_{\hat{\mathbf{x}}_i\in p_j}\left(\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}\right)^{\mathrm{T}}\mathbf{d}_i$$

Since this expression is a sum, we can instead model the change to the sensor bias estimate as a sequence of changes, one for each visit to a particular patch p_j:

$$\Delta\hat{\mathbf{B}}_j = \sum_{\hat{\mathbf{x}}_i\in p_j}\Delta\hat{\mathbf{B}}_j^i$$

$$\Delta\hat{\mathbf{B}}_j^i = -\gamma'\left(\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}\right)^{\mathrm{T}}\mathbf{d}_i \text{ where } \gamma' = \frac{1}{|p_j|}\gamma$$

The task now becomes one of computing the incremental bias update $\Delta\hat{\mathbf{B}}_j^i$, which requires computing $\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}$. We can rewrite $\mathbf{d}_i$ as:

$$\mathbf{d}_i = \mathbf{H}\,\mathbf{e}_{xi} + \left(\mathbf{B}(\mathbf{x}_i) - \hat{\mathbf{B}}_j\right) + \mathbf{v}_i$$

where

$$\mathbf{e}_{xi} = \mathbf{x}_i - \hat{\mathbf{x}}_i$$

Therefore:

$$\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j} = \mathbf{H}\frac{\partial\mathbf{e}_{xi}}{\partial\hat{\mathbf{B}}_j} - \mathbf{I}$$

Computing $\frac{\partial\mathbf{e}_{xi}}{\partial\hat{\mathbf{B}}_j}$ is a lengthy exercise involving a recursive formula. The end result when plugged back into the equation for $\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}$ is:

$$\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j} = \mathbf{H}\left(\sum_{\substack{k\le i\text{ s.t.}\\ \hat{\mathbf{x}}_k\in p_j}}\prod_{l=k}^{i-1}(\mathbf{I}-\mathbf{K}\mathbf{H})\mathbf{A}_i\right)\mathbf{K} - \mathbf{I}$$

Using this expression for $\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}$ would be difficult in a real-time system. It can be computationally intensive, involving the product of a series of matrices, and the total number of terms involved grows with time. As the counter i increases, we must store a longer history of patches visited and matrices $\mathbf{A}_i$, which could require significant storage space. Therefore, it is not attractive to use this expression to compute an exact value for $\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}$.

Instead, it is preferable to find an approximation that is more computationally efficient. We note that matrix $\mathbf{K}$ weighs the contribution from the sensor measurements and ($\mathbf{I}$-$\mathbf{K}$ $\mathbf{H}$) gives partial weight to the system dynamics. When these matrices are iteratively multiplied, they decay the terms. Therefore, values in the past are weighed less and less until they do not contribute significantly. This can be seen by rewriting $\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j}$, with the following assumptions: $\mathbf{H}$=$\mathbf{I}$, $\mathbf{A}_i$=$\mathbf{A}$, and $\mathbf{K}$=$a\mathbf{I}$ where $0 < a < 1$. Then:

$$\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j} = a\left(\sum_{\substack{k\le i\text{ s.t.}\\ \mathbf{x}_k\in p_j}}(1-a)^{i-1-k}\,\mathbf{A}^{i-1-k}\right) - \mathbf{I}$$

The terms where $k \ll i$ will not be significant. To avoid any storage issues, we simplify this greatly by dropping all of the first terms which might require storing past values of $\mathbf{A}_i$, so that

$$\frac{\partial\mathbf{d}_i}{\partial\hat{\mathbf{B}}_j} \approx -\mathbf{I}$$

Using such an approximate gradient often allows gradient descent to find an optimum through a "gradually downhill" path rather than a "directly downhill" path. This simplification makes the entire bias update algorithm very simple and easy to compute in real time. To verify that this simplification still generates reasonable results, we tested this algorithm on real data. Those results are shown in section 4.

With this simplification, the incremental change to the sensor bias estimate at patch p_j reduces to:

$$\Delta\hat{\mathbf{B}}_j^i = \gamma'\left(\mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\hat{\mathbf{x}}_i\right)$$

3.4. Algorithm

First, choose a learning rate γ'. Then for each time step i:

1) Given the estimated state $\mathbf{A}_{i-1}\hat{\mathbf{x}}_{i-1}$, determine the current patch p_j and retrieve the associated bias vector $\hat{\mathbf{B}}_j$ from the lookup table.

2) Estimate the new state for timestep i:

$$\hat{\mathbf{x}}_i = \mathbf{A}_{i-1}\hat{\mathbf{x}}_{i-1} + \mathbf{K}\left(\mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\mathbf{A}_{i-1}\hat{\mathbf{x}}_{i-1}\right)$$

3) Compute a 6 by 1 matrix $\mathbf{d}_i$, representing the difference between the bias-adjusted sensor input and the estimated state:

$$\mathbf{d}_i = \mathbf{z}_i - \hat{\mathbf{B}}_j - \mathbf{H}\hat{\mathbf{x}}_i$$

4) Compute an update to the bias vector, modify the vector and then store it back into the lookup table, using the simplification from section 3.3:

$$\Delta\hat{\mathbf{B}}_j^i = \gamma'\,\mathbf{d}_i$$

3.5. A pitfall

As stated in the derivation, there is a pitfall when computing the difference $\mathbf{d}_i$ using the estimated state $\hat{\mathbf{x}}_i$ rather than the actual state $\mathbf{x}_i$. The autocalibration routine has no ability to detect DC offsets between the estimated state and the actual state. Another way of stating this is the autocalibration routine is good at determining the *shape* of the compass distortion lookup table, but there could be an undetermined DC offset added to all the lookup table values that is needed to get accurate results. Figure 1 gives an example of this effect. This was generated from an artificial data set of 10,000 sensor samples collected over 10 seconds. This data was run through the autocalibration routine 25 times with a learning rate set to 0.05. Figure 1 shows the true compass distortion as the dotted line, and the distortion computed by the autocalibration routine as the solid line. The autocalibration routine has captured the shape of the compass distortion correctly, but there is a 5 degree offset between the two curves.

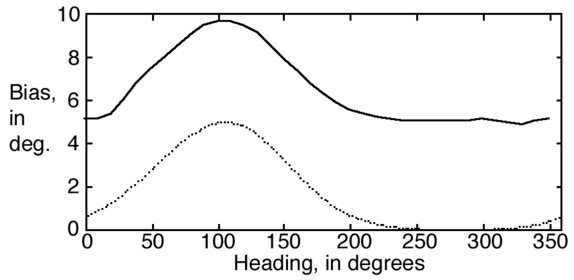

Figure 1: Calibration tables demonstrating offset problem. Dotted line is the true value; solid line is computed by the autocalibration routine.

This pitfall becomes obvious when one considers the nature of the sensors involved. We use a combination of rate gyroscopes and a compass and tilt sensor. The rate gyroscopes do not measure absolute orientation. Only the compass gives absolute heading information. If there is a systematic offset in the compass output, caused for example by magnetic declination, there is no other sensor available to detect and correct this error. The gyroscopes can calibrate relative errors (the shape of the overall curve) but not the absolute error (an overall offset).

To get around this problem, we need to measure the bias correction for one patch in the lookup table. One way to do this is to sight a landmark at a known location. Given the location of the landmark and the observer's known location (measured through a differential GPS receiver), it is easy to compute the true heading. Let's call this measured offset $\hat{\mathbf{B}}^*$ for a landmark observed in patch k. Then we run the algorithm as stated before, but periodically adjust all bias estimates by:

$$\forall j, \hat{\mathbf{B}}_j = \hat{\mathbf{B}}_j - \left(\hat{\mathbf{B}}_k - \hat{\mathbf{B}}^*\right)$$

This subtracts the constant bias error from the entire lookup table.

As the user walks around and the one absolute landmark-based correction $\hat{\mathbf{B}}^*$ becomes less relevant, the overall calibration table may become offset with respect to the true table value. There are two ways to correct this problem. The first is that the user could manually adjust the offset, by occasionally twisting a dial. This is much easier than trying to set an entire calibration table manually. However, this does require user interaction for a task that ideally should be free of user involvement. Therefore, the preferred approach would be to incorporate a video camera and vision-based recognition software that can occasionally find and recognize known landmarks in the environment. Then the calibration system would automatically compute a new landmark-based correction $\hat{\mathbf{B}}^*$ from time to time, as the user walks around, and this will serve to automatically remove the constant bias offset from the entire table.

4. Evaluation

To evaluate how well the autocalibration routine works, we need to compare the calibration table computed by autocalibration against a more accurate measurement of the true magnetic distortion. Therefore, we evaluate this method by first using a mechanical turntable to manually measure the distortion (Figure 2). This turntable is made of Delrin, to avoid adding any additional distortion to the magnetic field. The turntable is accurate to 0.25 degrees. We manually generate a calibration table for comparison purposes by turning the compass to known orientations and recording the average heading measurements.

Then we captured three datasets of head motion, averaging 20 seconds long. These datasets were captured in the same session as the manual calibration. Since they share the same geographic location and are taken at about the same time, we assume the distortion pattern is the same for all datasets.

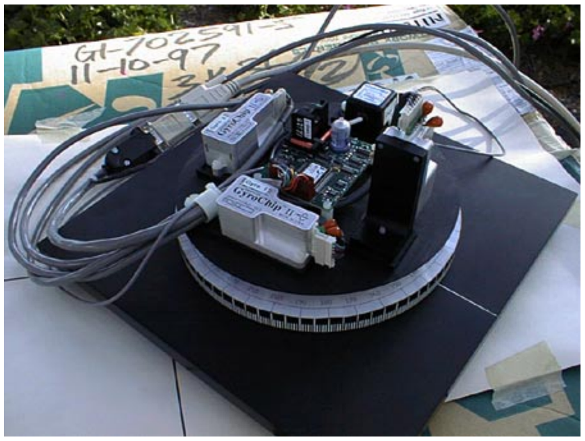

Figure 2: Mechanical turntable with TCM2-50 and gyroscopes mounted

Figure 3 compares the results of autocalibration against the manually measured calibration table. The dotted line is the manually generated calibration table, using the mechanical turntable. The three solid lines are the outputs of the autocalibration routine for the three datasets. Note that the shapes of all four curves are very similar, demonstrating that autocalibration has computed the distortion pattern at this location. The offsets separating the curves can be reduced through approaches discussed in section 3.5.

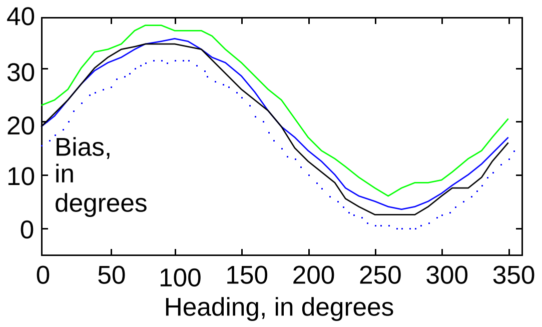

Figure 3: Comparison of manual calibration (dotted line) vs. autocalibration (solid lines)

We intend to do a more thorough demonstration of the autocalibration routine's ability to adapt to changing environments by checking the calibration table after walking to a different geographical location.

5. Future work

We are in the process of adding visual-based corrections in real time to reduce the registration errors in the outdoor AR system (using methods similar to [8]). Such visual corrections would provide another source of absolute measurements and could be used to improve the autocalibration of the compass. The compass could then provide absolute heading during the times that visual features or landmarks in the real environment are unavailable.

Autocalibration might apply to other sensors and calibration parameters in an AR system, providing that sufficient redundancy existed to allow the setup of an autocalibration strategy. This might reduce the calibration setup requirements in AR systems.

6. Acknowledgements

This work was mostly funded by DARPA Warfighter Visualization contracts N00019-97-C-2013 and N00019-99-2-1616.

7. References

[1] R. Azuma, B. Hoff, H. Neely, and R. Sarfaty, "A Motion-Stabilized Outdoor Augmented Reality System", *Proceedings of IEEE Virtual Reality 1999*, Houston, TX, 13-17 March 1999, pp. 252-259.

[2] R. Behringer, "Registration for Outdoor Augmented Reality Applications Using Computer Vision Techniques and Hybrid Sensors", *Proceedings of IEEE Virtual Reality 1999*, Houston, TX, 13-17 March 1999, pp. 244-251.

[3] S. Gottschalk, J. Hughes, "Autocalibration for Virtual Environments Tracking Hardware", *Proceedings of ACM SIGGRAPH 1993*, Anaheim, CA, 1-6 August 1993, pp. 65-72.

[4] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway, "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System", *Computers & Graphics* vol. 23, #6, Elsevier Science, December 1999, pp. 779-785.

[5] F. Lewis, *Optimal Estimation*, John Wiley and Sons, USA, 1986.

[6] W. Piekarski, B. Gunther, and B. Thomas, "Integrating Virtual and Augmented Realities in an Outdoor Application", *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, San Francisco, CA, 20-21 October 1999, pp. 45-54.

[7] G. Welch and G. Bishop. "SCAAT: Incremental Tracking with Incomplete Information", *Proceedings of ACM SIGGRAPH '97*, Los Angeles, CA, 3-8 August 1997, pp. 333-344.

[8] S. You, U. Neumann, and R. Azuma. "Hybrid Inertial and Vision Tracking for Augmented Reality Registration", *Proceedings of IEEE Virtual Reality 1999*, Houston, TX, 13-17 March 1999, pp. 260-267.