On the electromagnetic trackers calibration

Volodymyr Kindratenko
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign (UIUC)

Introduction

If there is a metal near the electromagnetic tracker's transmitter or receiver, the transmitter signals are distorted and the resulting position/orientation measurements contain errors. There are two components that contribute to the total error: *static error* and *dynamic error*. Assuming that the transmitter's position is fixed and the surrounding metal does not move, the static error is a function of the position of the receiver and it can be corrected as long as the magnetic field does not "fold back" on itself. In practice[1,2,3], only the position is corrected typically by trilinear interpolation the use of which is based on the assumption that the distortion of the magnetic field is linear within a given sub-space. It has been suggested[4] that both position and orientation can be corrected by a polynomial fit. There were no attempts found reported in the literature to correct the dynamic error. The goal of this study is to find how the static error for both position and orientation can be corrected with a minimal number of measurements to be taken.

Measurements

Distorted position (x_i, y_i, z_i) and orientation $(\alpha_i, \beta_i, \gamma_i)$ (azimuth, elevation, and roll), measurements are collected at m points of known true location $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ and known true orientation $(\bar{\alpha}_i, \bar{\beta}_i, \bar{\gamma}_i) = (0,0,0)$ of the receiver at each point i and stored in a calibration file.

Position correction

Position correction equations can be generated by fitting a curve or a surface to the data points. In the case of a polynomial fit of the order r, these equations take the general form of

$$\tilde{x} = x + \sum_{j=1}^{n} c_{xj} x^{p'_j} y^{p''_j} z^{p'''_j} \;,\; \tilde{y} = y + \sum_{j=1}^{n} c_{yj} x^{p'_j} y^{p''_j} z^{p'''_j} \;,\; \tilde{z} = z + \sum_{j=1}^{n} c_{zj} x^{p'_j} y^{p''_j} z^{p'''_j}$$

[1] S. Bryson, *Measurement and Calibration of Static Distortion of Position Data from 3D Trackers*, in the Proceedings of SPIE Conference on Stereoscopic Displays and Applications III, San Jose, CA, February 1992, pp. 244-255.
[2] M. Ghazisaedy, D. Adamczyk, D. Sandin, R. Kenyon, and T. DeFanti, *Ultrasonic calibration of a magnetic tracker in a virtual reality system*, in the Proceedings of IEEE VR Annual International Symposium (VRAIS '95), Research Triangle Park, NC, March 1995.
[3] M. Czernuszenko., D. Sandin, T. DeFanti, *Line of Sight Method for Tracker Calibration in Projection-Based VR Systems*, in the Proceedings of 2nd International Immersive Projection Technology Workshop, Ames, Iowa, May 11-12, 1998.
[4] Ascension Technology Corporation: ftp://ftp.ascension-tech.com/technical/curvefit.wp

where c_{xj}, c_{yj}, and c_{zj} are polynomial coefficients; n is the number of terms in the fitting polynomial that depends on the order r of the polynomial and is equal to the number of permutations $\{p'_j, p''_j, p'''_j\}$ where p'_j, p''_j and p'''_j are powers of the corresponding terms such that *i)* $p'_j, p''_j, p'''_j \in \{0,1,\ldots,r\}$, *ii)* $p'_j + p''_j + p'''_j \le r$ for any $j = 1,\ldots,n$, and *iii)* all permutations $\{p'_j, p''_j, p'''_j\}$ are unique; $\tilde{x}$, $\tilde{y}$, and $\tilde{z}$ are corrected values of the distorted measurements x, y, and z. For example, the second-order polynomial for x coordinate may look like this:

$$\tilde{x} = x + c_{x1}x^2 + c_{x2}y^2 + c_{x3}z^2 + c_{x4}xy + c_{x5}xz + c_{x6}yz + c_{x7}x + c_{x8}y + c_{x9}z + c_{x10}$$

where the number of terms in the polynomial $n = 10$, powers $p'_1 = 2$, $p''_1 = 0$, $p'''_1 = 0$, and so on.

In order to build the fitting polynomials, the polynomial coefficients c_{xj}, c_{yj}, and c_{zj} must be computed. It can be done by solving the following simultaneous equations built for each coordinate x, y, and z (all the equations below are given for x coordinate only, equations for two other coordinates are similar):

$$\begin{cases} c_{x1}x_1^{p'_1}y_1^{p''_1}z_1^{p'''_1} + \ldots + c_{xj}x_1^{p'_j}y_1^{p''_j}z_1^{p'''_j} + \ldots + c_{xn}x_1^{p'_n}y_1^{p''_n}z_1^{p'''_n} = \bar{x}_1 - x_1 \\ \ldots \\ c_{x1}x_i^{p'_1}y_i^{p''_1}z_i^{p'''_1} + \ldots + c_{xj}x_i^{p'_j}y_i^{p''_j}z_i^{p'''_j} + \ldots + c_{xn}x_i^{p'_n}y_i^{p''_n}z_i^{p'''_n} = \bar{x}_i - x_i \\ \ldots \\ c_{x1}x_m^{p'_1}y_m^{p''_1}z_m^{p'''_1} + \ldots + c_{xj}x_m^{p'_j}y_m^{p''_j}z_m^{p'''_j} + \ldots + c_{xn}x_m^{p'_n}y_m^{p''_n}z_m^{p'''_n} = \bar{x}_m - x_m \end{cases}$$

that can be written in the matrix form $\mathbf{At} = \mathbf{b}$ as

$$\begin{pmatrix} a_{1,1} & \ldots & a_{1,j} & \ldots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i,1} & \ldots & a_{i,j} & \ldots & a_{i,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m,1} & \ldots & a_{m,j} & \ldots & a_{m,n} \end{pmatrix} \begin{pmatrix} t_1 \\ \ldots \\ t_j \\ \ldots \\ t_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \ldots \\ b_i \\ \ldots \\ b_m \end{pmatrix}$$

where

$$a_{i,j} = x_i^{p'_j}y_i^{p''_j}z_i^{p'''_j},\ t_j = c_{xj}, \text{ and } b_i = \bar{x}_i - x_i.$$

The type of solution the simultaneous equations may have is solely determined by the ranks of the matrix $\mathbf{A}$ and the augmented matrix $\mathbf{A} \mid \mathbf{b}$: if $rank(\mathbf{A}) \neq rank(\mathbf{A} \mid \mathbf{b})$, the system has no solutions, if $rank(\mathbf{A}) = rank(\mathbf{A} \mid \mathbf{b}) = n$, the system has a trivial solution; if $rank(\mathbf{A}) = rank(\mathbf{A} \mid \mathbf{b}) < n$, the system has many solutions.

Simultaneous equations can be solved numerically using so-called *Gaussian elimination* method. The idea behind the method is to transform the augmented matrix $\mathbf{A} \mid \mathbf{b}$ into an *equivalent* triangular matrix applying *equivalent* transformations such as *i*) moving the rows of the matrix, *ii*) multiplying them by a constant that is not equal to zero, and *iii*) adding rows to each other. Once a triangular matrix is obtained, the solution of the system can be computed applying back substitution.

Typically, the number of equations, which is equal to the number of measurements m, is much larger than the number of terms n in the polynomial. Taking into account the fact that the magnetic field distortion is typically non-linear, the solution of the resulting simultaneous equations typically does not exist. That is why we can only fit the surface to the data points (e.g., in the least square sense) instead of computing the exact solution. The idea is to find a set of polynomial coefficients $\hat{c}_{xj}$ such that polynomial

$$\hat{c}_{x1} x^{p_1'} y^{p_1''} z^{p_1'''} + \ldots + \hat{c}_{xj} x^{p_j'} y^{p_j''} z^{p_j'''} + \ldots + \hat{c}_{xn} x^{p_n'} y^{p_n''} z^{p_n'''}$$

fits the error $\tilde{x} - x$ in the least square sense. Coefficients $\hat{c}_{xj}$ can be computed by solving simultaneous *normal equations* build from the known true and measured values:

$$\begin{cases} \sum_{j=1}^{n} \hat{c}_{xj} \sum_{i=1}^{m} a_{i,j} a_{i,1} = \sum_{i=1}^{m} b_i a_{i,1} \\ \ldots \\ \sum_{j=1}^{n} \hat{c}_{xj} \sum_{i=1}^{m} a_{i,j} a_{i,k} = \sum_{i=1}^{m} b_i a_{i,k} \\ \ldots \\ \sum_{j=1}^{n} \hat{c}_{xj} \sum_{i=1}^{m} a_{i,j} a_{i,n} = \sum_{i=1}^{m} b_i a_{i,n} \end{cases}$$

where $k = 1, \ldots, n$. Once polynomial coefficients $\hat{c}_{xj}$ are known, they can be used to compute the corrected value $\tilde{x}$ of x coordinate:

$$\tilde{x} = x + \sum_{j=1}^{n} \hat{c}_{xj} x^{p_j'} y^{p_j''} z^{p_j'''}$$

Correction equations for two other coordinates can be build in the same way using y_i, $\bar{y}_i$, z_i, and $\bar{z}_i$.

Orientation correction

Unfortunately, the above-described approach cannot be directly applied to correct orientation distortions. It has to be modified to take into account the nature of the Euler angles that cannot

be simply added together, rather they must be rotated.

Since the measurements were performed in such a way that $(\bar{\alpha}_i, \bar{\beta}_i, \bar{\gamma}_i) = (0,0,0)$ for $\forall i$, we can build and use fitting polynomials in the form

$$\alpha^* = \sum_{j=1}^{n} \hat{c}_{\alpha j} x^{p'_j} y^{p''_j} z^{p'''_j}, \quad \beta^* = \sum_{j=1}^{n} \hat{c}_{\beta j} x^{p'_j} y^{p''_j} z^{p'''_j}, \quad \gamma^* = \sum_{j=1}^{n} \hat{c}_{\gamma j} x^{p'_j} y^{p''_j} z^{p'''_j}$$

that fit the errors α^*, β^*, and γ^* instead of fitting the actual values of the Euler angles. Their coefficients $\hat{c}_{\alpha j}$, $\hat{c}_{\beta j}$, and $\hat{c}_{\gamma j}$ can be computed in the same way as those in the case of position fitting.

Once the errors of α^*, β^*, and γ^* at a given measured location (x, y, z) are computed, the corrected values $\tilde{\alpha}$, $\tilde{\beta}$, and $\tilde{\gamma}$ can be obtained rotating (α, β, γ) by $(\alpha^*, \beta^*, \gamma^*)$. A correction rotation matrix is formed from $(\alpha^*, \beta^*, \gamma^*)$:

$$\mathbf{M}^* = \begin{pmatrix} \cos\beta^* \cos\alpha^* & \cos\beta^* \sin\alpha^* & -\sin\beta^* \\ \sin\gamma^* \sin\beta^* \cos\alpha^* - & \cos\gamma^* \cos\alpha^* + & \\ -\cos\gamma^* \sin\alpha^* & +\sin\gamma^* \sin\beta^* \sin\alpha^* & \sin\gamma^* \cos\beta^* \\ \sin\gamma^* \sin\alpha^* + & \cos\gamma^* \sin\beta^* \sin\alpha^* - & \\ +\cos\gamma^* \sin\beta^* \cos\alpha^* & -\sin\gamma^* \cos\alpha^* & \cos\gamma^* \cos\beta^* \end{pmatrix}$$

A measurement rotation matrix $\mathbf{M}$ is formed in the same way using (α, β, γ) and is multiplied by the correction rotation matrix $\mathbf{M}^*$. The resulting matrix $\tilde{\mathbf{M}} = \mathbf{M}\mathbf{M}^*$ is then used to compute corrected Euler angles $\tilde{\alpha}$, $\tilde{\beta}$, and $\tilde{\gamma}$:

$$\tilde{\alpha} = \arctan\frac{\tilde{\mathbf{M}}_{1,2}}{\tilde{\mathbf{M}}_{1,1}}, \quad \tilde{\beta} = -\arcsin\tilde{\mathbf{M}}_{1,3}, \quad \tilde{\gamma} = \arctan\frac{\tilde{\mathbf{M}}_{2,3}}{\tilde{\mathbf{M}}_{3,3}}$$

Numerical analysis

Proposed technique was tested with a Flock of Birds (Ascension Technology Corporation) tracking system installed in the CAVE[5] at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. First, a calibration table was build by placing

[5] Cruz-Neira,C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., and Hart, J.C., *The CAVE: Audio Visual Experience Automatic Virtual Environment*, Communications of the ACM, Vol. 35, No. 6, June 1992, pp. 65-72.

the receiver at 100 points of known location and collecting the values reported by the tracking system. Initially the receiver was placed at the point $(-2, 2, -2)$, which is located near the left-down corner of the CAVE, and then was moved on the 2 feet grid until the entire CAVE space was not covered. Later some of the points were excluded from the calibration file because the reported position values were out of workable range of the transmitter. The resulting calibration table containing 88 points is given in the Appendix A. Second, a validation table was acquired by measuring 64 points 5 feet from the floor 1 foot apart starting at the point $(-3.49, 5, -3.5)$. This table is shown in the Appendix B. Calibration table is needed to build the fitting polynomials, validation table is needed to verify the accuracy of the fitting. All the measurements were recorded in the CAVE coordinate system the origin of which is in the middle of the floor 5 feet away from the left, right, and front walls, x axis is oriented towards the right wall, y axis points up, and z axis points away from the front wall. Distances were measured in *feet*, angles were measured in *degrees*, but internally they were converted to *radians*. Transmitter's location in the CAVE coordinate system was $(0.06, 8.42, -2.32)$.

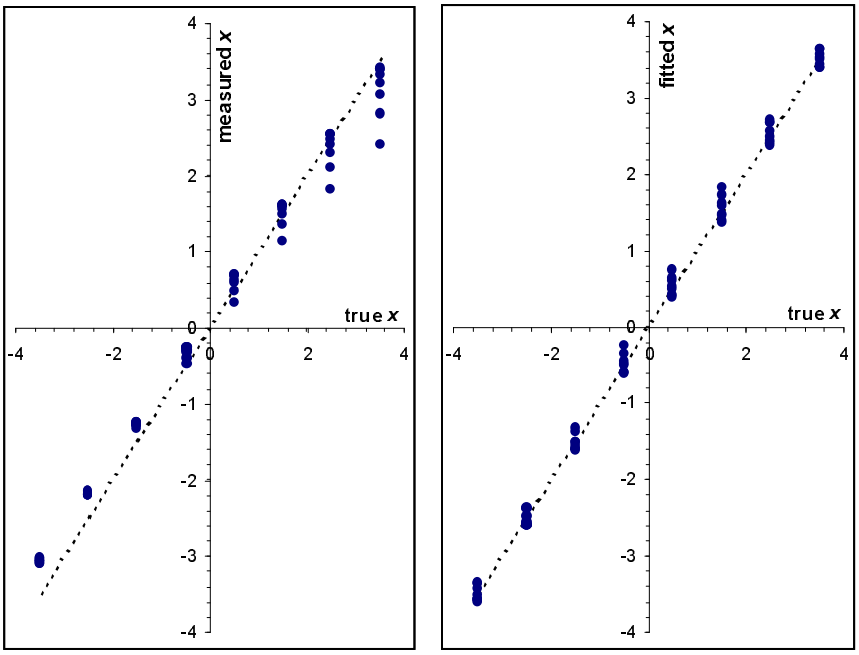
Polynomial coefficients were build from the calibration table for the polynomials of the order 2, 3, 4, and 5. Measurements from the validation table were fitted by the derived polynomials and the resulting corrected values of the position and orientation were used to calculate some statistics regarding the quality of the fit. The table below summarizes the results. Average, standard deviation, min, and max were computed for the absolute values of the differences (residuals) between true values of x, y, z, α, β, and γ and those reported by the tracker (rows $\Delta|x|$, $\Delta|y|$, $\Delta|z|$, $\Delta|\alpha|$, $\Delta|\beta|$, and $\Delta|\gamma|$) and for distances between the points defined by the true values of x, y and z and those reported by the tracker (row $\Delta|d|$). The same computations were performed after fitting the data by the polynomials of the 2nd, 3rd, 4th, and 5th order.

variable		non-fitted	fitting polynomial order			
			2	3	4	5
$\Delta\|x\|$	avr ± std	0.245 ± 0.186	0.296 ± 0.165	0.1 ± 0.061	0.129 ± 0.092	0.099 ± 0.079
	min - max	0.02 - 1.10	0.001 - 0.589	0.003 - 0.306	0.002 - 0.312	0.001 - 0.282
$\Delta\|y\|$	avr ± std	0.247 ± 0.307	0.111 ± 0.062	0.083 ± 0.057	0.109 ± 0.056	0.160 ± 0.086
	min - max	0 - 1.18	0.002 - 0.238	0.001 - 0.226	0.002 - 0.228	0.014 - 0.324
$\Delta\|z\|$	avr ± std	0.142 ± 0.113	0.129 ± 0.095	0.122 ± 0.084	0.086 ± 0.074	0.140 ± 0.090
	min - max	0 - 0.650	0.006- 0.347	0.002 - 0.292	0.001 - 0.264	0.000 - 0.456
$\Delta\|d\|$	avr ± std	0.421 ± 0.324	0.368 ± 0.145	0.194 ± 0.090	0.207 ± 0.100	0.256 ± 0.107
	min - max	0.078 - 1.563	0.077 - 0.647	0.036 - 0.395	0.042 - 0.459	0.062 - 0.468
$\Delta\|\alpha\|$	avr ± std	0.048 ± 0.053	0.029 ± 0.021	0.020 ± 0.015	0.018 ± 0.015	0.034 ± 0.023
	min - max	0.002 - 0.190	0 - 0.08	0 - 0.054	0 - 0.055	0.002 - 0.087
$\Delta\|\beta\|$	avr ± std	0.038 ± 0.041	0.041 ± 0.023	0.029 ± 0.018	0.017 ± 0.015	0.049 ± 0.038
	min - max	0 - 0.240	0.001 – 0.085	0.001 - 0.069	0 - 0.073	0.003 - 0.192
$\Delta\|\gamma\|$	avr ± std	0.053 ± 0.031	0.028 ± 0.026	0.034 ± 0.020	0.033 ± 0.023	0.040 ± 0.025
	min - max	0.005 – 0.163	0.001 – 0.116	0 - 0.086	0.002 - 0.085	0.002 - 0.088

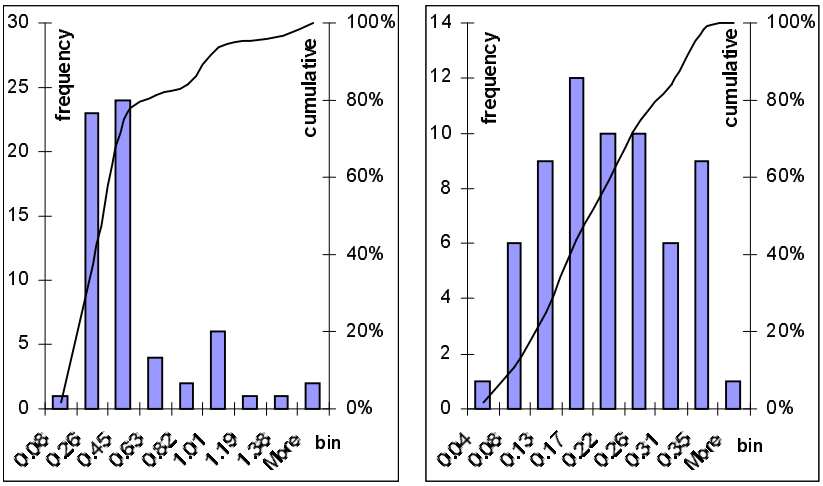
As seen from the table, polynomials of different order perform differently on different variables. For example, the smallest average of the residual for y was obtained while fitting it by the 3rd order polynomial whereas the 4th order polynomial performed better for z. Overall, the 3rd order

polynomials performed well for all variables.

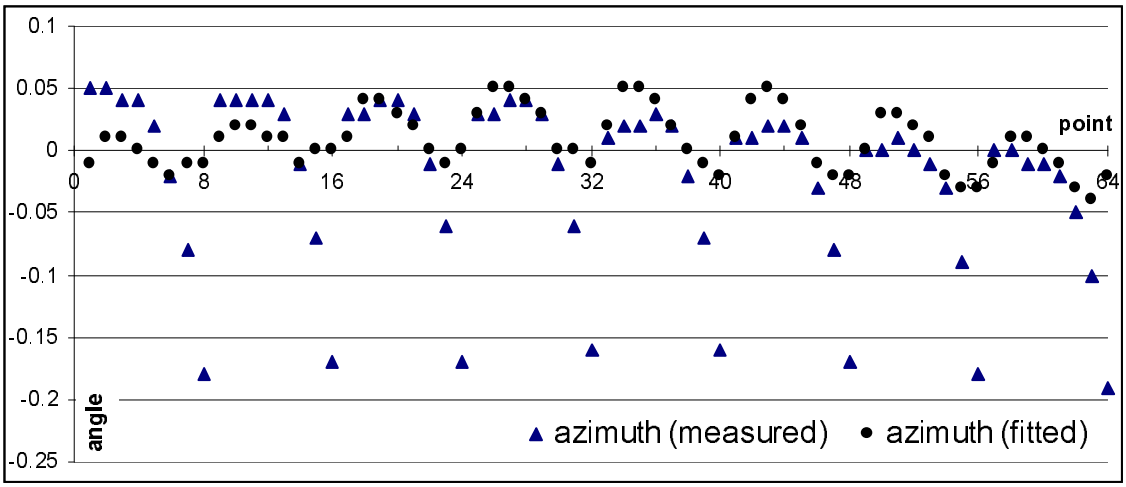
The graphs below contain measured (left plot) and fitted by the 3rd order polynomial (right plot) values of x coordinate of the points from the validation table plotted against its true value. As seen from the plots, fitted values are much closer to the true values (dotted line) comparing to the non-fitted values.



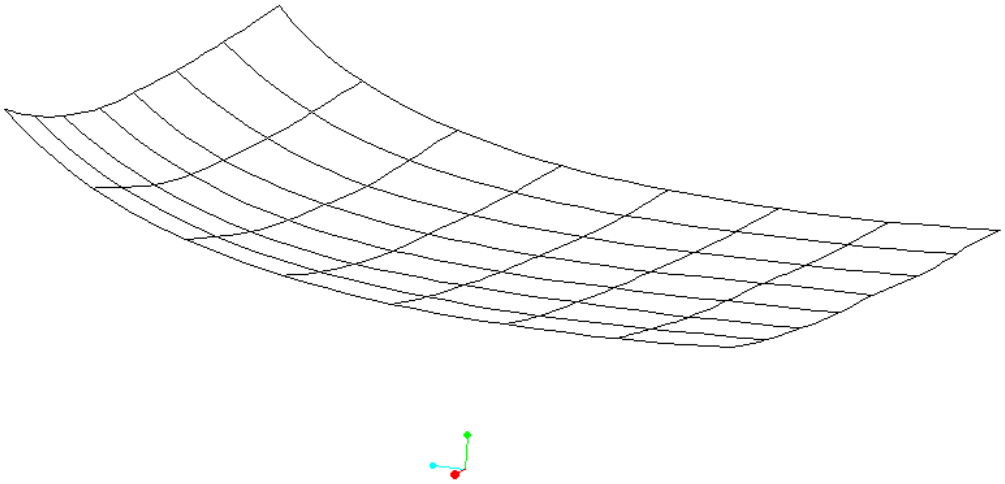
The following two graphs present frequency histograms obtained for the distances between the points defined by the true values of x, y and z and those reported by the tracker (left graph) and between the points defined by the true values of x, y and z and fitted by the 3rd order polynomial (right graph). As seen from the graphs, only about 38% of the non-fitted points are within 0.26 feet from the true points. At the same time, about 75% of the fitted points are within the same distance of 0.26 feet from the true points.



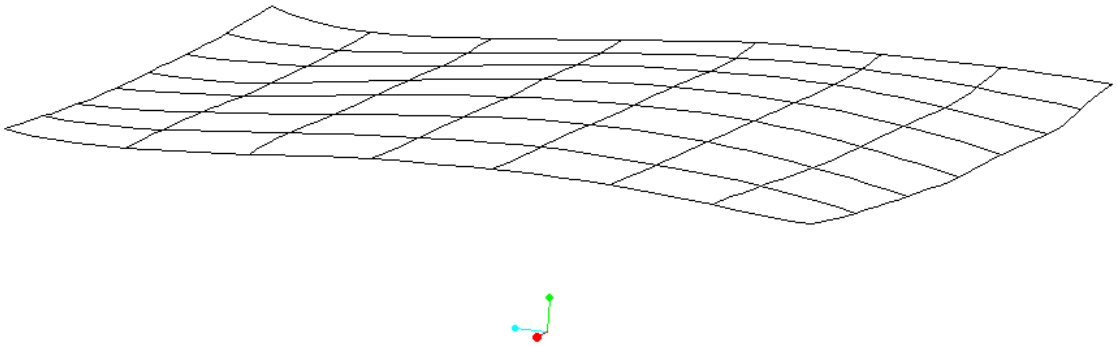
The next graph shows improvements achieved while correcting azimuth with the 3rd order polynomial. Measured and fitted values of azimuth of all 64 points from the validation table are plotted. True value of azimuth for all the points is zero. In a few cases fitted values are worth than the measured values, but overall they are much closer to the true value, especially for the points 2.5-3.5 feet back from the center of the CAVE.



The following image shows a surface build over the points from the validation table. Right edge of the surface is close to the front wall of the CAVE and corresponds to the measurements taken ~3.5 feet from the center of the CAVE, left edge corresponds to the measurements taken ~3.5 feet back from the center of the CAVE. As seen from the image, there is almost no distortions of the position measurements near the Flock of Birds' transmitter (right part of the surface is "almost flat") and there are significant distortions observed for the measurements taken at 1.5, 2.5, and 3.5 feet back from the center of the CAVE.



Finally, the following image shows a surface plotted over the points from the validation table after fitting them by the 3rd order polynomial. The surface is much "flatter" comparing to the previous picture, large distortions near the back of the CAVE are removed. However, the central part of the surface exhibits some distortions that contribute to the lover quality of the fit for the measurements taken in that area.



Implementation

The above described technique was implemented as a library[6] written in C and compiled on an SGI workstation. The following files are needed in order to use the library: calibfile.h (calibration file definitions), polyfit.h (calibration procedure definitions), libTrCalibr.a (the library). The following API is implemented:

int readcalibfile(**char** **fname*, **struct** *calibdata* **cdata*, **int** *rotunits*)

The function reads data from the calibration file *fname* and stores it into *cdata* structure. Calibration file has the following structure:

line 1: number of records
line 2: record 1
line 3: record 2
.....

where each record contains true x, y, z, measured x, y, z, and measured azimuth, elevation, and roll. Angles can be measured in degrees (*rotunits* = DEGREES) or radians (*rotunits* = RADIANS). It does not matter what units are used for x, y, and z as long as they are the same everywhere. The function returns the actual number of records if data was read, otherwise it returns 0. *cdata* structure must exist before calling the function, but the memory for its internal fields needed to store the data from the calibration file will be allocated by the function.

[6] http://www.ncsa.uiuc.edu/VEG/VPS/emtc/polynomial_fit.html

void deletecalibdata(**struct** *calibdata *cdata*)

This function removes the memory allocated by readcalibfile in *cdata*. It should be called after the fitting polynomials has been built.

int fit_initialize(**struct** *calibdata *cdata*, **int** *porder*, **struct** *fpolynomials *fpol*)

This function takes the calibration table loaded by the readcalibfile in *cdata* and computes the fitting polynomials of the order *porder* both for position and orientation components; results are returned in *fpol* structure. *fpol* structure must exist before calling the function, but the memory for its internal fields needed to store the polynomials will be allocated by the function.

void fit_evaluate(**struct** *fpolynomials *fpol*, **float** *xm*, **float** *ym*, **float** *zm*, **float** *azm*, **float** *elm*, **float** *rlm*, **float** **xf*, **float** **yf*, **float** **zf*, **float** **azf*, **float** **elf*, **float** **rlf*)

Once the polynomials *fpol* have been computed by fit_initialize, this function can be called to fit measured values *xm*, *ym*, *zm*, *azm*, *elm*, and *rlm* (angular values are in radians); results are returned in *xf*, *yf*, *zf*, *azf*, *elf*, and *rlf*. Returned angular values are corrected after the fitting.

void fit_statistics(**struct** *calibdata *cdata*, **struct** *fpolynomials *fpol*, **char** **fname*)

This function takes *cdata* loaded by readcalibfile, fits it by the polynomials *fpol* computed by fit_initialize, computes statistics and stores the results into a file *fname*.fit. This is a useful tool that allows to study quality of the calibration.

An example of the use of the library is given below.

```
#include <stdio.h>
#include "calibfile.h"
#include "polyfit.h"

int main (int argc, char **argv)
{
    struct calibdata cdata;
    struct fpolynomials fpol;
    float xm, ym, zm, azm, elm, rlm;
    float xf, yf, zf, azf, elf, rlf;

    if (argc != 3)
    {
        printf("Usage: %s callibration.file polyorder\n", argv[0]);
        printf("       polyorder - order of the polynomial to be used\n");
        exit(1);
    }

    porder = atoi(argv[2]); /* order of the fitting polynomial */

    /* read calibration file */
    if (!readcalibfile(argv[1], &cdata, DEGREES))
    {
        printf("Unable to read data file\n");
        exit(1);
    }
    else
        printf("file %s containing %i records is loaded\n", argv[1], cdata.M);
```

```
    /* build fitting polynomials */
    if (!fit_initialize(&cdata, 3, &fpol))
    {
        printf("Unable to build polynomials\n");
        exit(1);
    }
    else
        printf("fitting polynomials (%i terms) are built\n", fpol.N);

    /* free memory that was allocated for the calibratin file */
    deletecalibdata(&cdata);

    /* run the fitting loop */
    while (1)
    {
        /* now the angles are in radians */
        printf("enter measured values of x, y, z, az, el, and rl:\n");
        scanf("%f %f %f %f %f %f", &xm, &ym, &zm, &azm, &elm, &rlm);

        fit_evaluate(&fpol, xm, ym, zm, azm, elm, rlm, &xf, &yf, &zf, &azf, &elf, &rlf);

        printf("corrected values are: %2.3f %2.3f %2.3f %2.3f %2.3f %2.3f\n",
               xf, yf, zf, azf, elf, rlf);
    }
}
```

Future work

There are several issues that need to be addressed. First, a detailed study is required to identify a subspace near the tracking system that really needs to be calibrated. In the present study, almost entire CAVE space was included; however, in reality there are places that are never reached by the users. Excluding them from the calibration may lead to a better calibration quality of the remaining sub-space. Currently, a study is going on to identify such places. Second, it needs to be studied how many measurements are enough to reach a specific level of the calibration quality. Finally, the use of other basis functions for the fitting polynomials needs to be explored. In the present study only the functions in the form $x^{p'_j} y^{p''_j} z^{p'''_j}$ were used, however, they may not be the best suited ones taking into account the nature and the shape of the magnetic field distortions.

Acknowledgments

Several people helped to conduct the study. Henry Kaczmarski built a support device that was used to hold the tracker's receiver at known locations. Lance Arsenault advised with physics/math issues.

Appendix A

Calibration table obtained for the Flock of Birds (Ascension Technology Corporation) tracking system installed in the CAVE at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

i	$\bar{x}_i$	$\bar{y}_i$	$\bar{z}_i$	x_i	y_i	z_i	α_i	β_i	γ_i
1	−4.00	2.03	−4.00	−1.97	3.50	−5.05	29.6	−20.9	10.7
2	−4.00	2.03	−2.00	−2.61	3.11	−3.01	18.1	−10.5	7.2
3	−4.00	2.03	0.00	−2.67	3.00	−1.59	20.7	−10.1	6.2
4	−4.00	2.03	2.00	−2.38	2.83	−0.80	33.4	−12.2	14.3
5	−2.00	2.03	−4.00	−1.47	3.24	−4.43	17.4	−7.8	−1.1
6	−2.00	2.03	−2.00	−1.68	2.89	−2.52	11.5	−5.5	−2.1
7	−2.00	2.03	0.00	−1.79	2.93	−0.87	11.1	−7.5	−4.4
8	−2.00	2.03	2.00	−1.86	3.11	0.33	14.7	−11.1	−7.0
9	0.00	2.03	−4.00	0.03	3.07	−4.33	15.7	0.6	−2.1
10	0.00	2.03	−2.00	−0.10	2.70	−2.42	10.3	−1.4	−5.8
11	0.00	2.03	0.00	−0.30	2.71	−0.67	9.4	−5.0	−9.1
12	0.00	2.03	2.00	−0.60	2.95	0.72	11.0	−10.7	−12.3
13	2.00	2.03	−4.00	1.45	3.20	−4.40	16.2	7.9	−5.1
14	2.00	2.03	−2.00	1.40	2.69	−2.48	10.4	2.8	−10.7
15	2.00	2.03	0.00	1.12	2.58	−0.71	10.3	−4.1	−15.7
16	2.00	2.03	2.00	0.57	2.73	0.73	11.6	−12.9	−21.5
17	4.00	2.03	−4.00	1.91	3.21	−4.77	22.9	18.4	−20.4
18	4.00	2.03	−2.00	2.30	2.64	−2.66	11.3	5.5	−23.8
19	4.00	2.03	0.00	1.98	2.35	−0.96	12.4	−5.0	−31.3
20	4.00	2.03	2.00	1.19	2.22	0.48	15.0	−18.1	−39.0
21	−4.00	4.03	−4.00	−3.27	4.38	−4.13	7.5	−8.1	2.1
22	−4.00	4.03	−2.00	−3.46	4.23	−2.20	5.4	−3.4	1.5
23	−4.00	4.03	0.00	−3.47	4.31	−0.37	4.4	−2.6	0.1
24	−4.00	4.03	2.00	−3.46	4.69	1.26	2.0	−3.6	−2.5
25	−4.00	4.03	4.00	−3.67	5.64	2.45	−6.6	−8.4	−9.7
26	−2.00	4.03	−4.00	−1.75	4.15	−3.90	4.4	−1.0	−0.2
27	−2.00	4.03	−2.00	−1.77	4.06	−1.97	3.8	−0.1	0.2
28	−2.00	4.03	0.00	−1.78	4.14	−0.10	3.4	0.0	−0.6
29	−2.00	4.03	2.00	−1.80	4.47	1.63	0.8	−1.1	−2.9
30	−2.00	4.03	4.00	−1.99	5.33	3.11	−8.3	−4.9	−7.2
31	0.00	4.03	−4.00	0.12	4.08	−3.83	3.5	2.3	−0.9
32	0.00	4.03	−2.00	0.13	4.01	−1.91	3.0	1.6	−1.7
33	0.00	4.03	0.00	0.08	4.08	−0.01	2.9	0.5	−2.1
34	0.00	4.03	2.00	−0.05	4.39	1.77	0.3	−1.4	−2.9
35	0.00	4.03	4.00	−0.36	5.18	3.34	−8.4	−6.2	−5.1
36	2.00	4.03	−4.00	1.95	4.14	−3.82	2.9	4.1	−2.1
37	2.00	4.03	−2.00	1.95	4.05	−1.90	2.2	2.8	−3.8
38	2.00	4.03	0.00	1.86	4.12	0.00	1.9	−0.2	−4.5
39	2.00	4.03	2.00	1.61	4.43	1.81	−0.5	−3.3	−5.0
40	2.00	4.03	4.00	1.01	5.21	3.43	−8.6	−11.5	−4.9
41	4.00	4.03	−4.00	3.45	4.30	−3.90	3.4	8.7	−5.4
42	4.00	4.03	−2.00	3.55	4.16	−1.93	1.3	3.2	−7.2
43	4.00	4.03	0.00	3.41	4.22	0.01	1.0	−1.5	−8.7
44	4.00	4.03	2.00	2.96	4.51	1.82	−1.1	−7.9	−9.4
45	4.00	4.03	4.00	1.85	5.25	3.59	−9.4	−22.7	−9.6
46	−4.00	6.03	−4.00	−3.57	6.04	−4.04	1.9	−4.1	−1.5
47	−4.00	6.03	−2.00	−3.63	5.95	−2.05	1.5	−2.0	0.0
48	−4.00	6.03	0.00	−3.58	6.06	−0.09	0.3	−0.6	−1.8
49	−4.00	6.03	2.00	−3.48	6.56	1.76	−5.6	1.6	−5.8
50	−4.00	6.03	4.00	−3.35	7.83	3.25	−20.8	2.1	−14.9
51	−2.00	6.03	−4.00	−1.78	5.87	−3.90	0.5	−0.5	−0.2
52	−2.00	6.03	−2.00	−1.78	5.84	−1.92	1.7	0.3	0.8
53	−2.00	6.03	0.00	−1.77	5.89	0.02	1.9	0.7	−0.2
54	−2.00	6.03	2.00	−1.69	6.21	1.91	−2.9	1.5	−2.7
55	−2.00	6.03	4.00	−1.73	7.28	3.57	−17.4	0.9	−7.3
56	0.00	6.03	−4.00	0.13	5.87	−3.90	−0.4	1.1	−0.6
57	0.00	6.03	−2.00	0.17	5.87	−1.92	0.9	1.2	−0.9
58	0.00	6.03	0.00	0.17	5.87	0.05	2.0	1.1	−0.9
59	0.00	6.03	2.00	0.11	6.15	1.95	−3.0	−0.1	−1.0
60	0.00	6.03	4.00	−0.07	7.09	3.68	−16.2	−3.1	−1.3
61	2.00	6.03	−4.00	2.08	5.89	−3.86	−0.3	2.5	−2.0

62	2.00	6.03	-2.00	2.08	5.87	-1.88	0.4	1.2	-3.4
63	2.00	6.03	0.00	2.05	5.94	0.09	0.5	-0.3	-2.2
64	2.00	6.03	2.00	1.89	6.25	1.98	-3.8	-2.6	-0.6
65	2.00	6.03	4.00	1.46	7.24	3.73	-16.0	-9.3	3.6
66	4.00	6.03	-4.00	3.85	6.05	-3.86	-0.1	5.0	-1.6
67	4.00	6.03	-2.00	3.87	5.98	-1.84	-0.8	0.4	-3.8
68	4.00	6.03	0.00	3.76	6.14	0.19	-1.9	-3.6	-2.8
69	4.00	6.03	2.00	3.38	6.58	2.07	-6.7	-9.4	0.3
70	4.00	6.03	4.00	2.55	7.73	3.85	-18.0	-21.8	8.8
71	-4.00	8.03	-4.00	-3.70	8.08	-4.00	-1.2	-3.9	-7.1
72	-4.00	8.03	-2.00	-3.68	7.95	-2.03	-2.5	-2.1	-2.6
73	-4.00	8.03	0.00	-3.57	8.11	-0.05	-4.1	2.5	-5.5
74	-2.00	8.03	-4.00	-1.90	7.85	-3.87	-1.6	-0.2	-0.4
75	-2.00	8.03	-2.00	-1.90	7.83	-1.92	-1.0	-0.1	1.2
76	-2.00	8.03	0.00	-1.88	7.89	0.05	-1.4	0.8	-1.9
77	-2.00	8.03	2.00	-1.65	8.14	1.93	-7.6	4.6	-4.9
78	0.00	8.03	-4.00	0.09	7.83	-3.85	-4.8	-0.2	-0.8
79	0.00	8.03	-2.00	0.03	7.83	-1.87	-1.1	-4.6	-7.7
80	0.00	8.03	0.00	0.11	7.90	0.07	-2.1	0.6	-0.9
81	0.00	8.03	2.00	0.11	8.12	1.96	-8.8	1.2	-0.7
82	2.00	8.03	-4.00	2.06	7.89	-3.87	-2.0	0.5	-1.1
83	2.00	8.03	-2.00	2.04	7.88	-1.88	-1.5	-1.4	-2.8
84	2.00	8.03	0.00	2.03	7.96	0.10	-2.7	-2.4	1.1
85	2.00	8.03	2.00	1.87	8.18	1.95	-7.9	-4.0	3.5
86	4.00	8.03	-4.00	3.92	8.10	-3.82	-2.8	2.6	2.4
87	4.00	8.03	-2.00	3.84	7.99	-1.77	-2.5	-3.1	-1.9
88	4.00	8.03	0.00	3.66	8.19	0.30	-3.0	-10.0	2.3

Appendix B

Validation table obtained for the Flock of Birds (Ascension Technology Corporation) tracking system installed in the CAVE at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

i	$\bar{x}_i$	$\bar{y}_i$	$\bar{z}_i$	x_i	y_i	z_i	α_i	β_i	γ_i
1	-3.49	5	-3.5	-3.06	5.01	-3.42	2.77	-2.02	-0.7
2	-3.49	5	-2.5	-3.09	4.97	-2.46	2.59	-1.23	-0.31
3	-3.49	5	-1.5	-3.1	4.97	-1.5	2.46	-0.77	-0.44
4	-3.49	5	-0.5	-3.1	5.02	-0.55	2.13	0.09	-1.03
5	-3.49	5	0.5	-3.09	5.12	0.38	1.23	0	-2.02
6	-3.49	5	1.5	-3.05	5.32	1.27	-0.86	0.66	-3.36
7	-3.49	5	2.5	-3.01	5.64	2.1	-4.37	-0.11	-5.47
8	-3.49	5	3.5	-3.03	6.18	2.85	-10.59	-0.7	-9.34
9	-2.49	5	-3.5	-2.21	4.94	-3.34	2.26	0.18	-0.48
10	-2.49	5	-2.5	-2.2	4.91	-2.4	2.11	0.82	-0.26
11	-2.49	5	-1.5	-2.2	4.91	-1.44	2.18	1.39	-0.48
12	-2.49	5	-0.5	-2.2	4.94	-0.49	2.13	1.61	-0.83
13	-2.49	5	0.5	-2.18	5.03	0.47	1.65	1.19	-1.69
14	-2.49	5	1.5	-2.15	5.2	1.37	-0.4	1.61	-2.97
15	-2.49	5	2.5	-2.15	5.5	2.23	-3.78	1.47	-4.72
16	-2.49	5	3.5	-2.19	5.99	3.03	-9.71	0.18	-7.38
17	-1.49	5	-3.5	-1.25	4.9	-3.34	1.52	1.74	-1.25
18	-1.49	5	-2.5	-1.26	4.88	-2.37	1.76	2.16	-0.88
19	-1.49	5	-1.5	-1.26	4.88	-1.41	2.09	1.91	-0.88
20	-1.49	5	-0.5	-1.25	4.9	-0.45	2.11	1.52	-1.23
21	-1.49	5	0.5	-1.25	4.98	0.52	1.6	2.02	-1.87
22	-1.49	5	1.5	-1.26	5.12	1.42	-0.29	2.11	-2.59
23	-1.49	5	2.5	-1.28	5.41	2.31	-3.71	1.67	-4

24	−1.49	5	3.5	−1.34	5.87	3.14	−9.47	0.07	−5.67
25	−0.49	5	−3.5	−0.27	4.89	−3.28	1.63	2.27	−1.82
26	−0.49	5	−2.5	−0.27	4.88	−2.34	1.65	2.4	−1.82
27	−0.49	5	−1.5	−0.27	4.87	−1.37	2.09	2.2	−1.91
28	−0.49	5	−0.5	−0.28	4.89	−0.4	2.26	1.83	−2.04
29	−0.49	5	0.5	−0.3	4.96	0.56	1.71	0.84	−1.91
30	−0.49	5	1.5	−0.33	5.1	1.47	−0.35	0.22	−2.4
31	−0.49	5	2.5	−0.39	5.36	2.38	−3.52	0.13	−3.08
32	−0.49	5	3.5	−0.47	5.81	3.21	−9.38	−2.37	−3.87
33	0.51	5	−3.5	0.68	4.89	−3.32	0.64	2.62	−2.53
34	0.51	5	−2.5	0.69	4.88	−2.35	0.92	2.22	−2.79
35	0.51	5	−1.5	0.69	4.88	−1.38	1.36	1.61	−2.81
36	0.51	5	−0.5	0.67	4.9	−0.4	1.49	1.3	−2.68
37	0.51	5	0.5	0.64	4.97	0.54	0.88	1.26	−2.7
38	0.51	5	1.5	0.59	5.11	1.47	−0.88	0.05	−2.83
39	0.51	5	2.5	0.47	5.37	2.37	−3.93	−1.27	−2.83
40	0.51	5	3.5	0.32	5.8	3.23	−9.43	−3.73	−2.94
41	1.51	5	−3.5	1.62	4.91	−3.31	0.46	3.14	−3.19
42	1.51	5	−2.5	1.61	4.89	−2.34	0.51	2.16	−3.54
43	1.51	5	−1.5	1.6	4.9	−1.37	0.88	1.85	−3.52
44	1.51	5	−0.5	1.59	4.93	−0.39	0.92	1.01	−3.49
45	1.51	5	0.5	1.56	5	0.56	0.44	0.4	−3.47
46	1.51	5	1.5	1.49	5.15	1.49	−1.49	−0.79	−3.45
47	1.51	5	2.5	1.35	5.42	2.39	−4.5	−3.27	−3.01
48	1.51	5	3.5	1.14	5.86	3.27	−9.69	−5.84	−2.22
49	2.51	5	−3.5	2.54	4.95	−3.31	0.22	4.71	−3.8
50	2.51	5	−2.5	2.53	4.93	−2.33	0.22	3.3	−4.5
51	2.51	5	−1.5	2.53	4.93	−1.36	0.35	1.83	−4.83
52	2.51	5	−0.5	2.48	4.97	−0.37	0.24	0.88	−4.64
53	2.51	5	0.5	2.41	5.06	0.58	−0.33	−0.99	−4.5
54	2.51	5	1.5	2.29	5.23	1.52	−2	−2.85	−3.54
55	2.51	5	2.5	2.1	5.51	2.42	−5.12	−5.16	−3.05
56	2.51	5	3.5	1.81	5.96	3.29	−10.22	−9.34	−1.45
57	3.51	5	−3.5	3.38	5.02	−3.31	0.2	5.08	−4.44
58	3.51	5	−2.5	3.4	4.98	−2.32	−0.11	3.28	−5.47
59	3.51	5	−1.5	3.38	4.99	−1.36	−0.48	1.17	−5.71
60	3.51	5	−0.5	3.33	5.04	−0.35	−0.64	−0.33	−5.71
61	3.51	5	0.5	3.22	5.15	0.62	−1.32	−2.59	−5.34
62	3.51	5	1.5	3.05	5.32	1.53	−3.1	−4.81	−4.81
63	3.51	5	2.5	2.81	5.61	2.43	−5.95	−7.93	−3.76
64	3.51	5	3.5	2.41	6.1	3.35	−10.9	−13.73	−1.67