Instructions to setting up the

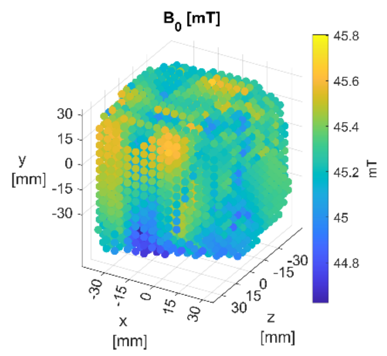
Mapping of B-Field

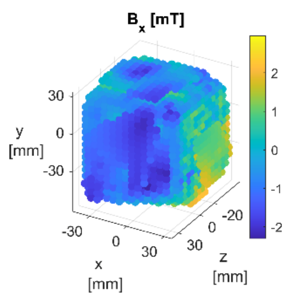
Project

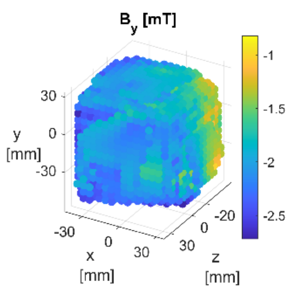
Easy Scalable, Low-Cost Open Source Magnetic Field Detection System for Evaluating Low-Field MRI Magnets using a Motion Tracked Robot

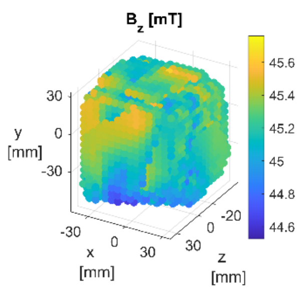
Pavel Povolni

October 2024









Content

1 Preamble

These instructions describe the Mapping of the magnetic field.
Further information can be found in the publication, as well as in the Appendix and Supplementary Information referenced there.
Software used:

1. Mapping: Matlab 2021b (w/o expensive toolboxes)

If you have any further questions, please contact us!

2 Measurement

You can find all scripts in the folder:

...\5_Mapping

2.1 Preparation

Before the measurement can start, a few preparation steps are necessary. An overview is given below.

2.1.1 Update SetUp File

All parameters that may need to be adapted to your setup and your robot are saved and commented in the file "\ Constants_Mechanical_Robot\get_SetFile.m".
Originally, this was supposed to be a JSON file that can also be read by the Python scripts (so that some of the values don't have to be copied/pasted). Well... at some point we forgot about that :D
Therefore, it is still partially duplicated here and you have to make sure that the values are the same in both cases.

Depending on how much you have changed in your setup, you will have to adjust the parameters significantly. If you have set it up the same way, you should have to only make little or no changes.

The following is an overview of all parameters:

1. **Mechanical Constants**
   - Mechanics of the robot (length of the individual links and joints)
   - Definition of the internal distances on the Sensor Head (distance to each other). The point "P5" defines the front screw-on point (see Figure 1), center of the screw, underside of the lower PCB
   - Definition of the translation of the robot base and the magnet (center of the FOV) to the origin of the Reference Coordinate System (ORB)

2. **Kinematics**
    - Definition of the possible angles (in °) with the limits set in a way, that it is not possible to drive in the mechanical limit. Ss well as offset values (definition of the zero angle differs in Motion Tracking and in robot control)
    - Horizontality (tol_horizontality)
      Due to the kinematics, the sensor cannot be moved to every point while it is positioned perfectly horizontally. This parameter specifies how large the angular deviation from the horizon may be.
    - Definition of the mechanical robot tolerances:
      Due to tolerances in the assembly of the robot done by us (in particular the mechanical mounting of the motors), there are differences between the angles set in the control system and the angles actually approached. We have determined this tolerance as follows: The arm should be completely vertical upwards and the base should be oriented in the direction of the magnet. To do this, we send the robot controller the following angles in a serial command window: “+090, +090,+090,+090”.
      These angles are changed iteratively until the arm reaches the desired position. In our case, for example, the robot is aligned towards the magnet with a base angle of 85° (instead of the ideal 90°). The offset for the base is then calculated as: 90-85
3. Communication Setup
    - Specify the serial ports that you have assigned to the Hall sensor and the robot controller (resp. which are assigned by your operating system)
4. Setup for Camera Based Measurement
    - Specify the recording limit of your camera.
    - Duration of how long the robot wait at a position (here 16s) and how long the robot has time to move to this position (4s)

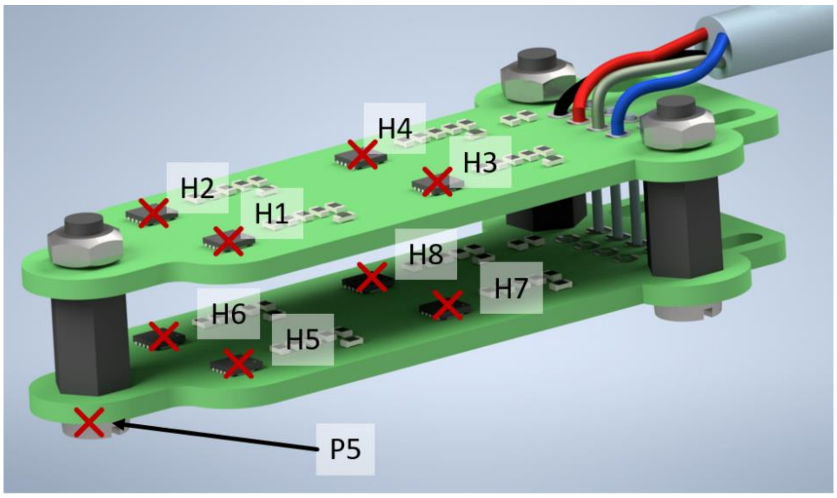


Figure 1: Position of Hall Sensors and Point "P5" on the Sensor Head

2.1.2 Generate Kinematics, FOV & Trajectory

Before the measurement, the points of the FOV to be approached and the resulting robot angles are calculated using inverse kinematics.
The inverse kinematics is underdefined (since we have 3 position coordinates, but are calculating 4 angles). The problem is optimized (using fmincon), considering the permitted horizontality.

Use this (commented) script for the calculation:
"\Generate_Kinematics_and_FOV\ Generate_Trajectory_FOV_Coordinates.m"

At the moment only cubic FOVs are calculated. If you want to calculate other FOVs (e.g. spiral, or cylindrical) you have to adapt the script (probably only the function Function_TrajectoryFOV_GenerateCube.m is enough, but check if the rest works).
The 2 main parameters that you should adjust are:

- **cube_StepSize**: Specification of the isotropic step size with which the FOV is scanned. For the maximum benefit of the 8Channel Hall sensors, common divisors/multiples of the Hall sensor distances should be used, which are predefined as 5,10,20,40mm. The script checks whether your selected steps size is part of this list and adjusts it if necessary.
- **cube_FOVSize**: Size of the FOV that is to be mapped. Specification of the x/y/z size as a vector. Uses a multiple of the StepSize so that the 8 Hall sensors distributed on the Sensor Head are used to maximum effect (and the measurement therefore has the maximum speed)

A step size of 5mm is ideal for precise Mapping. For gradient mapping and for tests, larger step sizes are useful (in our case 10mm). The FOV is such generated that the FOV center equals the center of the magnet (defined in the SetFile, see 2.1.1).

The calculated FOV is then used to calculate the joint angles to be set for the robot and to calculate the trajectory. The trajectory is simulated and displayed in a plot (good for debugging, see Figure 2). However, this simulation takes a certain amount of time. You can deactivate the plots with the "plot_flag" flag.

The points reached by the robot are not necessarily identical to the desired position. This is due to the limiting kinematics, the desired horizontality of the sensor and the control of the robot (only whole angles can be set).
The calculated angles are saved in the following file and imported again later in the Mapping:
\Save_Folder\FOV_Trajectory_alphas.mat

The trajectory (sequence of FOV points) is calculated in such a way that the robot moves as little as possible from point to point. We have found that this reduces the positioning error. Pay attention to this if you create your own FOV geometries.
The FOV is passed in yz-planes in positive x-direction.

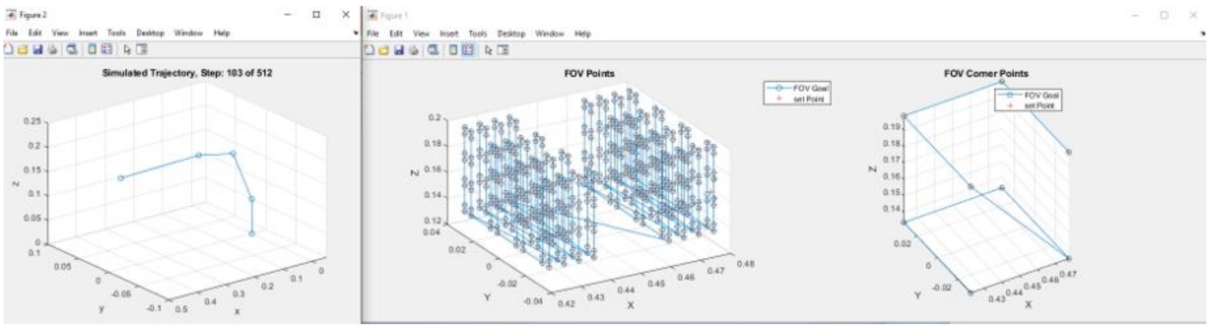


Figure 2: Plots of the FOV generation and calculation of the trajectory

2.1.3 In Case Gradient Mapping: Prepare Current Measurement

To be able to map gradients with this setup, strong fields are required (i.e. high currents and losses) to avoid resolution limit of the Hall sensors.

For current measurement, we have reused the setup from chapter 4 / 1_Hallsensor_manual. The manually switched relay there (and thus the flowing current) has been automated here.

An Arduino-Uno (script in folder "\Functions_Gradient\Code_CurrentSwitchOn") is connected serially (9600Bd) and switches on the current (by switching the relay) as required. In our case, the wiring between the Arduino and the relay was "flying" and not rigid (a wire was connected directly to the pin header & the corresponding pin). There is therefore no PCB for this. The overall structure is shown schematically in Figure 3.

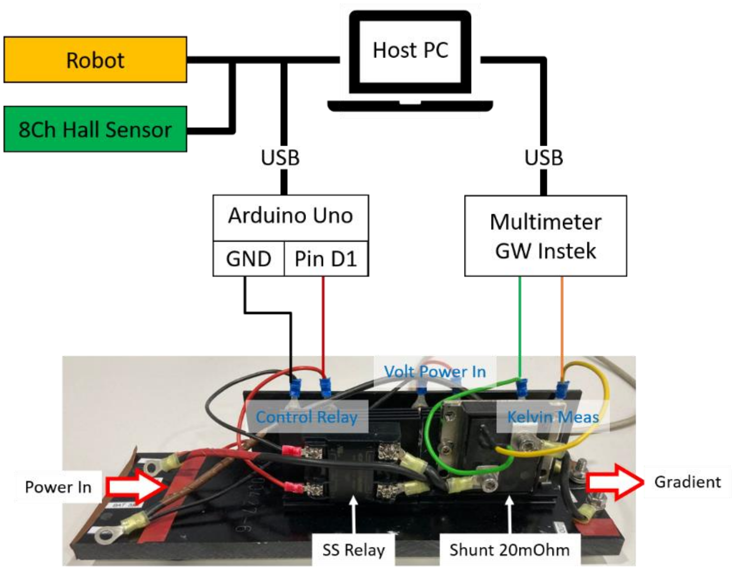


Figure 3: Schematic structure of the current control for gradient mapping

2.1.4 Move Sensor Head inside the FOV

“Chapter4/2_Robot_manual” has already described that the Hall sensor needs to be moved manually to the center of the magnet at the start of the Mapping. The reason for this is the complex trajectory required to move the Sensor Head into the magnet (due to the long arm on which the Hall sensor is mounted).
Pay particular attention to this, as a crash of the robot with the FOV can damage something (your DUT, the sensor or the arm itself). Unfortunately, this happened to us due to inattention. The servo motors generate an enormous amount of torque and broke the arm through (mounting of the sensor) during the crash.

2.2 Run Magnet Mapping

Now it’s time – the final Mapping can start!
After everything is set up and all the software is prepared… yes, that was a really long way - hang in there you're almost done!...

You start the Mapping of the magnet with the script "Main_Measurement_Halbach_Magnet.m". We have measured a Halbach, hence the script name. But of course, any other magnet can be measured too.
It takes a few seconds to establish serial communication with the Hall sensor and the robot when the script is started.
You can use the "flag_check_corner" flag to perform a collision check before the measurement. The robot moves into the corners of the generated cubic FOV and allows you to check whether the sensor will collide with something.
This could happen, for example, if the FOV has been selected too large or the desired position cannot be reached due to the kinematics.

As soon as everything is ready, the script informs about it in the command window, that the user can start the measurement.
Now start the camera recording and start the Mapping in the Command Window with "Enter".

- *Camera Control*

As soon as the camera's recording limit is reached, the measurement is interrupted. The computer beeps and reports in the command window that the camera limit has been reached. Restart the camera (see Chapter 4/3_Motion_Tracking_manual) and start further Mapping with "Enter" in the command window.

- *Retension Shoulder Relief Spring*

The FOV is passed yz-plane by yz-plane in the positive x-direction. After half the measurement, half the FOV has been traveled through. The x-position of the sensor is approximately at the center of the FOV.

In our setup, it was now necessary to retension the shoulder relief so that all points can be reached in the 2nd half (especially the upper FOV half with y>FOV center). The script stops for this (comment block 7.1, line 78) and informs the user.
Install the new springs or tighten the rubber band. Restart your camera recording. Press "Enter" in the command window to start the rest of the Mapping.

In a subsequent development, it would make more sense to integrate automatic retensioning (or to install a spring that has a longer linear range)

- *Data storage*

All data is sorted in the struct Save_Data. After each position, this variable is saved temporarily to protect against data loss (name "Measurement_ Save_Data").

At the end of the Mapping, the variable is saved under the name: "Measurement_ *datum*", whereby the current date is inserted in the name

2.3 Run Gradient Mapping

The script "Main_Measurement_Gradient.m" is used to map a gradient where the power supply must be switched. If you have a gradient that is cooled well enough (protection against overtemperature, constant current flow), the usual Mapping (2.2) can be used. In our case, this was not possible, so the script was extended to include switching on the gradient and a current measurement.

Essentially, there is no change to the magnet Mapping. The additional devices (multimeter and Arduino for relay control) are defined directly in the "Main_Measurement_Gradient.m" script. In our case, the multimeter could also be addressed serially. This changes depending on which multimeter you use:
All communication with the multimeter is bundled in the "Function_Gradient_GetCurrent" function. Change this function so that it is suitable for your multimeter. Alternatively, you can set up your own voltage measurement (see Chapter 4.2.3/1_Hallsensor_manual). No changes to the Matlab code should then be necessary.

Before the measurement starts, change the following parameters to your setup in this script:

- Relay_COM_Port = "COM9";
- Relay_SRL_Baud = 9600;
- Multi_COM_Port = "COM10";
- Multi_SRL_Baud = 115200;
- Multi_R_Shunt = 0.02;

The gradient is switched on for 16 seconds at each measuring point. The ohmic losses increase the temperature of the gradient and the ohmic resistance rises, which changes the current flow (depending on the current source). In our setup, a laboratory power supply was used in CC mode with sufficient voltage headroom. Therefore, the current flow was stable during the 16 seconds. To increase the accuracy, 2 current measurements (before and after the magnetic field measurement) are carried out and averaged in the evaluation at the end.

You start the Mapping by starting the script. There is no difference to the magnet Mapping in Chapter 2.2. Only a cooling phase, in which the gradient is cooled by convection, has been integrated:

- *Cooldown phase*

We have included a cool-down phase every 6 measurements (2 minutes). In Line 110, the current iteration is checked and the script is stopped. The user is informed via beeps and in the command window.
We had set up a fan that constantly cooled the gradient (Figure 4). The cooling phase was about 2-5 minutes. However, we could probably have carried out the entire measurement in one run with our setup, the cooling was good enough and the power supply had enough headroom.

The camera recording was stopped during this pause and only restarted shortly before the measurement was continued. Halfway through the measurement, the shoulder relief was retensioned.

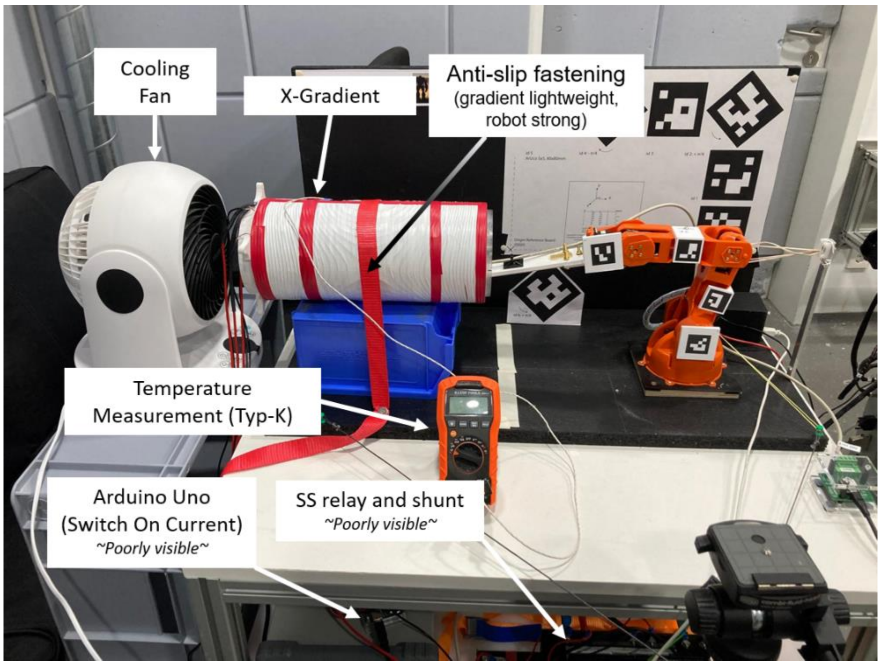

Cooling Fan
X-Gradient
Anti-slip fastening (gradient lightweight, robot strong)
Temperature Measurement (Typ-K)
Arduino Uno (Switch On Current) ~Poorly visible~
SS relay and shunt ~Poorly visible~

Figure 4: Image of the gradient measurement

3 Analysis in Post Processing

The measurements are complete and it's time for the evaluation. First of all, congratulations, you've come a long way and learned a lot about hardware and optical Motion Tracking!

This involves evaluating the measurement and linking the individual measurement data so that a data set is created at the end that can be used in further development steps (e.g. for calculating a shimming field).

3.1 Preparation

3.1.1 Data Handling

For the analysis you need the following 4 data collections, which are stored in the folder:

...\5_Mapping\Analysis_Mapping\Data_for_Evaluation

- Measurement data of the magnetic field from chapter 2 (this manual), Name: "*Measurement_datum*.mat"
- Motion Tracking evaluation from chapter 4.2 (3_Motion_Tracking_manual), Name: "MotionTracking_Log.csv"
- calculated (ideal) FOV from chapter 2.1.2 (this manual), Name: "FOV_Trajectory_alphas.mat"
- Calculated calibration of the Hall sensor from chapter 4.4 (1_Hallsensor_manual), Name: "*Calibration_50mT_datum*.mat"

Make sure that data from different measurements are not mixed together.

3.1.2 SetUp software (update parameters)

The script that is used for the evaluation is this one:

“...\5_Mapping\Analysis_Mapping\Main_PostProcessing_inclSensor.m”

In this evaluation, the "SetFile" (“get_SetFile”) is used again, which was already used in Chapter 2.1.1 has already been adapted to your system.
The entire script is heavily commented in case you want to customize anything. This script should actually work with any robot and system as long as the data looks the same. All relevant setup parameters are stored centrally in the setup file.

At this point, only a few parameters/flags remain that you can adapt to your system. These are summarized in the "`%% 1) User Settings`" block:

Flags

- show_plot: Flag indicating whether a plot of each tracked point (point P5 of the Sensor Head) should be shown. This plot can take quite a long time, so only every 10th point is shown (can be changed in Line 250)
- show_plot_HallSens: Flag indicating whether the trajectory of the Hall sensor should be shown. The trajectory of the Sensor Head is simulated. This plot is very helpful for debugging, but takes an extremely long time. Switch the flag off when you know that the analysis is working.
- use_CalFile: Flag indicating whether the calibration of the Hall sensor should be used. Can be helpful to qualify the effect of the calibration

Parameters

- active_Hallsens: Specification of how many of the installed Hall sensors are to be used (only relevant in the event that you only use 4 sensors, for example - i.e. only use one level)
- Tref: Room temperature in °C
- step_goal: Specifies the step size to which the measured magnetic field is interpolated. Due to the kinematics of the robot and the tolerance, the measured points deviate from the perfect FOV grid. The interpolation was helpful to simplify subsequent evaluations (e.g. shimming calculations).

Storage locations and file name

- file_HallData: Dataset of the measured magnetic field
- file_MT: Dataset of the evaluated Motion Tracking
- file_FOV: specified FOV points, which were measured
- file_CalHallData: Calibration data of the Hall sensors. It may be worth repeating the calibration of the Hall sensor over time. A more recent file can therefore also be used here.

Nice-To-Have parameters

The magnet is indicated in the plots. This helped us with debugging, as it was possible to see directly whether the tracked points are actually in the magnet itself (and for example not swapped axes in the Motion Tracking axes).
To do this, the inner cylinder of the magnet is drawn semi-transparently in the plot. In the local function "genFOVCylinder" (in the script "Main_PostProcessing_inclSensor.m", Line 510) there are 2 parameters for this:

- r: Radius of the cylinder
- h: Length (height) of the cylinder

3.2 Run Analysis

Once everything has been set and the data has been stored correctly in the right place, the script can be executed. To do this, simply start "Main_PostProcessing_inclSensor.m".

Depending on which plot flags you have activated, the evaluation takes between a few seconds and few minutes

3.3 Results

The calculated results are saved in the file "*Result_Analysis_datum*.mat'" in the folder:

...\5_Mapping\Analysis_Mapping\ Result_Mapping

The following calculated variables are saved:

- **x_vec/y_vec/z_vec**
  Position of each magnetic field point as a vector. In our Halbach Mapping, 512 positions were approached (times 8 sensors: 4096 magnetic field points, some points could not be measured and are NaN. NaNs are delated and not saved)
- **Bx/By/Bz**
  Magnetic field strength in mT at each measured point as a vector. Same length as x_vec/y_vec/z_vec
- **x_kin_q/y_kin_q/z_kin_q**
  Interpolated position grid (3D mesh grid) with the desired step size.
- **Bx_kin_ip/By_kin_ip/Bz_kin_ip/B_kin_ip**
  Interpolated magnetic field, divided according to the coordinate axis (as 3D mesh grid).

The result is displayed (depending on the flags set) in a total of 5 plots, which are described below:

1. **Base angle accuracy (Figure 5a)**

The comparison between the measured base angles is shown. On the one hand, Motion Tracking is used and on the other hand the additionally installed sensor itself. In our example, the poorness of the angle measured by Motion Tracking (blue) can be seen directly, as it is extremely noisy. The measured value of the additional sensor is hardly noisy and is stable over time.

2. **FOV comparison (mapped and actually measured), (Figure 5b)**

<u>Left:</u> desired FOV with trajectory for point P5.
<u>Right:</u> Positions actually reached by P5. Small coordinate systems are shown (x=red, y=green, z=blue) to check whether all rotations have been carried out correctly.

3. **Simulation of the Sensor Head trajectory, (Figure 5c)**

The trajectory of the Sensor Head is simulated. This simulation is helpful to check whether the Motion Tracking and the calculated positions are correct and senseful.

4. 3D plot of all mapped positions and the B0 field at the position(Figure 6a)

Left: Plot of the positions of all measured magnetic field points (red dot) with magnetic field vector (blue) at the same position, and only the position of P5 (black cross)

Right: Scatter plot of the measured magnetic field $|\overrightarrow{B_0}|$and positions of P5 (black cross)

5. 3D plot of the interpolated B0 field (Figure 6b)

Scatter plot of the interpolated magnetic field

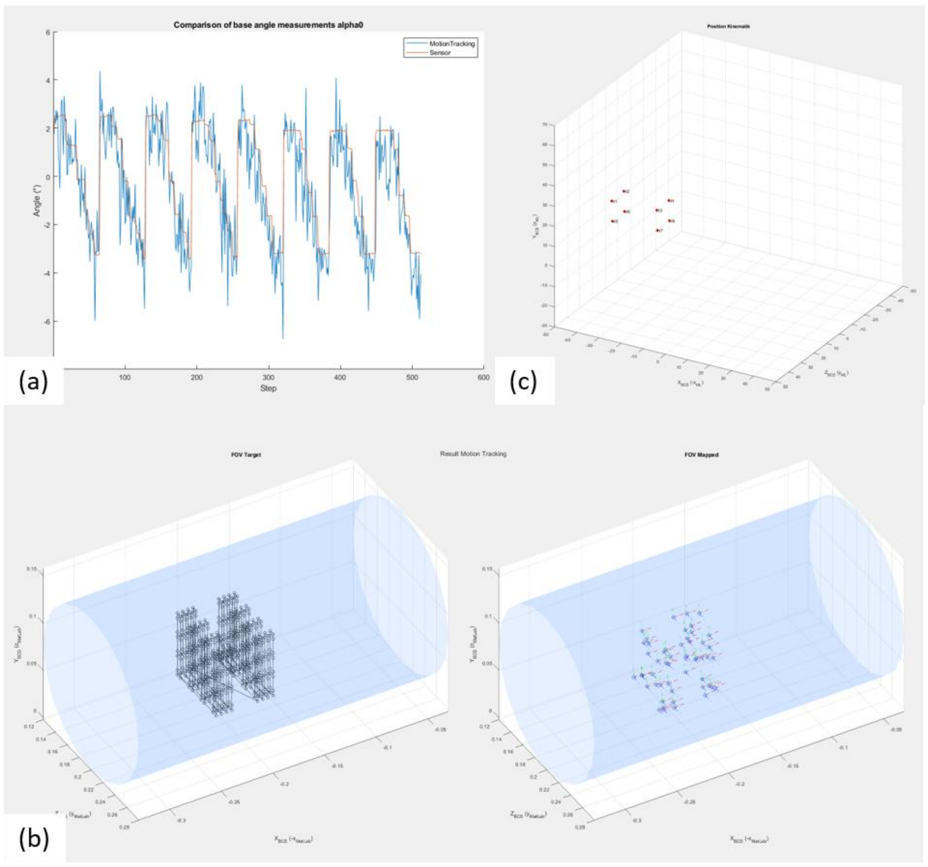


Figure 5: Figure 1/2/3 of the analysis

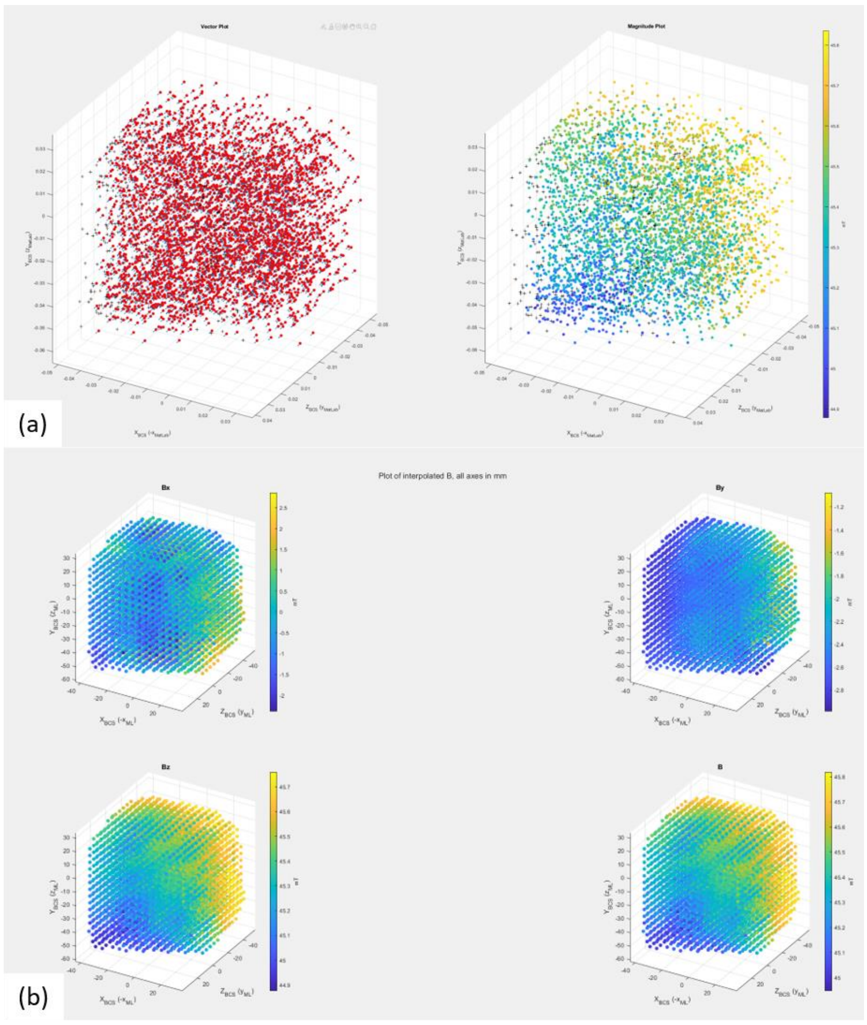


Figure 6: Figure 4/5 of the analysis