Localization Using a Microphone Network

2020-05-06, TSRT14 Sensor fusion

Group ID: 757

Table of Contents

Introduction

A common sensor fusion problem is to localize objects using audio. There are examples of localizing shooters with the help of the distinctive sound and airplanes with the help of radars. [1] In this report, the object is a sound emitting robot that circulates in a trajectory. The sound will be recorded with eight microphones. Two different network setups for the microphones will be used.

Positions of an object can be estimated by using the sound recorded independent. If the object is moving the recorded sound for each time instance will be dependent, and filters can be used for improving the estimated positions and creating a more reliable trajectory. [1]

The purpose of this report is to localize and track a sound emitting robot using a network of microphones. This is done in several steps. The first step is to record data. The two sensor models that is used are two different approaches of Time Different Of Arrival (TDOA). In the sensor model TDOA there is no assumption that the robot is synchronized with the microphones which is the case when data is collected [1].

After creating the sensor models, position localization and tracking can be done. For the two different sensor models the localization is done by a Gauss Newton approach. Tracking is done with two different motion models and two different filtering techniques, resulting in four estimated trajectories. In the end a sensitivity analysis is done to approximate how the uncertainty of the human error will affect the result when measuring the position of the microphones.

Method

The method describes the data collection and the necessary programming in MATLAB to track the robot.

Data Collection

Three different data sets were obtained before the lab. The data sets contained different setups of the microphones. One of the data sets was made for calibration and was used to estimate the measurement precision. The other two data sets had different microphone locations and was used for localization and tracking.

For the data collection eight microphones and one robot were placed on a wooden board with the dimensions 0.991 x 1.222 m^2. The robot played a signal with pulse width 0.1 seconds made of frequencies between 800-1200 Hz. Data was collected during 60 seconds for each experiment. The robot always started at the approximated x-coordinate 0.1 meter and y-coordinate 0.6 meter.

The data sets contained information about the sampling frequency in Hz (*fs*), the positions of the microphones measured by hand given in centimeters (*mic_locations*), the initial position of the robot given in centimeters (*x0*) and the estimated time of arrival pulse per microphone (*tphat*).

The calibration setup can be seen in Figure 1.

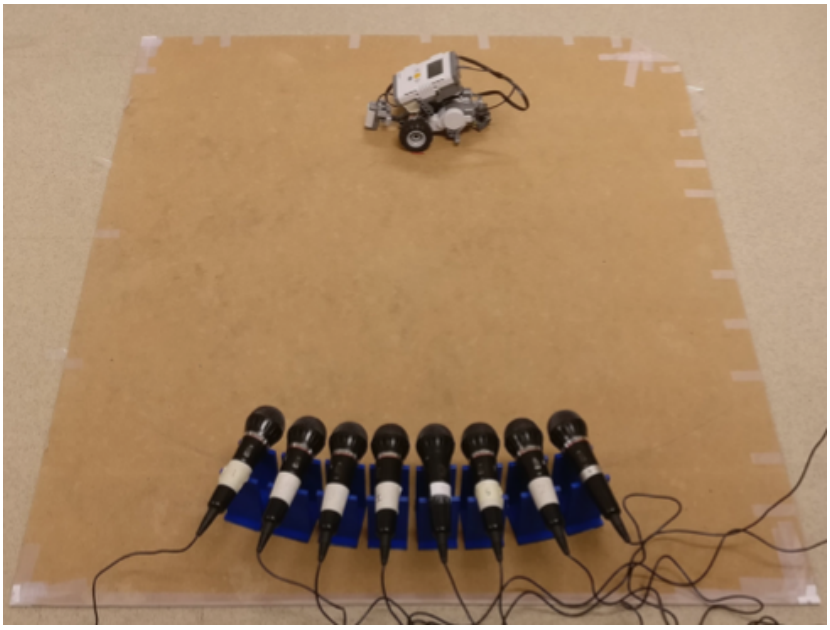

Figure 1. Microphone setup when collecting calibration data.

The calibration setup seen Figure 1 makes sure that it is 0.7 meter between each microphone and the robot. The robot did not move during the data collection. The first microphone in this setup is the one to the left and the eighth microphone is the one to the right. The time between the signal pulse generated by the robot was approximately 0.5 seconds.

For the other two data sets the robot moved along a trajectory. In Figure 2 the first setup can be seen.

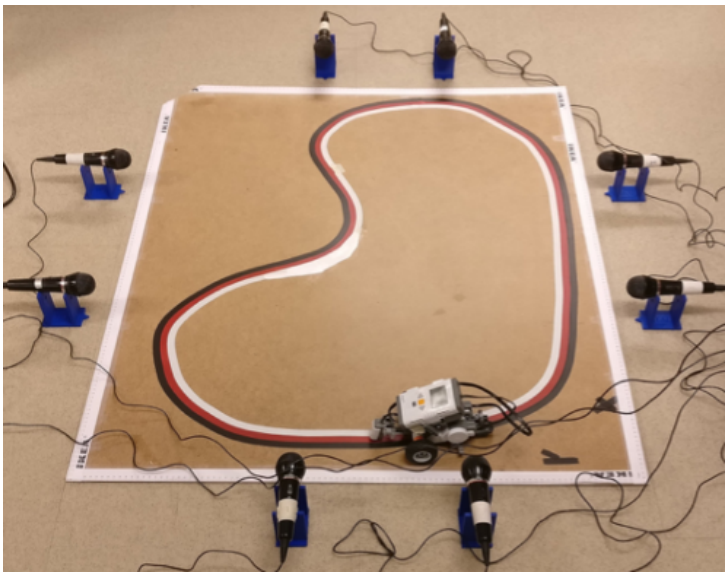

Figure 2. Microphone setup when collecting data to the first setup.

Figure 2 shows how the microphones were arranged for the first setup. Two microphones were placed at each side of the wooden board. The first microphone in this setup is the one to the left in the bottom and the other follows counterclockwise. When the robot is moving around the trajectory there is always at least one microphone nearby with this setup.

Figure 3 shows the second setup.

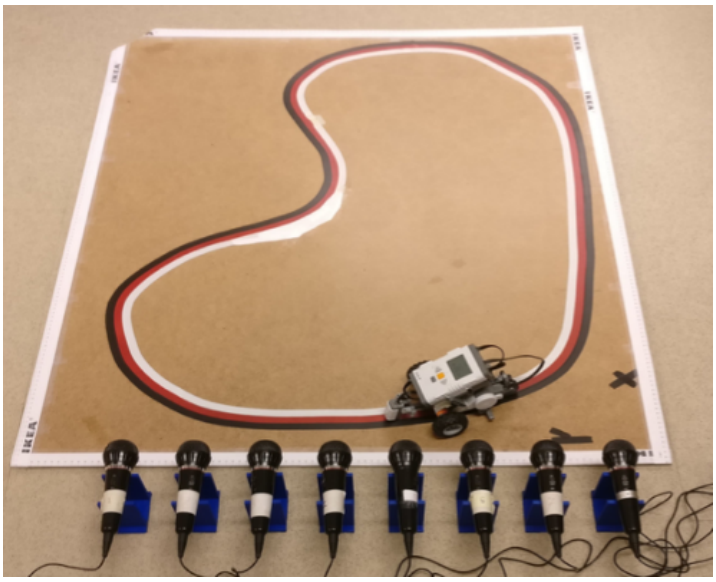

Figure 3. Microphone setup when collecting data to the second setup.

Figure 3 shows how the microphones were arranged for the second setup. All microphones were placed at one side of the wooden board. The first microphone in this setup is the one to the left and the eighth microphone is the one to the right. When the robot is moving around the trajectory all the microphones

will be nearby the robot at the starting point and all microphones will have a long distance to the robot when it has reached the other side of the wooden board.

It is conceivable that the different setups can affect the estimation of the trajectory. Therefore, both setups will be used and analyzed later.

After the data collection was done it seemed like the second microphone in all setups was malfunctioning during data collection. Therefore, all data from the second microphone was removed before proceeding with the rest of the lab. Since the dataset only contains information from seven, and not eight microphones, the names of the microphones have been shifted. The third microphone from Figure 3 is now representing the second microphone and so on.

Sensor calibration

The sensor model was calibrated using the dataset from the calibration.

All microphones have a measurement error at every time instance. The measurement error was calculated by subtracting the mean of every time instance in *tphat* from each of the time instances of the microphones. This is because all microphones are placed from the same distance from the robot and should be receiving the sound at the same time.

Histograms of the measurement errors were computed and compared to a normal distribution. Figure 4 shows the histogram of all microphones and the computed normal distribution.

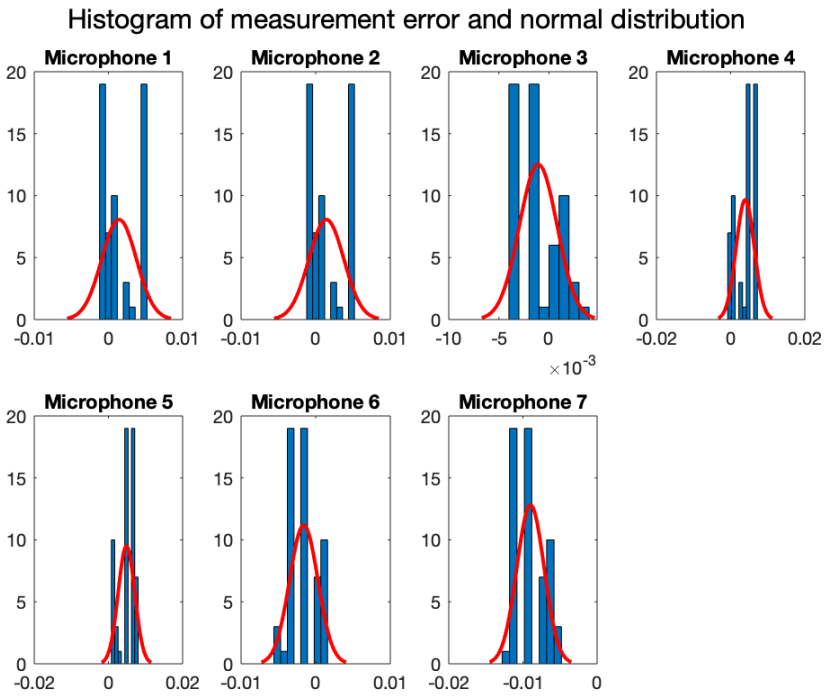

Figure 4. Histograms of the measurement errors for all seven microphones.

For simplicity, the distribution for all the microphones was approximated to be the normal distribution in Figure 4. All the histograms in Figure 4 shows a peak and the variance is reasonable. The normal distributions have the calculated bias as mean and the calculated variance as variance.

The code for generating the histograms and calculating the measurement error can be seen in Appendix A.

Signal Modeling

A sensor model is used to predict a sensor observation. In this report the sensor model is used to predict the position of the sound emitting robot.

The sensor models are based on TDOA and is defined with two different approaches. The first TDOA approach needs information about the time that the sensor is sending out the signal, which is missing from the data collection. [1] This could be solved by estimating the time r_0. A good estimate of r_0 is to assume that the time that the robot sent out the signal is before neither of the microphones received the signal.

The second TDOA approach is another way to solve the problem of the missing information about the time that the signal was sent out. [1] In the second approach, one sensor is chosen as a reference sensor and the model consist of the difference between the other six sensors. Normally the difference would be calculated between all pair of sensors but that would lead to correlated measurements which is not desirable in this model.

The sensor model for the first TDOA approach is given as the equation in Figure 5.

$$y = \begin{matrix} \sqrt{(x_1 - p_{1,1})^2 + (x_2 - p_{1,2})^2} + r_0 \\ \sqrt{(x_1 - p_{2,1})^2 + (x_2 - p_{2,2})^2} + r_0 \\ \sqrt{(x_1 - p_{3,1})^2 + (x_2 - p_{3,2})^2} + r_0 \\ \sqrt{(x_1 - p_{4,1})^2 + (x_2 - p_{4,2})^2} + r_0 \\ \sqrt{(x_1 - p_{5,1})^2 + (x_2 - p_{5,2})^2} + r_0 \\ \sqrt{(x_1 - p_{6,1})^2 + (x_2 - p_{6,2})^2} + r_0 \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} + r_0 \end{matrix} + e$$

Figure 5. Sensor model for the first TDOA approach.

In Figure 5, x_1 represents the unknown x-coordinate and x_2 represents the unknown y-coordinate of the robot. $p_{i,1}$ and $p_{i,2}$ represent the known x- and y-coordinate for sensor i. r_0 is the time when the signal was transmitted from the robot and e is the measurement noise.

For the second TDOA approach, the seventh microphone was used as a reference since that one had the lowest measurement error. By choosing the microphone with the smallest measurement error the covariance matrix will be the most diagonal dominant for all choices. [1]

The sensor model for the second TDOA approach is given as the equation in Figure 6.

$$y = \begin{matrix} \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{1,1})^2 + (x_2 - p_{1,2})^2} \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{2,1})^2 + (x_2 - p_{2,2})^2} \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{3,1})^2 + (x_2 - p_{3,2})^2} \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{4,1})^2 + (x_2 - p_{4,2})^2} \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{5,1})^2 + (x_2 - p_{5,2})^2} \\ \sqrt{(x_1 - p_{7,1})^2 + (x_2 - p_{7,2})^2} - \sqrt{(x_1 - p_{6,1})^2 + (x_2 - p_{6,2})^2} \end{matrix} + e$$

Figure 6. Sensor model for the second TDOA approach.

In Figure 6, the variables represent the same thing as for the first TDOA approach. Note that for the second approach, r_0 is not included.

In order to map between the state of the robot and the collected data *tphat*, two functions are needed which will return y. The m-files for the chosen sensor models can be seen in Appendix B.

Experiments

The two different chosen sensor models can be constructed as sensormod objects in Matlab with the Statistical Sensor Fusion Matlab Toolbox.

For the first TDOA approach the built-in example 'TDOA1' from the Statistical Sensor Fusion Matlab Toolbox was used in order to construct a sensormod object. To adapt the sensormod object to the networks in the first and second setup, the position of the microphones, the initial position of the target and noise distribution were adjusted.

For the second TDOA approach the built-in example 'TDOA2' from the Statistical Sensor Fusion Matlab Toolbox was used in order to construct a sensormod object. The built-in example 'TDOA2' included all 28 pairwise differences. The built-in 'TDOA2' will result in a singular covariance matrix and include linear combinations. This property is not preferable. Since the second TDOA approach described above used a reference microphone and therefore only included seven pairwise differences, the built-in sensormod object was modified. How TDOA2 was modified can be seen in Appendix C.

As for TDOA1, the modified TDOA2 had to adapt to our own network. The position of the microphones, the initial position of the target and noise distribution were adjusted.

The covariance matrix of the noise, R, for the modified TDOA2 got the elements as following:

$$R_{i,j} = \begin{cases} R_{i+1} + R_7, & i = j \\ R_7, & i \neq j \end{cases}$$

R was constructed by adding the measurement error variance of the compared microphones and then insert them into the diagonal of the matrix. The variance of the reference microphone was inserted at the rest of the positions in the matrix. [1]

Appendix C shows the MATLAB code for how TDOA1 and TDOA2 were constructed and modified.

The sensormod objects including the sensor network with the first and second setup can be seen in Figure 7.

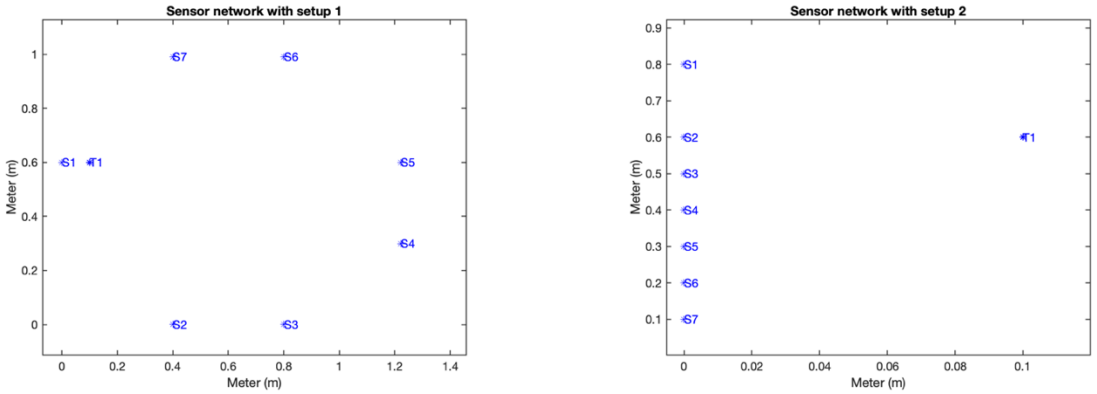

Figure 7. Sensormod objects with sensor networks from the first and second setup. First setup to the left and second setup to the right.

In Figure 7, $S1, S2, S3, S4, S5, S6$ and $S7$ represent the position for the seven microphones and $T1$ is the initial position of the robot.

Configuration analysis

A configuration analysis was done in order to find out which of the two setups that performed better.

Cramér-Rao lower bound (CRLB) was computed as a function of target position for each sensor model and setup. Each grid point from the calculations gave a bound of the Root Mean Sqaure Error (RMSE).

In Figure 8 the CRLB for the first setup is shown for both TDOA approaches.

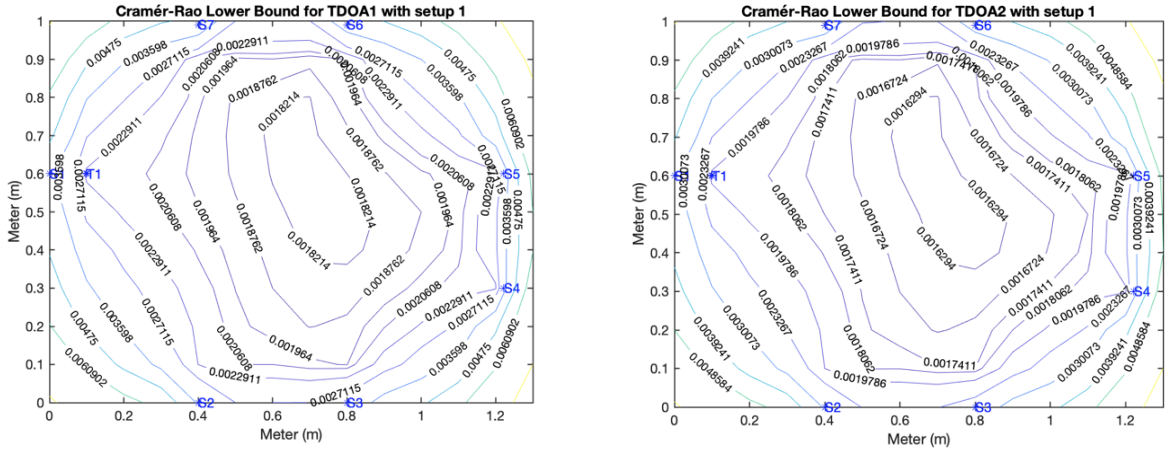

Figure 8. CRLB for the first setup for both TDOA approaches. The first TDOA approach to the left and the second TDOA approach to the right.

In Figure 8, the lowest RMSE is found in the center of the network and RMSE increases further out from the center. The differences between TDOA1 and TDOA2 is few, but TDOA2 performs a little better considering the RMSE for the inner circle. In Figure 9, is a three-dimensional plot of TDOA1 that can be seen in Figure 8.

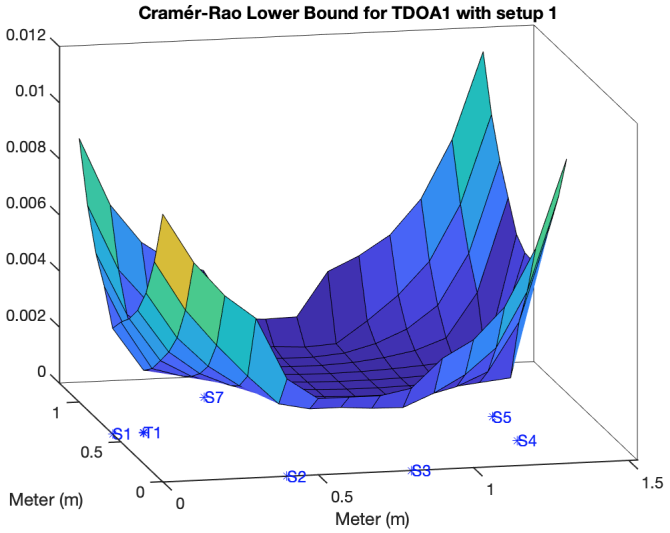

Figure 9. CRLB visualization for the first TDOA approach with the first network setup.

Figure 9 shows another way of visualizing the RMSE and the best estimation will be received in the center of the microphones.

In Figure 10 the CRLB for the second setup is shown for both TDOA approaches.

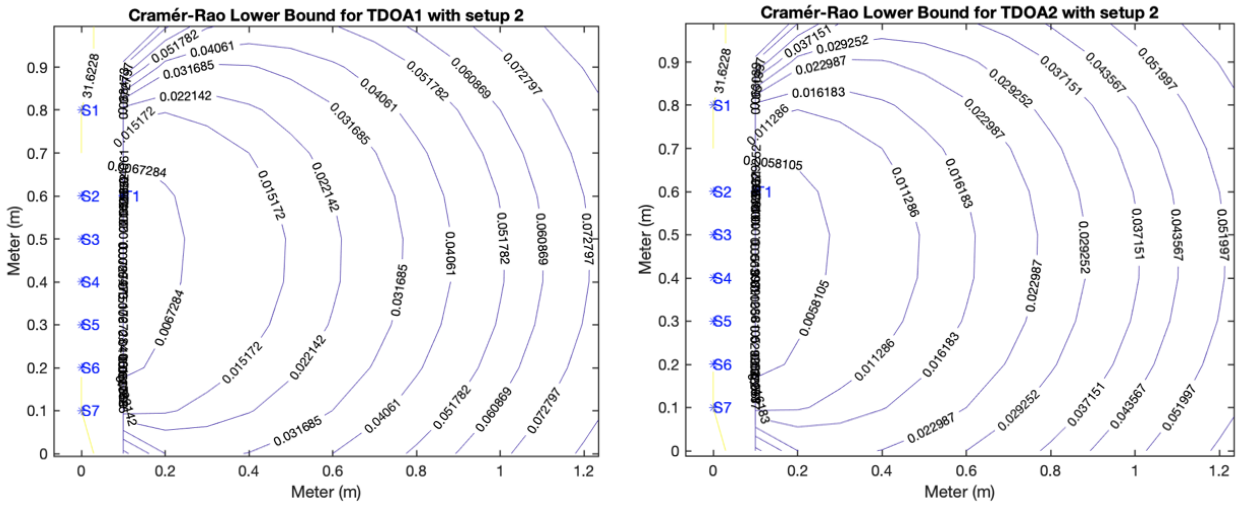

Figure 10. CRLB for the second setup for both TDOA approaches. First TDOA approach to the left and the second TDOA approach to the right.

With a different setup for the microphones, the CRLB grid is changed. Comparing the RMSE for the two setups, the first setup has over all a lower RMSE. The lowest RMSE for the second setup is 0.0058 and is

just a bit larger than the lowest RMSE for the first setup. In the second setup the TDOA2 is slightly better than the TDOA1 which was the case for the first setup as well.

The shape of the grid in Figure 10 implies that the best estimate will be right in front of the microphones. Since the robot will move in the trajectory that is placed in the middle, the first setup is preferred.

From the configuration analysis it is shown that the positions of the microphones affect the measurements. Therefore, only the measurements from the first setup will be used when localizing and tracking the robot.

The Matlab code for the CRLB calculations can be seen in Appendix D.

Localization

The position of the robot at each time instant was estimated with two different localization algorithms.

Each row in *tphat*, which corresponds to the time when the signal pulse sent from the robot reached each microphone, was used independently as input to the localization algorithms in order to get an estimation of the position at every time instant.

For the first localization algorithm, Nonlinear Least Square (NLS) which applies Gaus-Newton algorithm was used for search over $x0$ and $r0$. Since $r0$ was required, the sensormod object for TDOA1 was used as input.

For the second localization algorithm, one sensor was chosen as a reference and the pairwise differences were used as measurements. NLS was therefore used with the modified sensormod object TDOA2 as input.

The NLS function had different properties that could be changed. Different initial start positions for the search direction and different number of iterations in search direction and in the line-search were tested in order to get the best possible estimation of the positions.

The Matlab code for the two localization algorithms can be seen in Appendix E.

Tracking

For the localization problem, the rows in *tphat* was used independent as input to the localization algorithms. Since it is known that the robot is moving around and that the time instances are not independent, motion models with filters can be used in order to improve the estimated trajectory.

Two different motion models where chosen in order to describe the states of the robot. The first motion model included four states which represents the positions and velocity of the robot.

$$x(t) = \begin{pmatrix} p_x(t) \\ p_y(t) \\ v_x(t) \\ v_y(t) \end{pmatrix}$$

The robot was modeled to move with constant velocity and therefore the Constant Velocity (CV) model was used. The discrete time function of the CV model is as follows

$$x(t+T) = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x(t) + v(t)$$

where $v(t)$is the process noise and T is the sample time. [1]

The built-in example 'cv2D' from the Statistical Sensor Fusion Matlab Toolbox was used in order to construct the linear time invariant (LTI) object as the described CV model. The built-in example had $T = 0.5$ which is reasonably in our case as well.

The second motion model included six states which represents the positions, velocity and acceleration of the robot.

$$x(t) = \begin{pmatrix} p_x(t) \\ p_y(t) \\ v_x(t) \\ v_y(t) \\ a_x(t) \\ a_y(t) \end{pmatrix}$$

The robot was modeled to move with constant acceleration instead of velocity and therefore the Constant Acceleration (CA) model was used. The discrete time function of the CA model is as follows

$$x(t+T) = \begin{pmatrix} 1 & 0 & T & 0 & T^2/2 & 0 \\ 0 & 1 & 0 & T & 0 & T^2/2 \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} x(t) + v(t)$$

where $v(t)$is the process noise and T is the sample time. [1]

The built-in example 'ca2D' from the Statistical Sensor Fusion Matlab Toolbox was used in order to construct the linear time invariant (LTI) object as the described CV model. The same value of T was used in this case.

Two different filters were chosen for both motion models so that they later could be compared. The first filter chosen was a Kalman Filter (KF). For the first filter, the localization estimates from the TDOA1 approach was used as artificial measurements at every time instance. The linear motion model and the artificial measurements was used as input to the filter.

The second filter chosen was an Extended Kalman Filter (EKF). The EKF can handle nonlinear models, and therefore the generated linear models of CV and CA was converted to a nonlinear model structure. For the EKF, the sensor model for TDOA2 was included in each of the motion models. The nonlinear motion model and the TDOA2 measurements was used as input to the filter. The EKF will now compare the given

measurements and the position that should be calculated and compensate for that [1]. The result was two different estimated trajectories.

The big differences between the first and second filter is that the first filter use localization estimates as measurements and the second filter use the TDOA2 measurement model with six pairwise differences.

The Signal to Noise Ratio (SNR) is defined as $\|Q\|/\|R\|$ where Q represent the covariance of the signal noise and R represent the covariance of the measurement noise. SNR tells how fast the filter adopt to changes. In order to tune the filters, R can be fixed while Q can be multiplied with different factors. [1]

Different factors were tried in order to tune the model. The one that gave the best match to the true trajectory was chosen. The Matlab code for the motion models and filters can be seen in Appendix F.

Sensitivity analysis

When collecting data, the positions of the microphones was measured by hand. Since there is always a human error, the positions could be measured wrong. In the sensitivity analysis the uncertainty by measuring by hand will be evaluated to see how this could affect the result when tracking the robot.

The sensitivity analysis was done for the KF using both motion models. It is reasonable that one could not measure more than one centimeter wrong if the data collection is done carefully. Therefore, the microphones locations were offset with one centimeter.

In Figure 11, the original and disturbed positions for the microphones can be seen.

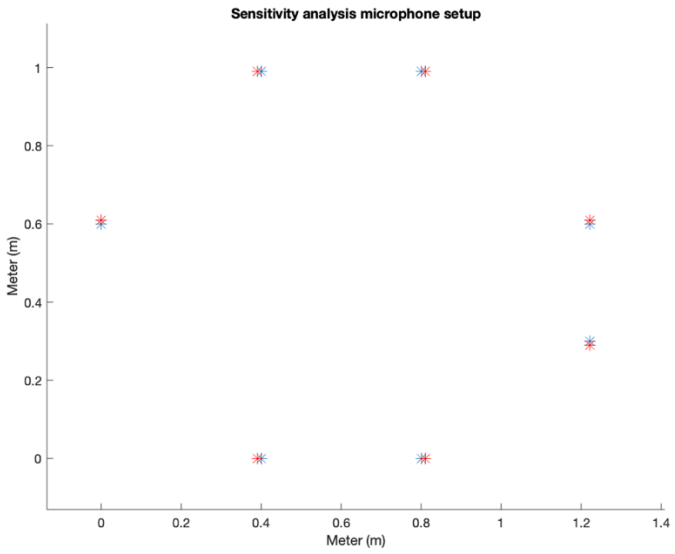

Figure 11. Microphone position for both measured and disturbed positions.

In Figure 11, blue dots are the measured positions of the microphones and the red dots are the disturbed position where the positions have been offset the most one centimeter each in any direction.

In order to perform the sensitivity analysis, the Matlab code in Appendix C was changed in order to disturb the positions. The tracking was made in the same way as in Appendix F.

Result

The result includes the estimated trajectories for localization, tracking and the sensitivity analysis.

Localization

The estimated trajectories for the localization algorithms can be seen in Figure 12.

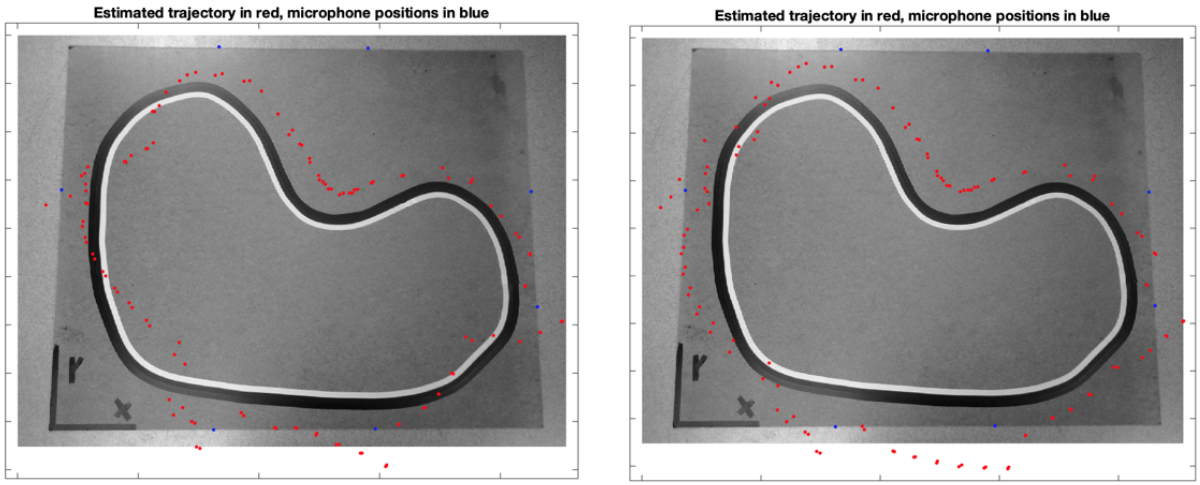

Figure 12. Estimated position using localization algorithm for TDOA1 and TDOA2. TDOA1 to the left and TDOA2 to the right.

In Figure 12, the background is a picture of the trajectory from when collecting data and the red dot represent the estimated positions. Both localization algorithms are mostly estimating positions outside the trajectory. Comparing the two localization algorithms there is not much differences and it is not possible to tell which one that is the best on to use.

Tracking

The estimated trajectories for the tracking algorithms can be seen in Figure 13 and Figure 14.

The both filters, KF and EKF were tuned by multiply the covariance matrix of the signal noise and measurement noise with different factors.

Figure 13 shows the result from KF and EKF with the motion model CV.

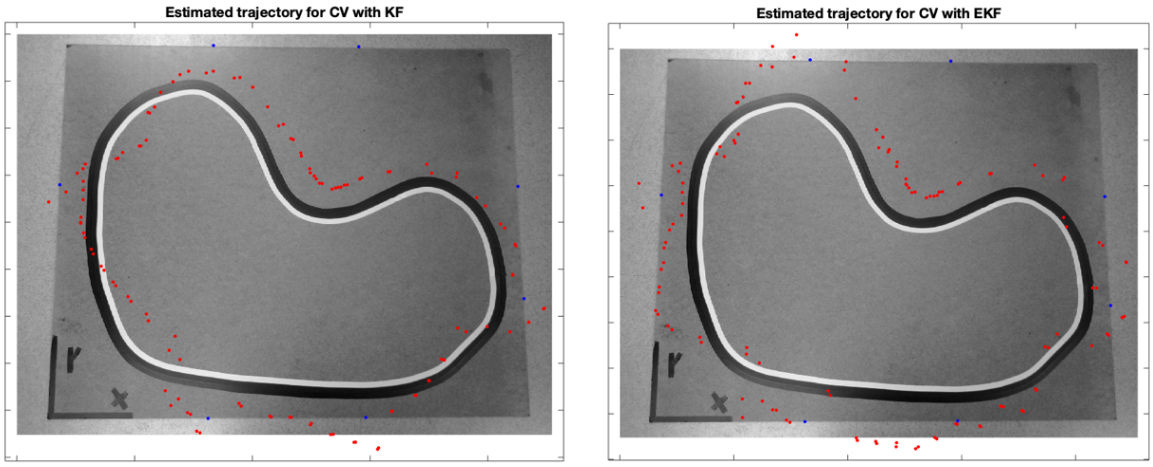

Figure 13. Estimated trajectories for the tracking algorithms using CV. KF to the left and EKF to the right.

In Figure 13 to the left, the estimated localizations from TDOA1 and the linear motion model CV have been filtered using KF. In Figure 13 to the right, the TDOA2 sensor model have been added to the nonlinear motion model CV and filtered using EKF.

The filtered estimations are not as spread out compared to the non-filtered estimated position in Figure 12. The estimations also follow the shape of the trajectory better when filtered even if they do not fit the trajectory perfect.

If the two trajectories in Figure 13 is compared, it can be seen that they have different shapes. That can depend both on which input that is used for the different filters but also which algorithm that was used. In this case it seems that the trajectory to the left, when estimated locations from TDOA1 and KF was used gave a better estimate of the trajectory.

Figure 14 show the result from KF and EKF with the motion model CA.

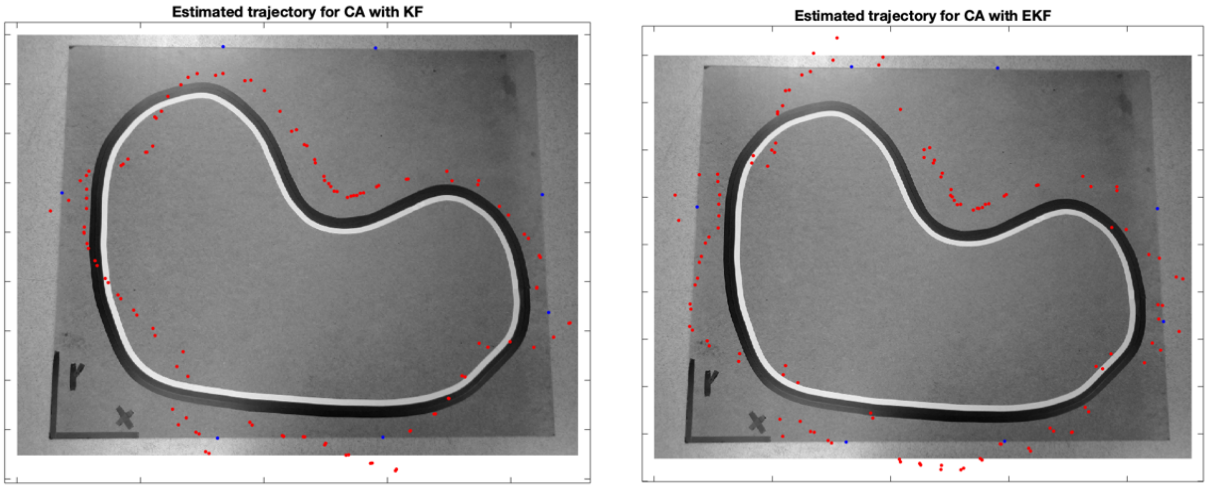

Figure 14. Estimated trajectories for the tracking algorithms using CA. KF to the left and EKF to the right.

As for the tracking using the motion model CV, both the KF and the EKF gives a smoother trajectory with less spread-out estimations. The resulting trajectories in Figure 14 is very similar to the ones when the motion model CV was used in Figure 13.

The estimated states with confidence bounds compared to the true ones for the motion model CV can be seen in Figure 15.

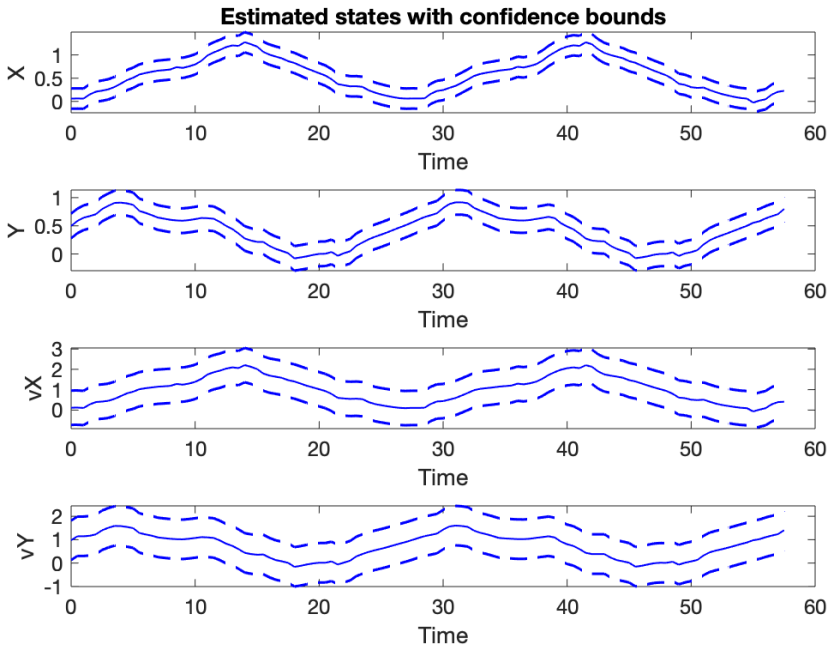

Figure 15. Estimated states with confidence bound for CV as motion model.

Figure 15 shows the confidence bounds for one of the motion models and the other motion model had confidence bounds as expected. The more states that are included in the motion model the larger confidence bounds for those states. In this case, the measurements do not include velocity, it is calculated based on the positions and the time instances. If acceleration is added as a state as well, it is calculated from the velocity and the confidence bounds will be even bigger. The uncertainty will increase with the number of states.

Sensitivity analysis

The result from the sensitivity analysis can be seen in Figure 16.

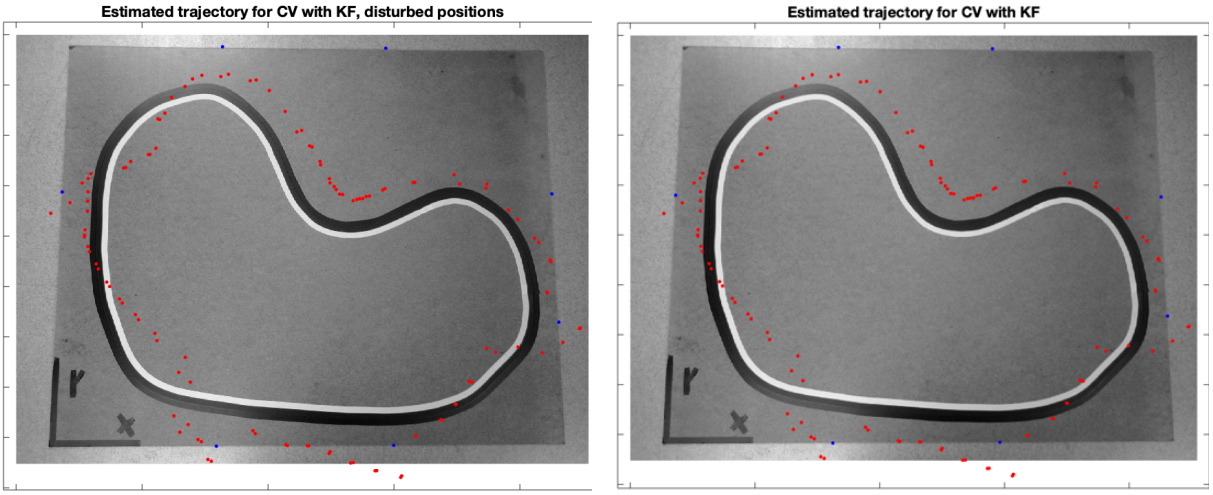

Figure 16. Estimated trajectory with disturbed microphone positions to the left and measured microphone positions the right.

Figure 16 shows the estimated trajectories for the tracking algorithm when KF and the motion model CV was used with both disturbed and measured positions.

In Figure 16, the offset of the positions of the microphones does not affect the result significantly. When KF and the motion model CA was used the same result was obtained.

Discussion

From the result we could see that the trajectories for the motion model CV and CA were very similar. The only difference between the two models is that the CA also contains states about the acceleration. Many other different motion models can be constructed as well. Another motion model may have resulted in a more different result between the chosen motion models.

As there are other motion models to choose, it also exists many different filters with different parameters. Therefore, it exists many combinations of different motion models and filters to use which probably would lead to different results than presented above.

When tuning the KF and EKF, the covariance matrices for the measurements noise and the signal noise was modified. Different reasonable values and combinations was tried, and the values that made the best fit to the trajectory was chosen. Since this was made by hand it was not possible to cover all combinations. Probably we did not find the optimal solution. More testing would therefore probably result in a better filtering.

A recurring pattern from all estimated trajectories is that the estimation tends to move slightly towards the center. This might be because of the microphone setup. As seen from the comparison of the first setup and the second setup, the setup affects the estimations. There can be a better setup than our best setup, the first setup. Figure 9 also confirm that the estimations will be worse in the corner. Probably more microphones would result in a better fit.

Another aspect regarding the microphone setup is the fact that the second microphone was malfunctioning and the data from the second microphone is missing in the estimations. If the second microphone would have worked than the estimations would probably be better.

Conclusion

To conclude, it is possible to track a sound emitting robot but there are several aspects that effects the quality of the estimations. The first understanding is that the setup of the microphones has a big influence and it is not certain that the first setup is the best possible microphone setup. The second understanding is that there is not a big difference between the motion models CV and CA regarding estimations, but the choice of filter matter the most. The last understanding was that measuring the positions of the microphones a little bit wrong does not affect the estimations.

References

[1] F. Gustafsson. *Statistical Sensor Fusion.* Studentlitteratur, Lund, Sweden, Third Edition, 2018.

Appendix A – Sensor Calibration

Appendix A includes the main code for the sensor calibration step.

```
load('calibration.mat');
% Remove the second mic
nr_of_mics = 7;
tphat(:,2) = [];

% Scale from seconds to meter
tphat = tphat*343;

for i = 1:nr_of_mics
    % Calculate measurement error, bias, variance and
    % standard deviation for each microphone
    e(:,i) = tphat(:,i) - mean(tphat, 2);
    bias(i) = mean(e(:,i));
    variance(i) = var(e(:,i));
    standDev(i) = std(e(:,i));

    % Plot the histogram
    subplot(2, 4, i);
    histfit(e(:,i))
    title(['Microphone ', num2str(i)]);

end
sgtitle('Histogram of measurement error and normal distribution');
```

Appendix B – Signal Modeling

Appendix B includes two functions that returns the SIG-object y depending on the sensor model.

```
function [y] = y_TDOA1(tphat)
    y = sig(tphat,2);
end
```

```
function [y] = y_TDOA2(tphat, mic_loc)
for i = 1:length(tphat)
     for j =[1:6],
        y(i,j) = tphat(i,7)-tphat(i,j);
     end
end
y = sig(y, 2);
end
```

Appendix C – Experiments

Appendix C includes main code for the experiment and code for the function *get_R*.

```
load('setup1.mat');
% Remove the second mic
tphat(:,2) = [];

% Scale from seconds to meter
tphat = tphat*343;
% Scale from centimeters to meter
mic_locations = mic_locations'/100;
mic_locations(:,2) = [];
x0 = x0/100;

% New locations used in Sensitivity analysis
% mic_locations2 =  [0       0.3900    0.8100    1.2220    1.2220    0.8100    0.39000;
                  0.6100          0         0    0.2900    0.6100    0.9910    0.9910];

% Create the first sensormod object, TDOA1
sm_TDOA1 = exsensor('tdoa1', nr_of_mics);
sm_TDOA1.th = mic_locations(:);
sm_TDOA1.x0 = [x0, 0];
sm_TDOA1.pe = diag(variance);

% Create the second sensormod object, TDOA2
h = inline('[sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(1)).^2+(x(2,:)-
th(2)).^2);sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(3)).^2+(x(2,:)-
th(4)).^2);sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(5)).^2+(x(2,:)-
th(6)).^2);sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(7)).^2+(x(2,:)-
th(8)).^2);sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(9)).^2+(x(2,:)-
th(10)).^2);sqrt((x(1,:)-th(13)).^2+(x(2,:)-th(14)).^2) - sqrt((x(1,:)-th(11)).^2+(x(2,:)-
th(12)).^2)]', 't', 'x', 'u', 'th');
sm_TDOA2 = sensormod(h, [2 0 6 14]);
sm_TDOA2.th = mic_locations(:);
sm_TDOA2.x0 = x0';
sm_TDOA2.pe = get_R(variance);
% Plot network (setup1 or setup2)
sm_TDOA2.plot
% sm_TDOA1.plot
```

```
function [ R ] = get_R(mic_var)
% Returns the covariance matrix R
    nr_of_mics = length(mic_var);


    % Let all elements in matrix R be the variance of microphone 7
    R = mic_var(7)*ones(6,6);


    i = 1;
    for k = 1:nr_of_mics-1
       % Update the values for the diagonal in matrix R
       y_var =  mic_var(7) + mic_var(k);
       R(i,k) = y_var;
       i = i+1;
    end
end
```

Appendix D – Configuration Analysis

Appendix D includes main code for the configuration analysis.

```
%Plot the CRLB for TDOA1 grid and display the RMSE
figure(1)
crlb2(sm_TDOA1, [], 0:.1:1.3, 0:.1:1, [1,2], 'rmse');
axis([0 1 0 1])

% To plot 3D graph
% [cx, X1, X2] = crlb2(sm_TDOA1, [], 0:.1:1.3, 0:.1:1, [1,2], 'rmse');
%  surf(X1, X2, cx);

% Plot the CRLB for TDOA2 grid and display the RMSE
figure(2)
crlb2(sm_TDOA2, [], 0:.1:1.3, 0:.1:1, [1,2], 'rmse');
axis([0 1 0 1])
```

Appendix E – Localization

Appendix E includes main code for the two different localization algorithms used.

```
% NLS, Gauss-Newton, TDOA1

% Get measurements for TDOA1 using function
y = y_TDOA1(tphat);


% Set the start position and the first r0
sm_TDOA1.x0 = [x0, min(tphat(1,:))-0.002];

for i = 1:length(tphat)
    % Use estimate to estimate the position
    [xhat1, res] = estimate(sm_TDOA1, y(i,:), 'maxiter',
1000, 'ctol', 1e-7);


   % Store the estimated position
   estimated_pos(:,i) = xhat1.x0(1:2);


   % Update x0 and r0
   sm_TDOA1.x0 = [xhat1.x0(1:2); min(tphat(i+1,:))-0.002];

end


SFlabCompEstimGroundTruth(estimated_pos, mic_locations);
```

```
% NLS, Gauss-Newton, TDOA2

% Get measurements for TDOA2 using function
y = y_TDOA2(tphat, mic_locations);


% Set the start position
sm_TDOA2.x0 = [x0];


for i = 1:length(tphat)
    % Use estimate to estimate the position
    [xhat2, res] = estimate(sm_TDOA2, y(i,:), 'maxiter',
1000, 'ctol', 1e-7);


    % Store the estimated position
    estimated_pos(:,i) = xhat2.x0(1:2);

    % Update x0
    sm_TDOA1.x0 = [xhat2.x0(1:2)]

end


SFlabCompEstimGroundTruth(estimated_pos, mic_locations);
```

Appendix F – Tracking

Appendix F includes main code for the two different filters used.

```
% Filter with estimated positions as artificial
measurments

% Load workspace from Localization TDOA1
load('artificial_measurments_TDOA1.mat')
z = estimated_pos;
y = sig(z');

% Change motion model (cv2D or ca2D)
%m = exlti('cv2D');
m = exlti('ca2D');
% Improving the filter
m.Q = 1 * m.Q;

% Apply the Kalman filter
xhat_TDOA1 = kalman(m, y);

% Plot the confidence bounds
xplot(xhat_TDOA1, sig(z).y, 'conf', 90);

% Plot the trajectory
SFlabCompEstimGroundTruth(xhat_TDOA1.y',
mic_locations);
```

```
% Filter with one sensor as reference as sensor
model

% Get y values from sensor model TDOA2
y = y_TDOA2(tphat, mic_locations);

% Change motion model (cv2D or ca2D)
%m = exmotion('cv2D');
m = exmotion('ca2D');

% Combine the motion and sensor models
m1 = addsensor(m, sm_TDOA2);

% Improving the filter
m1.pv = 0.1 * m1.pv;
m1.pe = 10* m1.pe;

% Set intital values for position, speed and
%acceleration, depending on which motion model
%m1.x0 = [x0 0.1 0.1];
m1.x0 = [x0 0.1 0.1 0 0];

% Apply the Extended Kalman filter
xhat_TDOA2 =ekf(m1, y);

% Plot the trajectory
SFlabCompEstimGroundTruth(xhat_TDOA2.x(:,1:2)',
mic_locations);
```