SoFi - Direction of arrival estimation using commodity hardware

Fachbereich 1 - Physics and Electrical Engineering
Institute for Telecommunications and High-Frequency Techniques (ITH)
Department of Communications Engineering
P.O. Box 33 04 40
D-28334 Bremen

Supervisor:	Johannes Demel
Date:	07.05.2017

I ensure the fact that this thesis has been independently written and no other sources or aids, other than mentioned, have been used.

Bremen, May 7, 2017 _____

1 Abstract

This thesis aims to explore if a low-cost direction of arrival (DOA) estimation system can be implemented by utilizing commodity software defined radio (SDR) hardware. And if common background noise at the different receivers can serve as a sufficient means of time-domain synchronization.

The measurement rig is intended to measure the angle of incident of multiple FM-Radio signals which should, assuming known broadcasting-tower locations, allow estimations of the current receiver location on a 2D-plane.

The rig utilizes DVB-T USB donles that where discovered [1] to contain a feature to pass raw I/Q ADC samples to the host computer, making it a receive-only SDR.

Most DOA algorithms require tight synchronization between the receiving units in time and frequency. To meet these requirements DOA rigs usually employ custom hardware and/or dedicated reference signal sources.

To explore the minimal requirements for a sucessful synchronization, the rig at hand uses only minimal hardware modifications to the stock USB dongles to lock the receiver frequencies and to prevent phase drifting.

The rig also does not use a dedicated reference signal for timing synchronization, but instead relies on correlations in the noise-floor of the receivers to synchronize them using the noise cross-correlation.

By utilizing consumer-grade SDR and computing hardware a DOA-rig can be assembled for about 50 €, assuming the availability of a sufficiently powerful computer.

2 Introduction

Direction of arrival estimation of RF-Signals works by analyzing the signals received by multiple antennas distributed in space.

The common approach is to exploit the fact, that the signals received at antennas further away from a source are more affected by the channel they are transmitted over.

The effect of the channel can be measured by observing the signal strength and/or phase at the receiving antennas.

For systems where the distance between the sending and the receiving antennas is much smaller than the distance between the receiving antennas, the difference in signal strength are usually very small and using the phase differences provides better accuracy without requiring more accurate receivers.

The system implemented in this thesis will use SDR technology, fourier transformations and a basic direction of arrival estimation algorithm to calculate DOA information for all signals present in a given frequency range. More sophisticated algorithms like MUSIC employ a cross-correlation Matrix as an early processing step, allowing for better separation of signals and noise but also limiting the number of signals analyzed at a time to be less than the number of receivers.

Using multiple cheap, independent receivers instead of purpose-built hardware also poses some challenges that have to be overcome and will be discussed in this thesis, like different LO frequencies and differences in sampling time.

2.1 DOA principles

Figure 1 visualizes how signal phases can be used to estimate the direction of arrival of a signal in the far field.

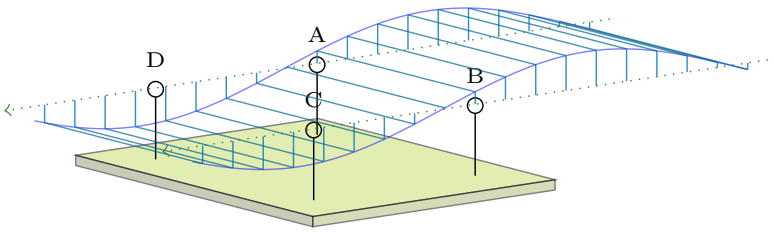

Figure 1: Array of four antennas with a transmitter to the far right

The diagram shows four antennas spaced out on a plane and a stilized wave that is received by all four of them. The source of the wave is assumed to be so far away, that the signal-rays are approximately parallel when they reach the receivers.

In the depicted constellation the signal has to travel the same distance to reach the

receivers A and B, thus the phases at those receivers are the same. On the other hand the signal has to travel further to reach receiver D than to reach A, the difference in phase can be directly calculated from the wavelength λ of the signal and the distance d_{AD} between the antennas.

$$\Delta\varphi = 2\pi \cdot \frac{d_{\mathrm{AD}}}{\lambda}$$

In a more realistic scenario, the the signals will reach the receivers at odd angles. Figure 2 shows an example where a signal impinges on the receivers at an angle of $\alpha = 45°$. The length, that can be calculated from the phase difference $\Delta\varphi$ between the receivers and the wavelength λ of the signal, is marked as Δl.

The direction of arrival α can then be calculated from those lengths using equation 1.

$$\alpha = \arccos\left(\frac{\Delta l}{d}\right) = \arccos\left(\frac{\Delta\varphi \cdot \lambda}{2\pi d}\right) \tag{1}$$

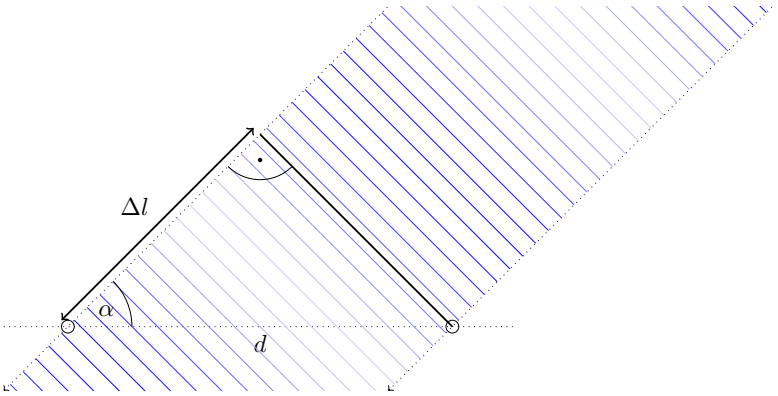

Figure 2: Signal arriving at an angle of 45°

2.2 Noise influence

It is assumed that some of the noises received by the receivers stem from external physical phenomena. These phenomena could be e.g. transients, creating wide-band pulses.

It is also assumed, that the probability of these noise-generating events is equal for every direction of incidence on the antenna array.

If these events would occur sporadically and the analyzed timeframe for each DOA estimation was small, each one of them would yield a distinct DOA estimation result, according to the direction of incidence.

In this thesis the time per DOA estimation is assumed to be large with respect to the mean time between these events. Thus the direction information results of these events

can be assumed to cancel out.

A phase difference between two receivers does, as shown in the previous section, imply a distinct direction of arrival. But, as the direction information of the noise-signals is expected to cancel out when analyzed over a long time-span, the phase differences also have to cancel out.

In a competely noise-based signal the measured phase differences would thus be, on average, zero.

In a compelely noise-free signal the phase differences would, as shown in the previous section, correspond to the antenna distances, wavelength of the signal and the angle of incidence.

It is assumed, that in cases where a signal is overlayed by noise, both effects will contribute to the measurement, yielding phase differences that are an average of the expected value in the noise-free case and the expected value (zero) in the noise-only case, weighted by the signal-to-noise ratio (SNR).

The DOA estimation algorithm will thus have to be able to work with phase differences that were scaled by a factor f that depends on the SNR. As the SNR of the closely-spaced antennas can be assumed to be approximately the same, the factor f can also be assumed to be approximately equal for every antenna pair.

2.3 Basic direction estimation

As discussed earlier the direction estimation works by analyzing differences in signal phase between receivers distributed in space.

It is assumed, that the signals s_0 ... s_n can be perfectly separated in the frequency domain, meaning that their bandwith is a lot smaller than the differences of their center frequencies and that the direction of incidence does not change rapidly for a fixed frequency.

FM-radio stations fulfill these requirements, as they are intended to be easily separable by frequency and the (stationary) transmission towers continously broadcast at the same frequency.

One example where these assumtions do not hold is UMTS, where transmitters are not completely separable by frequency due to code multiplexing and where the locations of active transmitters might change quickly due to time-sharing.

The algorithm used in this thesis operates completely in the frequency domain, so the time-domain SDR samples are fourier-transformed, prior to the DOA estimation, using an fast fourier-transformation (FFT). The FFT decomposes the signal into a pre-defined

number of frequency bins.

The same signal, received at different receivers, will be, according to its frequency, sorted into the same frequency bin in its respective FFT.

The phase information of the signal will be preserved when it is transformed by the FFT. This means, that the signal phase can be analyzed by analyzing the phase of the corresponding FFT bin. The phase difference $\Delta\varphi_n$ between two receivers for a the nth frequency bin can be calulated using the complex conjugate multiplication, as seen in equation 2, where $S_{1,\mathrm{n}}$ and $S_{1,\mathrm{n}}$ are corresponding frequency bins in the FFT of receivers one and two and A_n is a real valued scaling factor.

$$A_n \cdot e^{j\cdot\Delta\varphi_n} = S_{1,\mathrm{n}} \cdot S^*_{2,\mathrm{n}} \tag{2}$$

To estimate the angle of incidence for a signal in a specific FFT bin a vector $\vec{a}_s$ of the calculated phase differences for every receiver pair is constructed, where e.g. $\varphi_{1,2}$ is the phase difference between receiver 1 and 2 for that FFT bin:

$$\vec{a}_s = \begin{pmatrix} \varphi_{1,2} \\ \varphi_{1,3} \\ \varphi_{1,4} \\ \varphi_{2,3} \\ \varphi_{2,4} \\ \varphi_{3,4} \end{pmatrix}$$

Additionally a vector $\vec{b}(\theta)$ is constructed that contains the expected phase differences between the receivers based on the wavelength λ, the antenna distance d and the direction of incidence θ.

The scalar dot product of both vectors $\vec{a}_s$ and $\vec{b}(\theta)$, when swept over θ, will have a maximum (at $\theta_{\max}$) when both vectors are parallel and a minimum (at $\theta_{\min}$) when they are anti-parallel.

The vectors $\vec{a}_s$ and $\vec{b}(\theta_{\max})$ being parallel means, that the phase differences in $\vec{a}_s$ match the expected phases difference for the angle of incidence $\theta_{\max}$. Thus $\theta_{\max}$ is the estimated direction of arrival.

2.4 Receiver offsets: frequency

SDR devices are commonly structured as seen in figure 3, the antenna signal S_{in} is first amplified by a factor of A_{lna} by an LNA and then mixed down using an I/Q - LO signal S_{lo} to form a complex baseband signal S_{bb}.

$$S_{\mathrm{bb}} = S_{\mathrm{in}} \cdot A_{\mathrm{lna}} \cdot \left(S_{\mathrm{lo,i}} + jS_{\mathrm{lo,q}}\right) = S_{\mathrm{in}} \cdot A_{\mathrm{lna}} \cdot e^{j\cdot(2\pi f_{\mathrm{lo}} t + \varphi_{\mathrm{lo}})} \tag{3}$$

To an user of an SDR device the actual waveform of the local oscillator (LO) is usually irrelevant and can be modelled as a complex sinusiod of frequency f_{lo} and phase φ_{lo}, as seen in equation 3.

As discussed earlier DOA estimation uses the phase differences between signals received at different receivers to analyze the difference in signal runtime.

A phase difference between two complex signals can be calculated using complex conjugate multiplication as shown in 4.

$$
\begin{aligned}
A \cdot e^{j \cdot \Delta\varphi} &= S_{\mathrm{bb},1} \cdot S^*_{\mathrm{bb},2} && (4)\\
&= S_{\mathrm{in},1} \cdot e^{j \cdot (2\pi f_{\mathrm{lo},1} t + \varphi_{\mathrm{lo},1})} \cdot S^*_{\mathrm{in},2} \cdot e^{-j \cdot (2\pi f_{\mathrm{lo},2} t + \varphi_{\mathrm{lo},2})} \\
&= S_{\mathrm{in},1} \cdot S^*_{\mathrm{in},2} \cdot e^{j \cdot (2\pi (f_{\mathrm{lo},1} - f_{\mathrm{lo},2}) t + \varphi_{\mathrm{lo},1} - \varphi_{\mathrm{lo},2})} \\
&= S_{\mathrm{in},1} \cdot S^*_{\mathrm{in},2} \cdot e^{j \cdot (2\pi \Delta f_{\mathrm{lo}} t + \Delta\varphi_{\mathrm{lo}})} && (5)
\end{aligned}
$$

If the two signals are baseband signals that were downconverted using two LO signals with respective frequencies of $f_{\mathrm{lo},1}$ and $f_{\mathrm{lo},2}$ and phases of $\varphi_{\mathrm{lo},1}$ and $\varphi_{\mathrm{lo},2}$ the difference of these frequencies and phases will also be present in the resulting phase difference $\Delta\varphi$, as shown in equation 5.

In order to get the actual phase difference between the two input signals $S_{\mathrm{in},1}$ and $S_{\mathrm{in},2}$ the LO frequencies and phases have to be locked ($\Delta f_{\mathrm{lo}} = 0$, $\Delta\varphi_{\mathrm{lo}} = 0$).

2.5 Receiver offsets: time

Another receiver offset with negative impact on DOA performance is an offset in sampling-time.

If the same signal $s_{\mathrm{in}}(t)$ is received by two receivers a and b the samples wil be delayed by the delays t_{a} and t_{b} before being processed to extract phase informations.

These delays are introduced by unmached cable lengths, causing runtime differences, digitizing delays and time the samples spend in memory buffers.

When these delayed signals are fourier transformed the delays result in rotating signal phases [2, p. 802], as seen in equation 6, where $S_{\mathrm{in}}(j\omega)$ is the fourier transformation of $s_{\mathrm{in}}(t)$.

$$
\begin{aligned}
\mathcal{F}\left\{ s_{\mathrm{in}}(t - t_{\mathrm{a}}) \right\} &= e^{j\omega t_{\mathrm{a}}} S_{\mathrm{in}}(j\omega) && (6)\\
\mathcal{F}\left\{ s_{\mathrm{in}}(t - t_{\mathrm{b}}) \right\} &= e^{j\omega t_{\mathrm{b}}} S_{\mathrm{in}}(j\omega)
\end{aligned}
$$

When the phase difference between these two fourier transformed signals is calculated using complex conjugate multiplication the difference in the processing delays will also manifest in the output phase.

$$\mathcal{F}\left\{s_{\text{in}}(t-t_{\text{a}})\right\} \cdot \mathcal{F}\left\{s_{\text{in}}(t-t_{\text{a}})\right\}^{*} = e^{j\omega t_{\text{a}}} S_{\text{in}}(j\omega) \cdot e^{-j\omega t_{\text{b}}} S_{\text{in}}^{*}(j\omega) \tag{7}$$

$$= e^{j\omega \cdot (t_{\text{a}} - t_{\text{b}})} \cdot S_{\text{in}}(j\omega) S_{\text{in}}^{*}(j\omega) \tag{8}$$

$$= e^{j\omega \cdot \Delta t} \cdot S_{\text{in}}(j\omega) S_{\text{in}}^{*}(j\omega) \tag{9}$$

For small time differences Δt the effect on the output phases will be a linear slope with respect to the frequency $\varphi(\omega) = \omega \cdot c_{\text{slope}}$ which will wrap at $-\pi$ and π. For larger Δt the steepness will increase up to a point where, due to the wrapping effect, the original phase differences will be no longer visible.

2.6 Common noise

A common assumption in communications engineering, when studying MIMO-systems, is that reciever noises are uncorrelated.

This thesis does not use this assumtion but instead relies on the presence of a common noise component that can be used to synchronize the receivers.

The corresponding noise model is shown in equation 10, where $s_{\text{r},1}$ ... $s_{\text{r},4}$ are the receiver signals that are composed of the noise-free signals s_1 ... s_4, uncorrelated noises n_1 ... n_4 and a small common noise component n_{c}.

$$\begin{aligned} s_{\text{r},1}(t) &= s_1(t) + n_1(t) + n_{\text{c}}(t) \\ s_{\text{r},2}(t) &= s_2(t) + n_2(t) + n_{\text{c}}(t) \\ s_{\text{r},3}(t) &= s_3(t) + n_3(t) + n_{\text{c}}(t) \\ s_{\text{r},4}(t) &= s_4(t) + n_4(t) + n_{\text{c}}(t) \end{aligned} \tag{10}$$

Possible sources of these common noise components in an SDR system can be:

Passband noise Noise in the desired RF-Band caused by various background-radiation sources, received by the antennas.

Baseband noise Low frequency noise like power-grid radiation or EMI from e.g. noisy LED light fixtures that is coupled into the baseband transmission lines on the receiver PCB due to insufficient shielding.

Common clock noise Noise introduced into the shared clock signal used by the receivers.

Assuming the signals s_1 ... s_4 in equation 10 are filtered out completely the equation is simplified as can seen in equation 11.

$$
\begin{aligned}
n_{\mathrm{r},1}(t) &= n_1(t) + n_\mathrm{c}(t) \\
n_{\mathrm{r},2}(t) &= n_2(t) + n_\mathrm{c}(t) \\
n_{\mathrm{r},3}(t) &= n_3(t) + n_\mathrm{c}(t) \\
n_{\mathrm{r},4}(t) &= n_4(t) + n_\mathrm{c}(t)
\end{aligned} \tag{11}
$$

To prevent the problems stemming from time differences between processed signals, discussed in the previous section, the time offsets have to be calibrated out.

Assuming every noise component has an autocorrelation that is zero for $\tau \neq 0$ and n_1 ... n_4 and n_c are uncorrelated, the correlation between $n_{\mathrm{r},1}$ ... $n_{\mathrm{r},4}$ will have a maxmimum at $\tau = 0$.

The time difference between different signals can thus be calculated and compensated by filtering out the signals and finding the maxmimum correlation between the receiver noises.

2.7 MUSIC

The direction finding algorithm used in this thesis is very basic to simplyfy the implementation. There are also a number of more sophisticated algorithms that provide better direction estimation results, be it by providing better resolution, noise imunity or the ability to separate multiple signal sources occupying the same frequency band.

On of the earliest of these algorithms is MUltiple SIgnal Classification (MUSIC), first published in 1979 by Ralph O. Schmidt [3].

Unlike the algorithm used in this thesis MUSIC does not perform a transformation to the frequency-domain, but performs the direction estimation directly in the time-domain.

2.7.1 Signal compositon

MUSIC assumes, that the analyzed spectrum contains I discrete, narrowband signal sources overlayed by uncorrelated noise.

A delay in one of these narrowband signals can be approximated by a phase shift, as the signal's behaviour closely resembles that of a sinusoid [4, p. 4].

In a multi-antenna arrangement the same signal $s_i(t)$ will be received by the antennas with different amounts of delay. Given, that the delays can be modeled by phase shifts, the vector of received signals $\vec{x}(t)$ can be expressed by the multiplication of the input signal by a vector $\vec{a_i}(\theta)$ that models the relative phase-shifts.

$$\vec{x}(t) = \begin{pmatrix} e^{-j\cdot\Delta\varphi_{i,1}} \\ e^{-j\cdot\Delta\varphi_{i,2}} \\ e^{-j\cdot\Delta\varphi_{i,3}} \\ e^{-j\cdot\Delta\varphi_{i,4}} \end{pmatrix} s_i(t) = \vec{a_i}(\theta) s_i(t) \tag{12}$$

The phase-shifting vector $\vec{a_i}(\theta)$ is called the steering vector, it depends on the geometry of the antenna array and the direction of arrival of the signal i [5].

For multiple signals the model can be expanded by introducing a signal vector $\vec{s}(t)$ consisting of the I narrowband signals and a steering matrix A consisting of the separate steering vectors.

$$\vec{x}(t) = \begin{pmatrix} e^{-j\cdot\Delta\varphi_{1,1}} & e^{-j\cdot\Delta\varphi_{2,1}} & e^{-j\cdot\Delta\varphi_{3,1}} \\ e^{-j\cdot\Delta\varphi_{1,2}} & e^{-j\cdot\Delta\varphi_{2,2}} & e^{-j\cdot\Delta\varphi_{3,2}} \\ e^{-j\cdot\Delta\varphi_{1,3}} & e^{-j\cdot\Delta\varphi_{2,3}} & e^{-j\cdot\Delta\varphi_{3,3}} \\ e^{-j\cdot\Delta\varphi_{1,4}} & e^{-j\cdot\Delta\varphi_{2,4}} & e^{-j\cdot\Delta\varphi_{3,4}} \end{pmatrix} \cdot \begin{pmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \end{pmatrix} = \boldsymbol{A}\vec{s}(t) \tag{13}$$

In an actual system the received signals $\vec{x}(t)$ will also each be superimposed by noises $n_1 ... n_4$.

$$\vec{x}(t) = \boldsymbol{A}\vec{s}(t) + \vec{n}(t) \tag{14}$$

With full knowledge of the steering-matrix $\boldsymbol{A}$ the directions of arrival could be inferred for every received signal. Unfortunately the noise vector $\vec{n}$ and the original signals $\vec{s}$ are unkown at the receiver prohibiting a direct calculation of the coefficients of $\boldsymbol{A}$.

2.7.2 Signal decomposition

To separate the noise and signal subsignals MUSIC makes use of a correlation Matrix that estimates the correlation between the received signals, and principal component analysis to extract the main sources ofcorrelation [6] (the signals).

A MUSIC implementation keeps track of the received signal correlations $\rho_{i,j}$ in a correlation matrix R.

$$\boldsymbol{R} = \begin{pmatrix} 1 & \rho_{1,2} & \rho_{1,3} & \rho_{1,4} \\ \rho_{2,1} & 1 & \rho_{2,3} & \rho_{2,4} \\ \rho_{3,1} & \rho_{3,2} & 1 & \rho_{3,4} \\ \rho_{4,1} & \rho_{4,2} & \rho_{4,3} & 1 \end{pmatrix}, \quad \text{where} \rho_{i,j} = \frac{\operatorname{Cov}(x_i, x_j)}{\sqrt{\operatorname{Var}(x_i)\operatorname{Var}(x_j)}} \quad [7] \tag{15}$$

As the R matrix is non-negative definite [4, p. 9] it can be transformed into a diagonal form using principal component anaylsis by finding the eigenvectors and -values [2, p. 325].

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^H \tag{16}$$

The decomposition of R is shown in equation 16, where the diagonal elements of $\boldsymbol{D}$ are the eigenvalues of $\boldsymbol{R}$, and $\boldsymbol{U}$ consists of its normalized, corresponding eigenvectors. $\boldsymbol{D}$ and $\boldsymbol{U}$ should be sorted in a way, that the eigenvalues are descending in value.

For a number of signals I smaller than the number of receivers N, the first I eigenvalues will correspond to the signal strengths of the signals and the remaining $N - I$ eigenvalues will be low as they correspond to the correlation of the noise.

This means, that as long as the number of signals is known and smaller than the number of receivers the diagonal matrix $\boldsymbol{D}$ can be separated into a part corresponding to the signals (large eigenvalues) and a part corresponding to the noise (small eigenvalues).
MUSIC uses informations from the noise-subspace (small eigenvalues and corresponding eigenvectors) to extract DOA informations [8], other algorithms, like e.g. ESPRIT, use the signal-subspace for DOA-estimation.

To extract direction information from the noise-eigenvectors a test steering-vector $\vec{a_t}(\theta)$, that contains the expected phase-shifts based on the direction of incidence θ and the array geometry, is constructed.
To calculate the MUSIC estimation of the signal strength for this direction θ the evaluation function 17 is used [4, p. 12] where $\vec{U_n}$ is the nth (noise-)eigenvector of R.

$$P(\theta) = \frac{1}{\sum_{n=0}^{N-I} \left(\vec{a_t}(\theta) \cdot \vec{U_n} \right)^2} \tag{17}$$

The results of this evaluation function over the direction of incidence θ form a "spatial spectrum", that exhibits peaks for directions where a signal is impinging on the array.

This estimation works by exploiting the orthogonality of the eigenvectors of R as it implies, that the space spanned by the signal eigenvectors has to be orthogonal to the space spanned by the noise eigenvectors. Thus the scalar dot product of the steering vector $\vec{a_t}(\theta)$ and a noise eigenvector $\vec{U_n}$ will be minimal if $\vec{a_t}(\theta)$ is in the space spanned by the signal eigenvectors.

The evaluation function calculates the dot product for every noise eigenvector and returns the multiplicative inverse so that a $\vec{a_t}(\theta)$ in the signal subspace yields a large result.

3 Hardware modifications

Figure 3 shows a block diagram of one of the receiver dongles that where used in this project. The two main components on the dongle are a `R820T2` tuner IC and a `RTL2832U` DVB-T receiver IC.

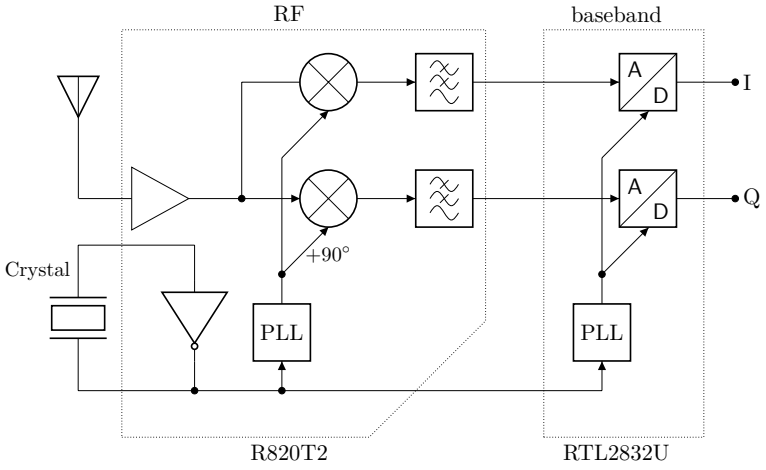

Figure 3: Block diagram of an unmodified SDR-Dongle

The tuner IC contains circuitry to drive an external crystal for clock generation. The crystal used on the hardware at hand operates at a frequency of 28.8 MHz. The tuner uses this frequency to derive the local oscillator signal used for mixing down the RF-signal using a phase-locked loop (PLL). It also provides a clock output pin that outputs a triangle waveform at the crystals oscillation frequency [9].

The given USB-dongles use this clock output signal as a clock input for the `RTL2832U` receiver IC, that handles USB-communication and analog to digital conversion, using an 8-bit analog to digital converter (ADC) . The ADC sampling frequency is also derived from the tuner clock output by using a phase locked loop.

In the basic setup four SDR dongles are used to operate the antenna array. In order to be able to compare the signal phases received by these dongles they have to be tightly synchronized in sample-aquisition time and frequency.

Gaining sufficient synchronization for DOA applications by compensating for hardware impairments completely in software was deemed unfeasible, so a hybrid approach was chosen that requires minimal hardware modifications and offloads most synchronization tasks to the software domain. As demonstrated in 2013 by Juha Vierinen for coherent radar applications [10].

Figure 4 shows a schematic representation of the performed modifications. The clock output of one tuner IC is daisy chained to the clock input of the next tuner IC. This way all tuners derive their clock frequency from the same crystal, while the load on any of the clock outputs is kept minimal.

To provide the input pin with the correct signal levels a DC-blocking RC network has to be used, as specified in the datasheet [9] and shown in figure 4.

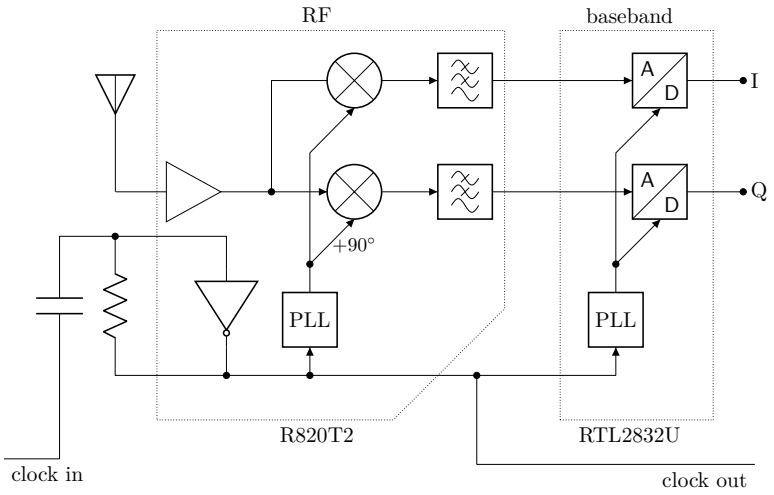

Figure 4: Block diagram of a chained SDR-Dongle

The modifications, as performed on the hardware, can be seen in figure 5. The dongle that contains the clock source for the rest of the system can be seen in the top left image, it still contains the original 28.8 MHz crystal and was only equipped with thin wires that tap off the clock out signal and a ground reference.

The next image to the right shows a dongle in a later position in the clock distribution chain. The original crystal was removed an replaced with the RC-network shown in figure 4. The network is fed by the previos dongle in the chain.

The bottom right image in the figure shows the locations of the main components on the dongles. The signal path on the dongles is from the top right to the left. The signal enters the receiver through the antenna connector at the top right, is mixed-down by the tuner-IC and digitized by the ADC on the left.

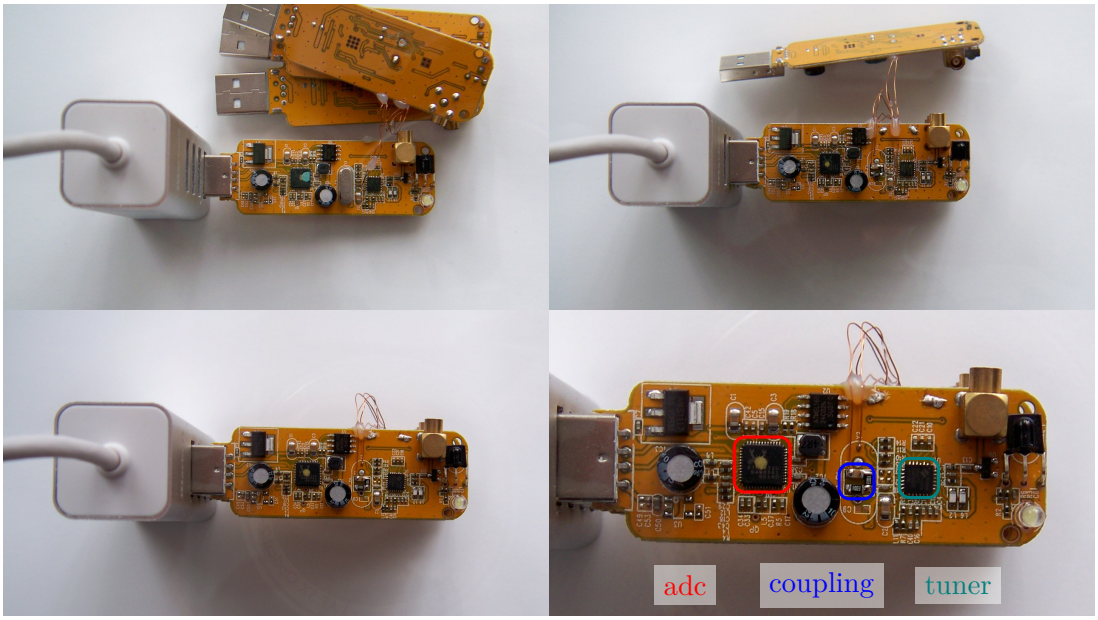

Figure 5: Hardware modifications to synchronize the clocks

4 Synchronization

In addition to frequency locking, that is performed in hardware, additional synchronization steps are performed. These steps are now presented in the order they are performed.

4.1 Sample-exact time compensation

The first offset that needs to be compensated stems from the fact, that the SDR-dongles do not start recording samples at the same time. This effect is demonstrated in figures 6 to 8.

Figure 6 shows the times at which samples were taken and put into the sample-buffer of the corresponding receiver, along with their index in the sample buffer.

In a more realisitic diagram the differences between the times at which the first samples are recored would be a lot bigger compared to the sampling rate, as the differences in startup time are usually in the range of hundreds of milliseconds and the sampling rates are in the order of a couple kilohertz to a few megahertz.

Upon reading the samples from the device-buffers the information considering the absolute aquisition time is lost. To a program reading the ADC data the stream of samples thus look like in figure 7.

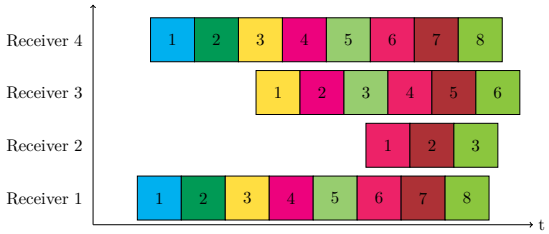

Figure 6: Samples and the actual time they were taken

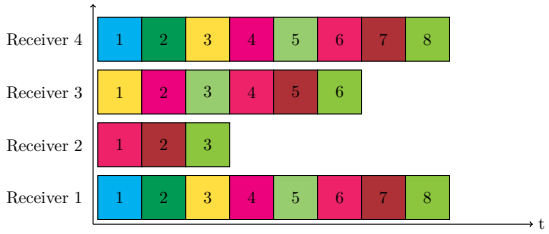

Figure 7: Sample times when read from the device

In order to be able to analyze phase differences between the signals later on, the receivers have to be synchronized to an accuracy of within a couple sampling times.

The necessary synchronization is implemented by discarding all samples in the device buffers that were recorded before every device was recording samples.

This step is demonstrated in figure 8. Receiver two was the last receiver to start aquiring samples, thus every sample recorded before receiver two was aquirirng data was discarded.

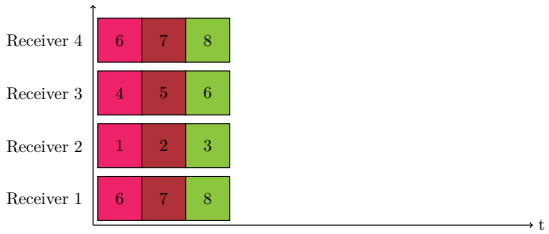

Figure 8: Samples after sample-exact synchronization

In order to perform this sample-exact synchronization one of the receivers is used as a reference. The other receivers are synchronized to this reference by calculating the cross-correlation of the samples between the receiver and the reference. And discarding samples until the cross-correlation peak occurs at a time-offset of zero.

To facilitate calculating the cross correlation over more than 100 k samples (a conservative upper bound of the sample offsets), a fast fourier-transformation is used to

transfer the calculation into the frequency domain. The transformed cross-correlation calculation is demonstrated seen in equation 18, where $\mathcal{F}$ denotes the fourier transformation, $\mathcal{F}^{-1}$ is the inverse fourier transformation and an asterisk marks the complex conjugate.

$$f \star g = \mathcal{F}^{-1}\left(\mathcal{F}(f)^* \cdot \mathcal{F}(g)\right) \tag{18}$$

The calculation above actually yields a cyclic cross-correlation [11, p. 329], so some precautions have to be taken when determining the position of the peak for synchronization.

This compensation step only has to be performed once upon startup, as the sampling clocks are derived via PLLs from the same base clock (due to the hardware modifications). Thus, as long as the PLLs stay locked, the ADCs will produce samples at the same rate and will not drift apart.

4.2 Sub-sample time compensation

As figure 6 shows, the differences in sample time between the receivers do not have to be integer multiples of the sample aquisition durations.

This implies, that the sample-exact time compensation above can not provide complete sample timing compensation, as a fractional difference might still remain after synchronization.

Compensating these offsets in the time domain would require fractionally resampling the incoming data streams in order to be able to shift the signals by less than one sample.

Instead, the input signals are now transferred to the frequency domain. Figure 9 shows the steps that are performed for every receiver pair prior to the sub-sample synchronization.

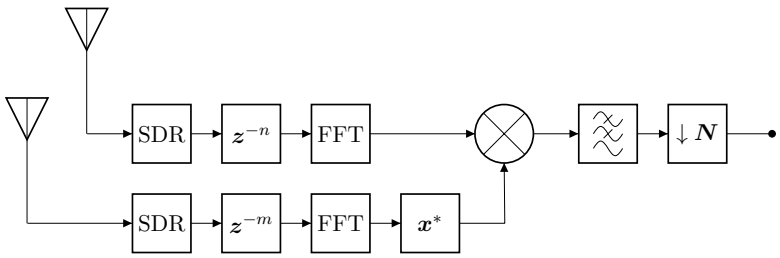

Figure 9: Block diagram of the preporcessing chain for two receivers

The input streams from the receivers are first delayed by the number of samples determined in the previous synchronization step, so that samples recoreded at the same time are in the same position in the streams.

Next the samples are transformed to the frequency domain using FFTs. For one of the output vectors of the two FFTs the element-wise complex-conjugate is calculated.

The resulting frequency-domain vectors are then multiplied element-wise. This complex conjugate multiplication yields a vector of complex numbers which contain the phase differences for the observed frequencies in their argument. As shown in equation 19.

$$\begin{aligned} \boldsymbol{z}_1 \cdot \boldsymbol{z}_2^* &= |\boldsymbol{z}_1|\, e^{\arg(\boldsymbol{z}_1)} \cdot |\boldsymbol{z}_2|\, e^{-\arg(\boldsymbol{z}_2)} \\ &= |\boldsymbol{z}_1| \cdot |\boldsymbol{z}_2|\, e^{\arg(\boldsymbol{z}_1) - \arg(\boldsymbol{z}_2)} \\ &= A^2 \cdot e^{\Delta\varphi} \end{aligned} \tag{19}$$

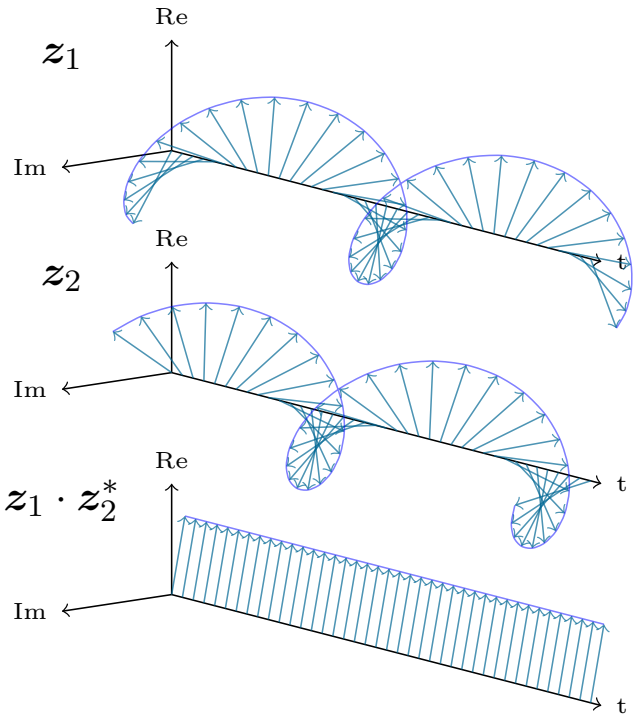

Figure 10: Complex conjugate multiplication of phase shifted signals

The effect of this operation on the same complex signal shifted in phase is shown in figure 10

In order to reduce the processing load on the following stages, the resulting complex values are then low pass filtered by averaging and subsequently decimated.

As these operations are performed for every pair of antennas in the array, n_{ant} input signals are mapped to $(n_{\text{ant}} - 1)!$ output signals. The rig at hand uses four antennas that produce six difference-signal streams.

Figure 11 shows the mean absolute values of the resulting complex output vectors for an actual measurement in the FM-Radio band. The peaks in the diagram indicate the presence of a transmitter at the corresponding frequency.

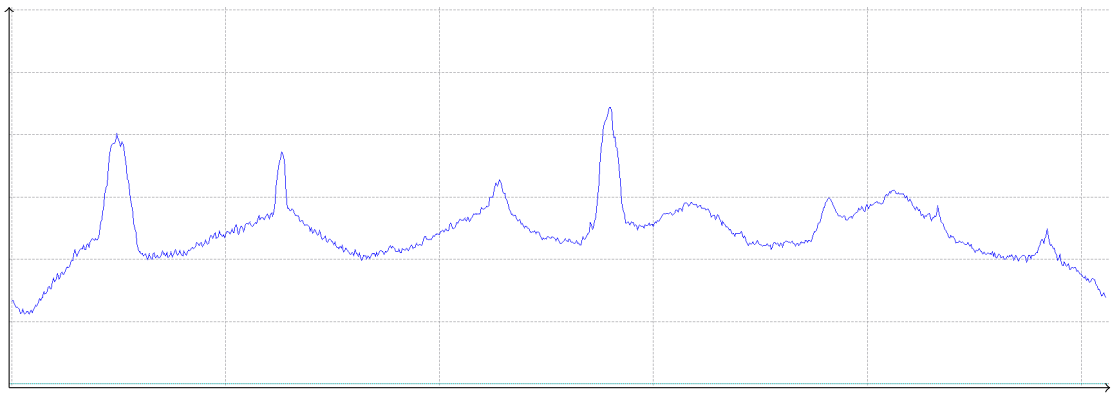

Figure 11: Mean magnitude spectrum for four receivers

Figure 12 shows the phases of the six complex output vectors. When correlating the diagram with figure 11 one can already see some bumps in the phase differences where there are strong transmitters.

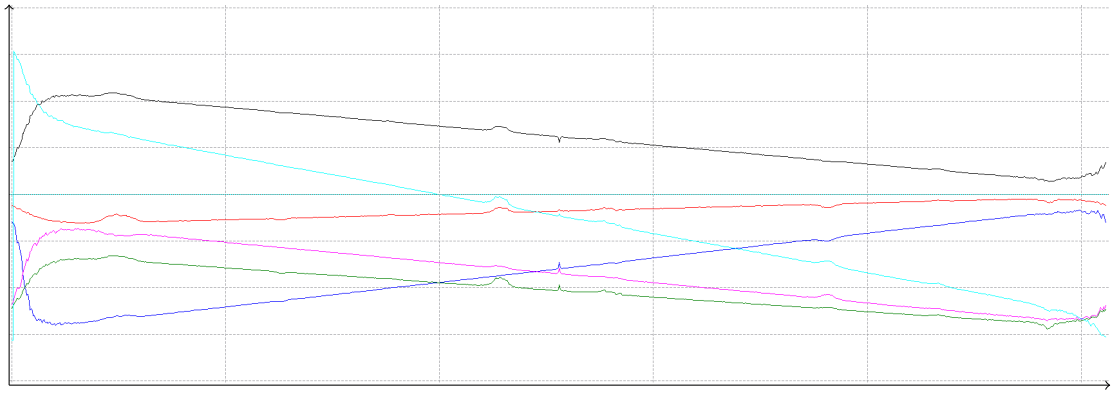

Figure 12: Raw phase differences for four receivers

One can also see a rather steep sloping of the phases, this is due to the sub-sample timing offsets between the ADCs, to compensate them a linear correction is applied to the phase. The resulting sample time compensated phase diagram is shown in figure 13.

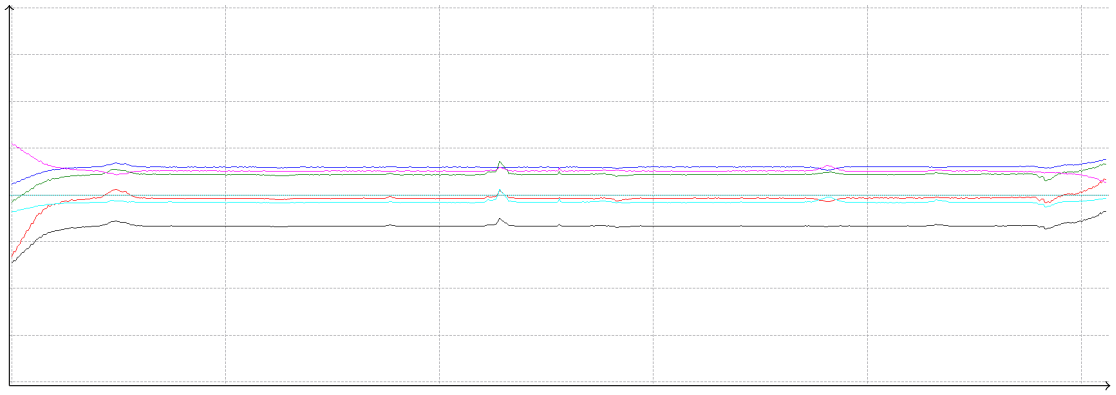

Figure 13: Phase differences after sample time offset compensation

4.3 Tuner phase offset compensation

Figure 13 shows a flattened phase diagram, but there are still constant offsets. These offsets are caused by different phases of the LO-signals used to mix down the RF-signals in the tuners.

Figure 14 shows the phases after compensating for these offsets. Frequency ranges where the noise is dominant show no phase differences between the receivers, while ranges with active transmitters contain visible bumps. When zoomed in like in figure 15 these bumps become a lot more prominent.

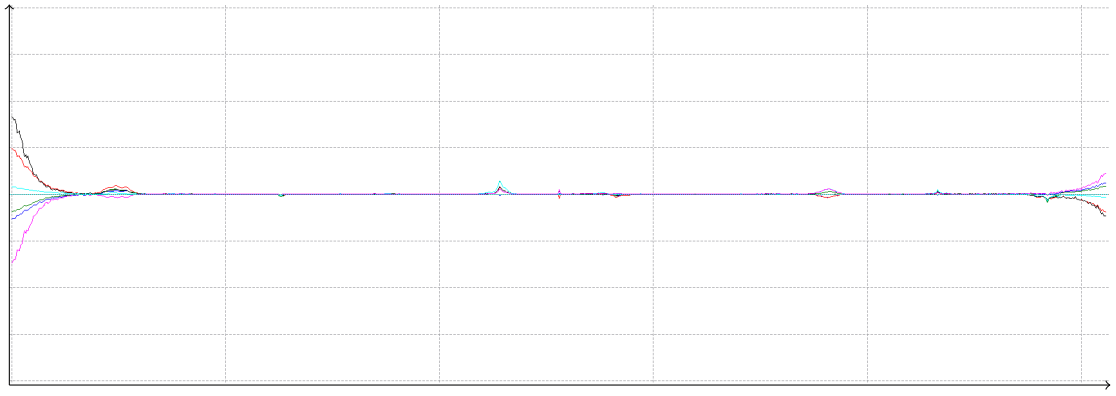

Figure 14: Completely compensated phase diagram

Figure 15: Completely compensated phase diagram with different scaling

The relative phase differences will later be used to determine the direction of arrival of the incoming signals.

5 Software implementation

The implementation is split into two software parts, one implemented in C, that performs basic signal processing on the high-throughput input streams, and one implemented in Python, that performs direction estimation at a lower speed.

5.1 `libsofi.c`

As the preprocessing code has to deal with samples before the first downsamling stage, the data rates to be processed are rather high (about $2\,\mathrm{MS\,s^{-1}}$ per receiver). In order to be able to run on low-range computing hardware this part of the processing chain is written in C and uses the highly optimized `fftw3` [12] and `volk` [13] software-libraries, for accelerated FFT calculations and vector operations.

The program flow starts off by initializing the SDR dongles. The communication with the dongles is performed by using a hardware abstraction layer for software-defined radio devices, that was introduced into recent versions of the Linux kernel [14]. This means, that the program should work with all SDR devices supported in the Linux kernel, as long as their clock sources are synchronized.
The actual transfer of samples is performed using buffers, that are shared between the application and the kernel, to minizize the number of copy operations performed [15].

To better utilize the processing power of multi-core processors FFT calculations are performed in one thread per SDR device, distributing the load over the available cores.
Once the SDR devices are initialized and acquiring samples they are synchronized in time by calculating the cross-correlation between the per-device streams and discarding samples until the peak appears at a sample-offset of zero.

Every operations after the synchronization phase is performed in the frequency domain.

After the sample-exact synchronization the processing threads, that take the time domain samples from the shared buffers and transfer them into the frequency domain, are started.
For every receiver m the FFT calculation yields phase and magnitude information for N_{FFT} frequency bins. The values $i_{\mathrm{m,n}}$ in these frequency bins can not be averaged directly, as the current signal phase can be influenced by signal modulation and thus behaves randomly. When these stochastic samples are averaged over a sufficiently long timespan they would thus cancel out to zero.

In order to be able to perform the necessary averaging and downsampling, instead of using the absolute phases directly from the FFT, the phase differences between the receivers are calulated prior to the averaging step, using complex conjugate multiplication. For four receivers this leads to six difference signals, that are calulated according to the

equations shown below:

$$o_{1,\mathrm{n}} = i_{1,\mathrm{n}} \cdot i^*_{2,\mathrm{n}} \qquad o_{4,\mathrm{n}} = i_{2,\mathrm{n}} \cdot i^*_{3,\mathrm{n}}$$
$$o_{2,\mathrm{n}} = i_{1,\mathrm{n}} \cdot i^*_{3,\mathrm{n}} \qquad o_{5,\mathrm{n}} = i_{2,\mathrm{n}} \cdot i^*_{4,\mathrm{n}}$$
$$o_{3,\mathrm{n}} = i_{1,\mathrm{n}} \cdot i^*_{4,\mathrm{n}} \qquad o_{6,\mathrm{n}} = i_{3,\mathrm{n}} \cdot i^*_{4,\mathrm{n}}$$

The resulting values $o_{1,\mathrm{n}}$ ... $o_{6,\mathrm{n}}$ are then averaged over time to reduce the sample-rate and forwarded to the next processing step.

5.2 `libsofi.py`

After the downsampling is performed in the backend code the data rate is reduced by a factor of more than 100. Further processing can thus be performed in Python, an interpreted scripting language that, in conjunction with numpy, a software library for numeric calulations, makes writing DSP task easier than expressing the algorithms in C.

The data-transfer from C to Python is performed using a bit of glue-code, that uses the Python `ctypes` foreign-function-interface (FFI). This glue-code allows the C backend to be imported into the Python code just like a native Python module.

5.3 `direction.py`

The direction estimation code uses the `numpy` [16] and `scipy` [17] libraries for easy and efficient vector operations and signal processing tasks.

This part of the processing chain operates completely in the frequency domain, it is responsible for performing the compensation tasks to get from the raw phase diagram, as seen in Figure 12, to the fully compensated phase diagram, as seen in Figure 15.

It also uses these phase diagrams to calulate direction informations for the signal sources in the spectrum.

The compensation works by first finding the local minima in the amplitude spectrum, as seen in figure 11, these are the frequencies with the lowest signal strength and can thus be assumed to be dominated by noise.

As discussed before noise carries no direction information and the phase difference between the receivers should thus be, on average, zero.

The actually measured phase differences at the amplitude minima is used as an error input to PID-controllers that are configured to compensate for the per-receiver mixer phase offsets and sub-sample acuisition time differences. The controllers constantly optimize the compensation parameters to zero out the phase differences at the amplitude minima.

After compensation the phase spectrogram looks mostly flat with destinct deviations for frequencies with strong signals, as can be seen in figure 15.

In addition to finding the frequencies, that are dominated by noise, the program also finds peaks in received signal strength. These peaks correspond to the parts of the spectrum, that are occupied by transmitters. These frequencies are then added to a list of sources to be analyzed.

As the direction estimation model depends on the signal wavelength, the program generates a modelling matrix for every frequency that is associated with a distinct signal source.

The code, that generates these matrices, can be seen in listing 1. The snippet creates a vector consisting of possible input direction of arrival angles (from $-\pi$ to π in radians) and calculates the expected phase difference for each antenna pair based on the antenna pair's orientation, distance, input angle and signal wavelength.

Listing 1: Code fragment that calculates the receiver model

```
1     # Calculate "relative wavelengths" (in radians)
2     # based on the antenna distances in the
3     # edge_distances vector and the actual signal wavelength
4     rel_wl= 2 * np.pi * edge_distances / wavelength
5
6     # Generate a vector of length self.dir_len and populate
7     # it with incidence angles from -pi to pi
8     test_angles= np.linspace(-math.pi, math.pi, self.dir_len)
9
10    # Generate an empty target matrix
11    pos_mat= np.zeros((self.dir_len, self.edges_count))
12
13    # Go through every incidence angle and populate
14    # the corresponding row with the expected phase
15    # difference
16    for (idx, test_angle) in enumerate(test_angles):
17        rel_angles= edge_angles + test_angle
18        pos_mat[idx, :]= rel_wl * np.sin(rel_angles)
```

Performing a matrix-vector multiplication of this modeling matrix and a vector consisting of the measured phase differences is equivalent to calculating the scalar dot product of the modeled phase difference vector and the actually measured phase difference vector for every input angle. The code implementing this calculation is thus very short, as shown in listing 2.

Listing 2: Code fragment that calculates the DOA estimation

```
1       # Get a test matrix according to the
2       # wavelength
3       pmat= self.get_position_matrix(wl_start, wl_end)
4
5       # Perform a matrix-vector multiplication
6       # of the test matrix and the measured phases
7       return(pmat @ phase_vector)
```

As the scalar dot product yields its largest result when both vectors are in parallel and its smallest result when both vectors are anti-parallel the matrix-vector multiplication yields a vector, where the position of the largest value corresponds to the angle of incidence of the signal.

5.4 `sofi_ui.py`

The Visualization of the direction estimation is performed using the `matplotlib` and `Gtk+` software-libraries.

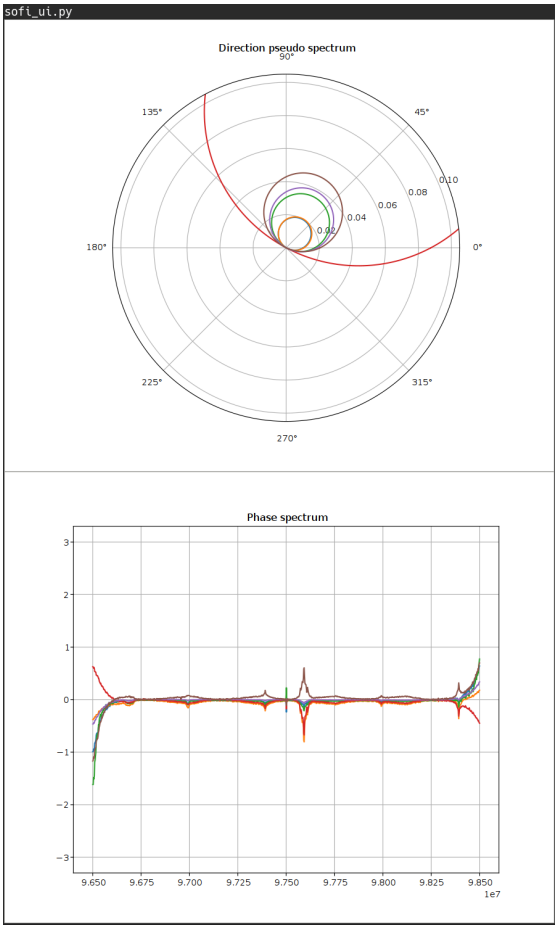

Figure 16: The SoFi user interface

Figure 16 shows the SoFi interface. The bottom half displays the phase spectra and the top half displays the direction estimation data.

The diameter of the circles in the top plot correspond to the signal strengths of the analyzed signals. The orientations of the circles indicate the direction of arrival of the corresponding signals.

6 Conclusion

The planned goal of this thesis was to explore if a a low-cost DOA rig could be built by utilizing comodity SDR receivers and to analyze if common background-noise can be used as a sufficient means of synchronization for DOA applications.

To check these hypotheses a hardware-setup was assembled, consisting of four frequency-locked rtl_sdr receivers and the four corrseponding antennas spread out on a plane.

Software components were written that used cross-correlations of receiver data-streams to synchronize samples in time and performed further compensations to mitigate hardware impairments stemming from the independent receivers.

Finally a basic DOA algorithm was implemented that utilized an FFT to separate the analyzed spectrum into multiple frequency bins. The phase differences of the receivers in these frequency bins were used directly to derive the direction of arrival of the impinging signals.

Within the processing time of this thesis no conclusion could be found on whether a working rig could be implemented with the specified constraints.
The resolution and accuracy of the measured DOA results were rather bad. It is, as of now, unkown if this was due to the bad performance of the algorithm used, other implementation errors or a conceptional problem of the receivers used.

Following are some aspects of the system that were determined to be working as expected:

- The hardware modifications to lock the input clocks to the same source.
  There was a remaining slow drift in the signal phase, that was assumed to be most likely due to jitter in the LO-frequency generating PLLs. But, due to this drift being slow, it can easily be compensated by the offset-compensating PID-controllers in the frequency domain.
- The sample-exact time compensation worked as expected and only required a few iterations to sucessfuly lock onto the correct time-offset. To within sufficiently few samples.
- The sub-sample time compensation and tuner-phase offset compensation were assumed to be mostly working. When the number of found noise-points upon startup were too small, the controllers failed to lock correctly, as the algorithm only tried to minimize the offsets at these noise-points. The results of a failed lock can be seen in figure 17.

Possible reasons for the bad DOA performance include multi-path effects in the tested frequency range at about 100 MHz, a conceptually bad performance of the DOA-estimation

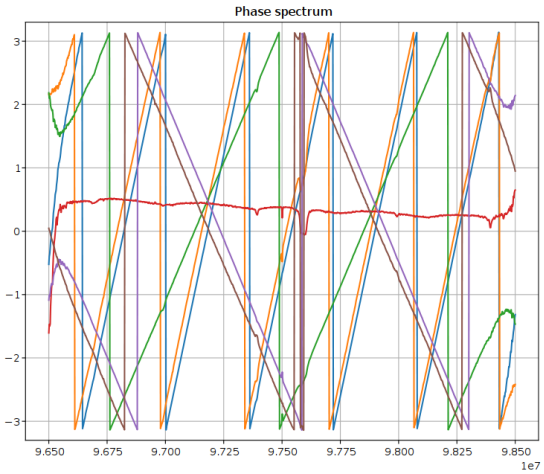

Figure 17: A failed sub-sample compensation lock

algorithm used or implementation mistakes.

It is also possible that, due to their proximity, the antennas in the array interfered and acted as a single antenna with unkown and possibly problematic characteristics.

The synchronization by using common background noise was also not well characterized. Although the compensated phase diagrams looked well-formed there might have been a residual systematic compensation error, distorting the direction estimation.
A setup as demonstrated in rtl_coherent[18], that utilizes a dedicated wideband noise-source, would have provided a more solid base to independently characterize the different procesing steps.

To test the DOA-algorithm a simulator was written, that simulated a spectrum including a few transmitters and independant noise but no multi-path effects. The simulator replaced the SDR devices in the SoFi processing chain.
In the simulated case the SoFi DOA estimation worked as expected, indicating that multi-path effects were indeed a problem or that the simulation was otherwise not accurate enough.

Due to the simulation only simulating independant noises, as there was no reliable model for the assumed common noise term, the sychronization procedures had to be partially disabled. The simulation could thus also not be used to validate the synchronization code.

7 Outlook

In the initial SoFi implementation I tried to avoid as many pitfalls as possible, I stuck to programming languages and libraries I already knew fairly well and preferred algorithms and implementations that felt intuitive rather than relying on well studies approaches presented in academic publications.

After gaining some experience using that approach I would structure a second implementation quite differently.

Firstly I would ditch most of the custom C and Python code and replace most of the signal processing code with implementations from the GnuRadio framework.
The primary concerns for not choosing GnuRadio in the first place were lacking support for current versions of Python (Python 3.X) and a steep learning curve, especially with respect to writing custom processing steps. Nevertheless the flexibility of quickly plugging in filters and other processing steps, that are known to work, far outweigh the initial time spent getting used to the framework.

Secondly I would introduce a dedicated signal source for synchronization purposes, possibly a wideband noise source as demonstrated in rtl_coherent [18], as relying on characteristics of background noise introduces a lot of unkowns, making characterization of the results and constructing an usefull simulation quite difficult.
A well characterized reference source would also allow characterizing the current approach of using background noise as a means of synchronization by comparing the end-to-end quality of the DOA estimation results.

I would further use the MUSIC algorithm in a new implementation, as it is well studied and known to provide better results with fewer antennas than more naive algorithms, like the one used in this thesis [6].

As working with narrowband signals makes the compensation of hardware impairments and using MUSIC easier I would still employ an FFT, as shown in figure 18.
The system in the figure uses an array of two antennas and analyzes the DOA of signals in four frequency bins.

The FFT is used as a filter bank to separate the incomming stream into n substreams of $1/n$ the original sample rate.
As the n substreams are narrowband, the phase errors introduced by the LO phase and the sub-sample timing offsets can be compensated by a term $\Delta\varphi(\omega) = \omega \cdot m + b$ as previously discussed in this thesis.

For every substream a cross-correlation matrix of the receivers is calculated and used to estimate the direction of arrival of the signal in the corresponding frequency bin by using the MUSIC algorithm.

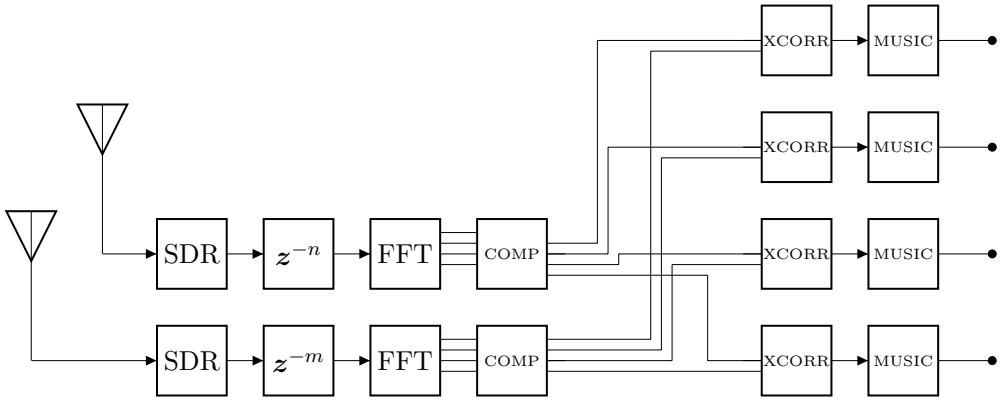

Figure 18: A hybrid FFT and MUSIC based approach

The number of frequency bins should be large enough that one bin is at most occupied by one signal and, that the narrowband assumtion, when compensation phase errors, still holds.

Glossary

ADC analog to digital converter. 2, 12–14, 16, 18

DOA direction of arrival. 2–5, 7, 23, 24, 26, 28

DSP digital signal processing. 22

FFI foreign-function-interface. 22

FFT fast fourier-transformation. 5, 6, 15, 17, 21, 26, 28, 29

I/Q in phase and quadrature. 2, 6

LNA Low noise amplifier. 6

LO local oscillator. 6, 7, 19, 26, 28

MIMO multiple input multiple output). 8

MUSIC MUltiple SIgnal Classification. 3, 9, 11, 28, 29

PCB printed circuit board. 8

PID proportional integral derivative. 26

PLL phase-locked loop. 12, 16, 26

RF radio frequency. 3

SDR software defined radio. 2, 3, 5–8, 21, 26, 27

SNR signal-to-noise ratio. 5

UMTS Universal Mobile Telecommunications System. 5

References

[1] *About RTL-SDR*. URL: `http://www.rtl-sdr.com/about-rtl-sdr/` (visited on 05/28/2017).

[2] K.A. Semendjajew I.N. Bronstein. *Taschenbuch der Mathematik*. Europa Lehrmittel, 2016. ISBN: 978-3-8085-5789-1.

[3] Ralph O. Schmidt. "Multiple Emitter Location and Signal Parameter Estimation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37 (1979).

[4] *An Introduction to MUSIC and ESPRIT*. Tech. rep. GIRD Systems, Inc. URL: http://www.girdsystems.com/pdf/GIRD_Systems_Intro_to_MUSIC_ESPRIT.pdf (visited on 05/28/2017).

[5] Zhizhang Chen, Gopal Gokeda, and Yiqiang Yu. *Introduction to Direction-of-arrival Estimation*. Artech House Signal Processing Library. Artech House, Inc, 2010. ISBN: 9781596930896. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=324516&site=ehost-live.

[6] M. Viberg and B. Ottersten. "Sensor array processing based on subspace fitting". In: *IEEE Transactions on Signal Processing* 39.5 (May 1991), pp. 1110–1121. ISSN: 1053-587X. DOI: 10.1109/78.80966. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=80966.

[7] URL: https://de.wikipedia.org/wiki/Korrelationsmatrix (visited on 05/28/2017).

[8] Vijay K. Madisetti. *The Digital Signal Processing Handbook*. CRC Press, 2010. ISBN: 978-1-4200-4604-5.

[9] *R802T2 Tuner datasheet*. Tech. rep. Rafael Micro. URL: http://rtl-sdr.com/wp-content/uploads/2013/04/R820T_datasheet-Non_R-20111130_unlocked.pdf (visited on 05/31/2017).

[10] Juha Vierinen. Sept. 26, 2013. URL: http://kaira.sgo.fi/2013/09/16-dual-channel-coherent-digital.html (visited on 06/03/2017).

[11] Kristian Kroschel Karl-Dirk Kammeyer. *Digitale Signalverarbeitung*. Springer Vieweg, 2012. ISBN: 978-3-8348-1644-3.

[12] URL: http://www.fftw.org/ (visited on 06/05/2017).

[13] URL: http://libvolk.org/ (visited on 06/05/2017).

[14] *V4L SDR Interfaces - Linux Kernel documentation*. Tech. rep. The Linux kernel Developers. URL: https://www.kernel.org/doc/html/v4.10/media/uapi/v4l/dev-sdr.html (visited on 06/03/2017).

[15] *V4L Streaming I/O - Linux Kernel documentation*. Tech. rep. The Linux kernel Developers. URL: https://www.kernel.org/doc/html/v4.10/media/uapi/v4l/mmap.html#mmap (visited on 06/03/2017).

[16] URL: http://www.numpy.org/ (visited on 06/05/2017).

[17] URL: https://www.scipy.org/ (visited on 06/05/2017).

[18] Github user tejeez. *rtl_coherent, Synchronized RTL-SDR receivers and direction finding*. URL: https://github.com/tejeez/rtl_coherent (visited on 05/28/2017).